3D Object Tracking: From Front to Surround View

Emran Yasser Moustafa*†, Hidde J-H. Boekema†, Julian F. P. Kooij†
*Trinity College Dublin
†TU Delft

***Abstract*—Typical 3D auto labelling pipelines rely on the need for fully labelled training data. Moreover, they tend to focus on the tracking and labelling of vehicles in a controlled driving environment. In this work, we present a novel auto labelling pipeline for extending the View-of-Delft (VoD) dataset, an autonomous driving dataset recorded in the historic city of Delft. Our pipeline proposes two significant contributions. The first is it demonstrates a method for extending partially labelled datasets. In the case of VoD, we stitched together predictions made on a set of incrementally rotated point clouds in order to get a final "surround view" prediction. Second, it explores the application of popular tracking and auto labelling techniques in complex, urban driving environments. In this work, we tracked pedestrians, cyclists and cars with a detection F1 score/HOTA score of 0.462/37.5, 0.469/49.2 and 0.405/49.7, respectively. Our framework also increased the number of tracked objects and unique bounding boxes in VoD by 148.9% and 71.9%, respectively**

I. Introduction

DATA has proven to be a key component in the development of deep learning based systems [1]. One such area in which this has proven to be true is the field of autonomous driving. The need for large amounts of labelled data has proven challenging to researchers and those in the industry. In order to save both the time and monetary costs of manually labelling data, many autonomous driving companies have begun to develop their own auto labelling pipeline [2], [3]. These systems can also rival, if not outperform, human annotators in terms of accuracy in many cases [2], [4].

The challenge of building an auto labelling pipeline is typically broken down into two/three different challenges [2], [3], [4]; detection, tracking and refinement. The methods and techniques for each of these stages are different depending on the dataset used, resources available and target application. LiDAR-based auto labelling pipelines [2], [3], [4] are especially useful in the domain of autonomous driving. In these frameworks, a timestamped set of orderless point clouds are input to the system and a set of corresponding object tracks are output. The offline nature of these systems allows for information from the full sequence to be leveraged in all parts of the pipeline, such as aggregating points to produce better bounding box predictions [3], improved trajectory estimation [4], [5] or the ability to join and smooth fragmented tracklets [6], [7].

Although perfectly annotated data is desired, the use of rough, machine automated "pseudo-labels" has gained popularity in fields such as speech recognition [8], image classification [9] and autonomous driving [10]. This idea relates to the technique of training a model on labelled data which can then be used to label previously unlabelled data, creating what is referred to as pseudo-labelled data. This approach is especially useful in applications where the amount of available data is low or hard to obtain/label. In this work, we will use the terms "auto labels" and "pseudo labels" interchangeably, both terms used to describe labels that have been obtained via automated, machine annotation.

An example is the View-of-Delft (VoD) dataset [11]. This dataset consists of 8693 frames of labelled driving data recorded in the historic city of Delft, The Netherlands. In this work, we propose an offline object detection and tracking pipeline (see Fig. 1) that could be used to extend this dataset, both in providing a solution for obtaining rough annotations for previously unlabelled data, as well as in extending the labelled area around the vehicle, from just in the front to 360 degrees around the vehicle.

We believe our pipeline presents a practical solution to extending pre-existing datasets as well as generating new pseudo-labelled data from raw LiDAR point clouds.

II. Related Work

In this section we will discuss the the body of literature that exists around the three major themes of this work; 3D object detection, 3D multiple object tracking and 3D auto labelling. We are especially interested in the application of these topics in the domain of autonomous driving, particularly in a dense urban environment.

A. 3D Object Detection

Light Detection and Ranging (LiDAR) has proved to be useful sensor in autonomous driving applications. There has been many different approaches to the challenge of 3D object detection, especially in recent years. The two primary issues faced when dealing with point clouds are the sparsity and unordered nature of the data [12]. To deal with this, the data is typically categorised into a different representation. Voxel-based grids [13], [14], [15] are sometimes used to simplify this problem. A popular example of this is VoxelNet [16]. This network utilised PointNets [12] to encode features for each voxel, which can then be used later in the network. Models such as SECOND [17], which utilised sparse 3D convolutions, improved on this architecture by both improving performance and latency.

Similar to the voxel based approach, and especially in contexts where the assumption is made that objects are not above or below each other, points can be assigned to vertical pillars [18], simplifying the representation problem. In some works [19], [20], [16], all the points are projected to the ground plane and the point cloud is represented in a "birds-eye-view" manner. 2D convolution based techniques can then easily be applied to data in this form.

Object detection networks can be classified as one-stage or two-stage detectors. Typical single-stage detectors [21], [5] regress and classify anchor boxes in a single stage. This is in contrast to two-stage detectors [22], [23], [24], which propose bounding boxes in the first stage, then refining and classifying these bounding boxes in the second stage. PointPillars [18] is a popular one-stage detector. This model uses a pillar-based point cloud representation as input. It then employs a convolution section and detection head to encode and decode point features, outputting a bounding box prediction at the last layer of the model. CenterPoint [25] is a popular two-stage detector. In this model, a voxel-based or pillar-based point cloud representation is first encoded using a 2D CenterNet [26]. The resulting heat map representations are used to regress 3D bounding boxes for each predicted object, concluding the first stage of the network. The second stage of the network then further extracts features to then refine and score the final set of bounding boxes.

B. 3D MOT

In 3D multiple object tracking (MOT), a set of detections are provided to the system and a set of tracks are output, representing objects classes, position and motion through time. AB3DMOT [27] presented a baseline for 3D object tracking. This system took 3D object detections and used a simple Kalman filter [28] in conjuction with the Hungarian algorithm [29] to track objects in space. In this case and with similar approaches, motion is inferred from the detections and a constant velocity or acceleration is assumed in the transition model.

Likely inspired by the work done in 2D MOT [30], [31], [7], AB3DMOT used 3D intersection over union (IoU) to associate predictions together frame-to-frame. Extensions on this work used Euclidean distance [SimpleTrack citation] between the centre point of predictions. Specifically in the image domain [31], it is common for an object descriptor to be generated and considered when performing associations.

In many tracking frameworks [27], [32], [33], tracks that are "unseen", i.e. with no predictions associated with them for a predefined number of frames, are considered to have left the scene and are deleted from the overall set of tracks. ImmortalTracker [34] extends this lifetime indefinitely, and suggests that this adjustment greatly reduces both ID switches and early termination. Other works [35], [6], [31] chose to instead match current tracklets with past tracklets either via bipartite matching or deep learning on the object or track history.

C. 3D Auto Labelling

A number of different autonomous driving companies, as well as research groups, have proposed their own auto labelling framework for autonomous driving. Most works perform tracking first with the set of detections and a Kalman filter, then perform some post-processing to refine the final set of tracks.

[3] first tracks objects in 3D using a Kalman filter, then refining both the bounding box size and object trajectory using deep learning.

Auto4D [2] and CTRL [36] both classify objects as dynamic or static, dealing with both types differently. The latter auto-labelling framework also proposes a track-centric learning approach. The authors suggest using learned features from the track to better predict the bounding box sizes for a given frame. DetZero [4] takes a similar approach, but uses an attention-based mechanism to fully harness the long-term context of each object.

[6] takes a novel approach to the problem, utilising map information to better track objects in the MOT section of the pipeline and also improve tracklet matching via deep learning.

Finally, there are also a number of semi-supervised auto-labelling frameworks. These works use a mix of deep learning [37], [38] and non-deep learning [39], [40] based methods to reduce the human annotation effort.

These works have show that machine annotated labels can outperform human annotators, both in terms of accuracy and speed [2], [4]. However, they all rely on fully labelled training data. Moreover, they also only focus on the labelling of objects in a controlled rules-based driving environment. In this work we tackle both of these issues, providing a solution to extending partially labelled datasets and exploring auto labelling in a dynamic, complex, urban driving environment.

III. View-of-Delft Dataset

The View-of-Delft (VoD) dataset [11] is a novel autonomous driving dataset recorded in the historic city of Delft. The dataset contains measurements taken from a multi-sensor configuration, consisting of LiDAR, a stereo camera setup, a 3+1D radar, as well as odometry data from both IMU and GPS. The publicly available data is synchronised and provided in KITTI [41] format. The dataset is smaller than other popular autonomous driving datasets [33], [41], [42]. Annotations are also only provided for objects 50 meters from the LiDAR sensor and partially or fully in the camera's field of view (horizontal FoV: ±32°, vertical FoV: ± 22°).

A. LiDAR

The dataset was used a Velodyne HDL-64 S3 LiDAR, recorded at a rate of 10 Hz. The point cloud is ego-motion compensated and each point is described with four features, $[x, y, z, r]$, representing 3D position relative to the LiDAR, (x, y, z), and reflectivity of that point, r. Points are described in LiDAR frame, where the positive x direction is the direction the LiDAR sensor is facing, the positive y direction is left of the sensor and the z direction points upwards from the sensor. 2D orientations in this frame are also described on the xy plane, range from $[-\pi, \pi]$ and the postive direction of

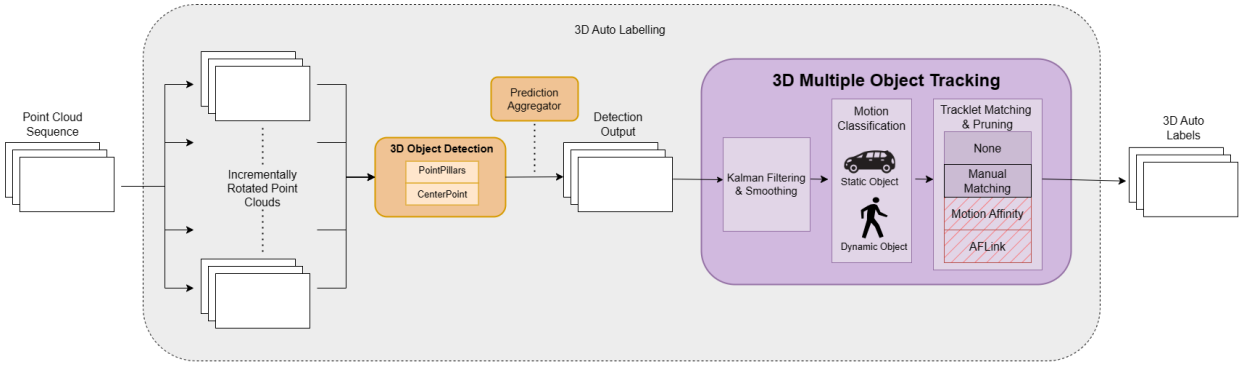

Fig. 1. Overview of our proposed auto labelling framework. The system takes as input a sequence of point clouds. These point clouds are then incrementally rotated about the vertical axis. The 3D object detector than performs inference on each of these point clouds, outputting a set of bounding box predictions for each point cloud. The bounding boxes are transformed back into the correct frame and aggregated into the set of "surround view" detection outputs. Tracking, smoothing and motion classification is then performed to obtain a set of valid tracks. Tracklet matching and pruning can then be performed to obtain the final set of tracks/pseudo labels.

orientation is the counterclockwise direction about the z axis. Annotations are provided in the camera frame, however the transformation from camera to LiDAR frame is provided in the dataset. The transformation from camera frame to a set of world frames, described either by the initial GPS or IMU position, are also provided.

B. Classes

VoD uses 13 unique classes, the distribution of which is shown in Table I. For our auto-labelling framework, we decided to reduce set of classes in the VoD dataset from thirteen to three; considering only the Pedestrian, Cyclist and Car classes. This was done to simplify the detection and tracking problem, and also to align the model with others trained on the KITTI dataset [41]. We intend on expanding the framework to include a subset of dynamic classes (see Sec. VI for more details).

IV. 3D Object Detection

Let $\mathcal{P} = \{(x, y, z, r)_i\}$ be an orderless 3D point cloud, where (x, y, z) represent the 3D positions of points in space and r represent the reflectance of these measurements. 3D object detection aims to predict a set of 3D object bounding boxes $B = \{b^k\}$, where k is the number of objects in a particular point cloud as input. Each bounding box $b = (cls, S, x, y, z, l, w, h, \theta)$ consists of a predicted object class cls (i.e. Pedestrian, Cyclist, Car, etc.), a confidence score S, a centre location (x, y, z), a set of dimensions (l, w, h), and an orientation in 2D θ. The mapping between the input $\mathcal{P}$ and output B can be approximated by data, and is typically learned with the use of a deep learning model.

A. Model

For our object detection backbone, we trained two different popular 3D object detection models; PointPillars [18] and CenterPoint [25].

In the PointPillars networks, the input point cloud is first discretised into a voxel-grid representation using pillars. Pillars are voxels which are extended in the vertical plane so as to encode points of different heights into one feature. There is then an encoding stage of the network, where the network learns high-level features from the pillar representation using a convolutional neural network (CNN). Finally, a detection head performs a bounding box regression to predict the location of objects, their classes and associated confidence scores.

The CenterPoint network is designed in a similar way. This model feeds either a pillar or standard voxel representation into its CNN based encoder. In this work we only considered the pillar based approach. The feature encoding is used to generate a 2D heat map per class in the output. Areas of high activation in these heat maps represent areas in which the target class is likely be, the centre of these areas being the centre point of the object. This data is then used to perform a bounding box regression for object position, dimensions and 2D velocity. A second stage of the network than extracts additional features from the backbone of the network and the output to refine these bounding box predictions.

Both models performed similarly, however we decided to use the PointPillars model as the backbone for our 3D object detection section of our pipeline. Our reasoning for this decision is discussed in Sec. VI.

B. Front View to Surround View

One approach to obtaining predictions 360 degrees around the car would be to use the entire point cloud as input to the model. This approach however provides the model, which has been trained to predict objects in front of the vehicle, with a lot of data that is outside of the distribution of training data. The approach we decided to take was to perform inference on a set of rotated point clouds and stitch the predictions together. In our implementation, we rotate the point cloud in 45 degree increments, meaning we perform inference a total of eight times. We then rotate the predictions back to their original positions and aggregate the bounding boxes into one "surround view" prediction. Finally, we perform a non-maximal suppression step to ensure we have one bounding box for each object that may have been observed twice in the previous steps.

TABLE I
VoD STATISTICS:NUMBER OF ANNOTATED OBJECTS (TOP), NUMBER OF UNIQUE OBJECTS (MIDDLE), AND PERCENTAGE OF MOVING OBJECTS (BOTTOM), PER CLASS.THE RATIOS COMPARED TO THE WHOLE DATASET ARE GIVEN IN BRACKETS

	car	**pedestrian**	**cyclist**	**rider**	**unused bicycle**	**bicycle rack**	**human depiction**	**moped or scooter**	**motor**	**other**	Σ
# objects	26949 (21.9%)	26587 (21.6%)	10800 (8.8%)	12809 (10.4%)	24933 (20.3%)	12025 (9.8%)	370 (0.3%)	5403 (4.4%)	629 (0.5%)	2601 (2.1%)	123106
# unique obj	423 (22.6%)	380 (20.3%)	183 (9.8%)	222 (11.8%)	371 (19.8%)	156 (8.3%)	10 (0.5%)	80 (4.3%)	13 (0.7%)	36 (1.9%)	1875
% moving	7.2%	73.2%	96.1%	95.5%	0.7%	0.0%	0.0%	10.7%	59.9%	34.5%	37.4%

(The "other" column combines the classes *ride_other*, *vehicle_other*, *truck*, and *ride_uncertain*). Table adapted from [11].

V. Tracking

In this work, the tracking framework receives a set of bounding boxes and their associated frame, $B_{1:T} = \{B_1, B_2, ..., B_T\}$. Each frame can be described by $B_t = \{b^k\}$, where k represents the number of predicted objects in frame t. The goal of 3D Multiple Object Tracking (MOT) is to take this input and produce a set of states for each time step, $S_{1:T} = \{S_1, S_2, ..., S_T\}$. Each frame can be described with a number of states $S_t = (s_t^1, s_t^2, ..., s_t^{k_t})$, where s_t^k denotes the state of the k-th object in the t-th frame and k_t represents the number of tracked object in the t-th frame.

A. Filtering and Smoothing

The tracker that we have built is largely agnostic to the choice of detector, working only on the position information of the bounding box ignoring any predicted velocity input that one might get as a result of using models such as CenterPoint [25]. The tracker uses a simple 3D Kalman filter to predict object position and motion. Similar to AB3DMOT [27], the Kalman filter state consist of an 11-dim vector $Z = [x, y, z, \theta, l, w, h, \dot{x}, \dot{y}, \dot{z}, \dot{\theta}]$, representing the position (x, y, z, θ), dimensions (l, w, h), and velocities $(\dot{x}, \dot{y}, \dot{z}, \dot{\theta})$ of the object in 3D. We included the angular rotation term in our state as we believed it would better modelled the motion of objects through time when compared to AB3DMOT which omitted it. Notably, we also modelled the objects position and motion along each axis as independent of one another. This made our Kalman filter more robust to errors in the predicted orientation. The measured state consisted of a 7-dim vector $D = [x, y, z, \theta, l, w, h]$. Following ImmortalTracker [34], we tracked all objects in the world frame and each object was kept "alive" indefinitely, meaning their position was always predicted even when not observed for some time. We do however keep track of how recently tracks have been observed, outputting only tracks which have been associated with a prediction within a predefined window of frames. We chose to follow ImmortalTrackers approach as it proposed a simple solution for handling occluded objects and false negatives. We used Euclidean distance between centre points of the predicted objects as our association metrics, with these values being gated with manually set thresholds.

After the filtering step had been completed, we performed one Kalman smoothing step to produce smoother tracks that were less effected by inaccurate predictions in position or dimensions. This operation was performed over the whole lifespan of all tracks.

B. Post Processing

Once the final set of tracks were obtained, we performed some short post processing steps in order to improve the quality of the annotations. First, we filter the tracks according to the number of predictions associated with them. Then, following [3], we classified the objects as either dynamic and static. We performed this classification only on the Car class. This decision was supported by the quantity of static Car objects in the dataset (See Table I). For each viable track we extracted both the variance of the detections center point and the begin-to-end distance of the tracked boxes in the world coordinate. We manually set a threshold for these two heuristics in order to classify the object as static. Then for each static objects, we replaced their states with the median of all valid states belonging to that track.

Ideally, a tracklet matching would then be performed using either bipartite matching [35] or a deep learning based method [6], [31], [7]. We tested two different deep learning based methods but were unable to fully implement them within the time constraints of the project. We will discuss these implementations more in Sec. VI.

VI. Experiments

In this section we will discuss the experiments we undertook and their results. Specifically, we will highlight our methodology in creating the object detection backbone for our pipeline, a comparison between PointPillars and CenterPoint, the evaluation of our MOT system, an exploration into the use of an expanded set of classes for VoD, a look into our method for tracklet matching, and concluding with a comparison and analysis of the original View-of-Delft dataset and our generated pseudo labels.

A. Model Training Setup

In our experiments, we trained two popular 3D object detection models; PointPillars [18] and CenterPoint [25]. We used an off-the shelf architecture and pre-trained backbone for both models, provided by the MMDetection3D framework [43]. The training was also done in the MMDetection3D environment, however we evaluated the model manually. We used the off-the-shelf architectures and hyperparameters provided in MMDetection3D. Both models were trained with a valid point cloud range consisting of a 50 x 50 meter box in front of the vehicle, 25 meters left and right of the vehicle and 50 meters in front of the vehicle. We also performed rotation augmentations, between $-\pi$ and π, as well as flipping augmentations across the xz plane, when training both models.

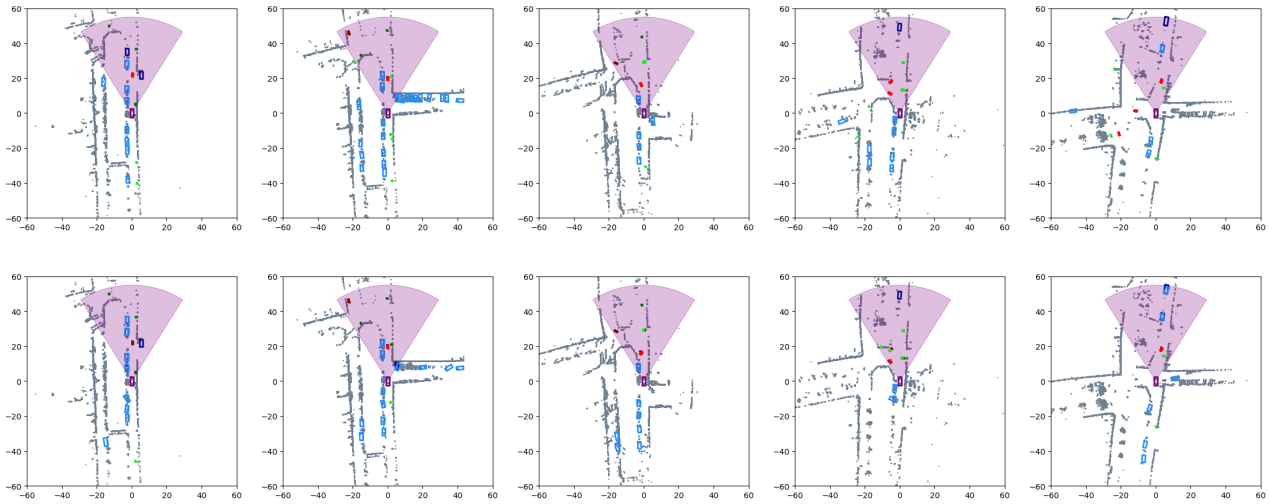

Fig. 2. Comparison between PointPillars (top row) and CenterPoints (bottom row) performance on validation sequence. Highlighted region displays the annotated portion of the point cloud. Pedestrians are displayed in green, cyclists in red and cars in blue. Lighter colours are predictions. Darker colours are labels. Grey points represent the point cloud rendered in 2D, w/ ground removal. Left to Right: Frames 400, 430, 460, 490, 520.

Finally, both models were trained in a distributed manner across four Nvidia Tesla V100 SXM2 32GB GPUs.

For PointPillars, we used a backbone pre-trained on the KITTI [41] dataset. We used a constant learning rate of 0.001 in our experiments. We found that the model's best performance was at 50 epochs. We also determined a prediction threshold of 0.5 worked best across all classes. Finally, the model had a mean latency of 132 ms per inference, with a standard deviation of 73 ms.

For CenterPoint, we used a backbone pre-trained on the nuScenes [42] dataset. We used a constant learning rate of 0.00001 in our experiments. We found that the model's best performance was at 185 epochs with the optimal prediction threshold of 0.3 across all classes. Finally, the model had a mean latency of 107 ms per inference, with a standard deviation of 88 ms.

B. PointPillars-CenterPoint Comparison

To evaluate our detector performance, we decided to use F1 score as our metric for comparison. Only the predictions that were located within the annotated portion of the point cloud were considered when calculating F1 score. An IoU threshold of 0.01 was also used when calculating F1 score.

TABLE II
F1 SCORES OF 3D OBJECT DETECTORS ON VOD DATASET FOR AN IOU THRESHOLD OF 0.01

Model	Pedestrian	Car	Cyclist	Combined
PointPillars (train)	0.468	0.460	0.386	0.464
PointPillars (val)	0.439	0.503	0.481	0.480
PointPillars (overall)	**0.462**	0.469	0.405	**0.467**
CenterPoint (train)	0.436	0.493	0.450	0.466
CenterPoint (val)	0.337	0.460	0.387	0.460
CenterPoint (overall)	0.416	**0.486**	**0.437**	0.466

The Combined column represents a class agnostic matching of bounding boxes. The overall row is an average of each classes performance on the train and validation set, weighted by the number of frames present in each set. Higher values are better. The best overall performing model for each class is highlighted in bold.

Our experiments resulted in the following performance on the VoD dataset, as seen in Table II. Our reasoning for evaluating our model over the whole dataset, rather than just the test set, is due to the purpose of this auto labelling framework being to label *this* whole dataset.

Based on the results in Table II, it is unclear as to which model is favourable. However after qualitative analysis of the results (see Fig. 2), we determined that PointPillars was a better backbone model for our framework. Bounding boxes predicted by PointPillars were much more consistent and robust to transformations of the input point cloud. In the Figure, one can see the CenterPoint network incorrectly predicts the orientation of objects or misclasssifies them entirely. It is also worse at predicting objects outside of the front view of the vehicle. These are behaviours that are not described by the evaluation method we employed in this work.

In order to correctly set the prediction threshold for each class, we visualised the F1 score profile for each model during training. An example of this can be seen in Fig. 3. We did this for all classes as well as a class-agnostic configuration, which was used to indicate overall prediction performance.

C. Tracking Performance

To evaluate tracking performance, we decided to use HOTA [44] as our evaluation metric. This choice was largely inspired by the reasoning provided in the paper. Namely that a perfect MOT evaluation metric should describe performance in not only detection, but in localisation and association also. It is our belief that the more popular MOT metric, MOTA [45], emphasises detection performance significantly more than association or localisation performance. MOTA can also incorrectly penalise good tracking performance in some situations. These claims are explained more deeply in Sec. 9 in [44].

In our experiments we evaluated both the tracker using predictions from the PointPillars network as well as from the CenterPoint network, see Table III and Fig. 4. These results

TABLE III
HOTA TRACKING PERFORMANCE ON VOD DATASET

Model	HOTA	DetA	AssA	HOTALocA(0)	HOTA(0)	LocA(0)
PointPillars (Pedestrian)	37.5	32.7	43.0	37.8	61.0	62.0
PointPillars (Cyclist)	49.2	43.5	55.6	49.4	68.8	71.7
PointPillars (Car)	49.7	43.6	56.7	48.6	66.5	73.1
CenterPoint (Pedestrian)	20.6	12.4	34.23	20.6	39.5	52.4
CenterPoint (Cyclist)	35.1	23.9	51.7	36.0	55.4	66.0
CenterPoint (Car)	38.6	29.64	50.3	36.8	53.7	68.6

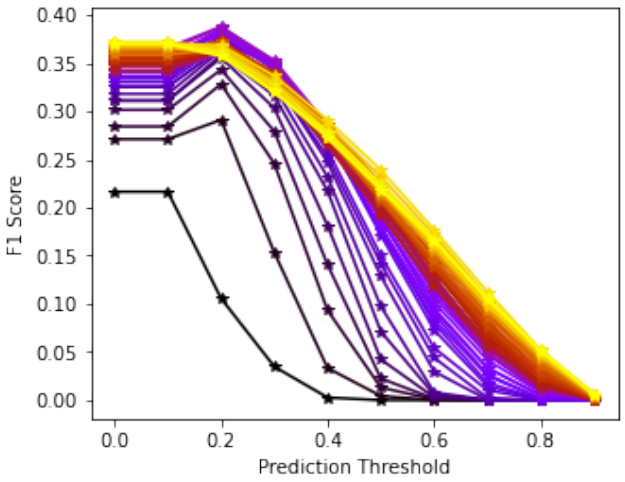

Fig. 3. CenterPoint F1 score profile on VoD validation set. The figure displays the models performance over a training cycle of 180 epochs, with each line representing a 5 epoch increment. Darker lines represent earlier epochs, while brighter lines represent later epochs. This figure shows the models performance on the Combined class, which performs a class agnostic matching of bounding boxes.

were calculated over the whole dataset. Our reasoning for this decision is the same as described in Section VI-B.

TABLE V
DISTRIBUTION VOD DYNAMIC DATASET CLASSES BETWEEN TRAINING AND VALIDATION SET

	Pedestrian	Car	Cyclist	MMS	Truck
Training	16143	15608	6685	3878	191
Validation	3749	4291	1434	395	28
Combined	19892	19899	8119	4273	219

The MMS column represents the detectors performance on the *motor_moped_scooter* class. The Combined shows a sum of the training and validation occurrences of a class.

D. Inclusion of VoD Dynamic Classes

We also performed some experiments to explore the auto labelling pipelines performance on an expanded set of labels. We again restricted the original thirteen VoD classes to a subset of five classes; this time representing pedestrians, cyclists, cars, a combination of motorcycles/mopeds/scooters, as well as trucks. The motivation behind this class selection is due to these classes being dynamic and occurring in the dataset enough to be considered relevant, see Table I.

Object detection performance on the dynamic VoD dataset is worse than on the three class VoD dataset, see Table IV. Between 5 and 110 epochs, the model learns a strategy of not predicting any trucks or motorcycles/mopeds/scooters. After this point the model begins predicting these classes, however performance on the pedestrian, car and cyclist classes reduce. We chose to compare the model's performance at 110 and 255 epochs, as CenterPoint had the best combined performance at epoch 110 and best performance on the motor/moped/scooter class at epoch 255, with best performance indicated by highest F1 score on the validation set. To further understand these results, we have included Table V which indicates the distribution of these classes between the training and validation set.

E. Tracklet Matching

We tried two different deep learning based tracklet matching networks for our auto-labelling pipeline; AFLink proposed in [7] and the motion affinity network proposed in [6]. We intended on finding an off-the-shelf matching solution that required no further training. Unfortunately, neither network provided adequate results using the pre-trained weights provided (See Fig. 5).

Regarding the performance of the motion affinity network, there are many false positive associations present and the models reasoning for matching tracks is unclear. There are also matches which seem intuitive to a human to match, but the model fails to label. The AFLink network has very low recall and the matches it does make are almost entirely incorrect.

In order to produce an auto labelling pipeline with higher quality track in lieu of an automated tracklet matching system, we decided to build a tool to handle manual track matching and pruning. This tool is described in more detail in Appendix A.

F. VoD Extension Statistics

In order to evaluate how much our pipeline augmented the original View-of-Delft dataset, we measured the total number of unique bounding boxes and tracks present in the training and validation set for the three classes that we targeted, see Table VI.

As one can see, the use of the pipeline proposed in this work does in fact increase the amount of tracks and annotated bounding boxes when compared to the original View-of-Delft dataset. There is an overall reduction of 213 tracks and 2478 bounding boxes after manual tracklet matching and pruning is performed. This represents a drop of 14.5% and 2.0% in tracks and bounding boxes, respectively. The largest per class reduction belongs to the Pedestrian class, which

TABLE IV
F1 SCORES OF CENTERPOINT (CP) ON VOD VS VOD DYNAMICS DATASET FOR AN IOU THRESHOLD OF 0.01

Model	Pedestrian	Car	Cyclist	MMS	Truck	Combined
CP *original* (train)	0.436	0.493	0.450	-.–	-.–	0.466
CP *original* (val)	0.337	0.460	0.387	-.–	-.–	0.460
CP *original* (overall)	**0.416**	**0.486**	**0.437**	-.–	-.–	**0.466**
CP *dynamic-e110* (train)	0.379	0.470	0.425	0.000	0.000	0.420
CP *dynamic-e110* (val)	0.341	0.482	0.401	0.000	0.000	0.388
CP *dynamic-e110* (overall)	0.371	0.472	0.420	0.000	0.000	0.414
CP *dynamic-e255* (train)	0.433	0.484	0.463	0.321	0.024	0.474
CP *dynamic-e255* (val)	0.371	0.288	0.378	0.054	0.000	0.371
CP *dynamic-e255* (overall)	0.421	0.445	0.446	**0.267**	**0.19**	0.453

The MMS column represents the detectors performance on the *motor_moped_scooter* class. The Combined column represents a class agnostic matching of bounding boxes. The overall row is an average of each classes performance on the train and validation set, weighted by the number of frames present in each set. The label "-eXXX" indicates a model which has been trained for XXX epochs. Higher values are better. The best overall performing model for each class is highlighted in bold.

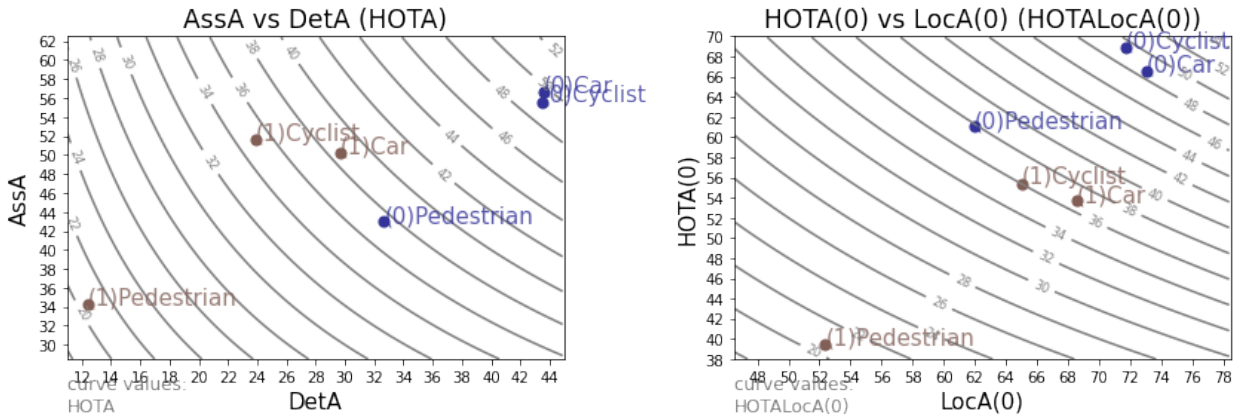

Fig. 4. HOTA (left) and HOTALocA(0) (right) performance of PointPillars (0) and CenterPoint (1) over the whole dataset. The DetA and AssA axes indicate the detection accuracy score and association accuracy score, respectievly. The LocA(0) and HOTA(0) axes indicate the localisation accuracy score and HOTA score at a IoU threshold of 0.05, respectively. Curved lines on the graph represent the HOTA and HOTALocA(0) score for each region.

saw a 22.1% and 12.3% reduction in tracks and bounding boxes. Overall when comparing the original VoD dataset to the manually matched and pruned data, the use of our proposed auto labelling pipeline increased the total amount of tracks and annotated bounding boxes by $524(71.9\%)$ and $71,306(148.9\%)$, respectively.

TABLE VI
VOD VS AUTO LABELS COMPARISON: NUMBER OF ANNOTATED BOUNDING BOXES (TOP) AND NUMBER OF UNIQUE TRACKED OBJECTS (BOTTOM) FOR THE ORIGINAL VIEW-OF-DELFT DATASET (VOD), THE RAW AUTO LABELS (RAW) AND THE MANUALLY MATCHED AND PRUNED AUTO LABELS (MANUAL). DISPLAYING DATS FOR TRAINING AND VALIDATION SEQUENCES ONLY.

	Pedestrian	Car	Cyclist	Total
VoD	19891	19898	8115	47904
Auto Labels *raw*	46031	53763	21894	121688
Auto Labels *manual*	40392	57543	21275	119210
VoD	284	309	136	729
Auto Labels *raw*	601	596	269	1466
Auto Labels *manual*	468	549	236	1253

VII. DISCUSSION

In this section, we identify the weak points of our work and highlight some areas that need further investigation.

Poor CenterPoint Performance: The results of our 3D object detection experiments were unexpected and difficult to explain. Based on its performance in [25] and adoption in other tracking frameworks [34], [6], [4], one would expect the off-the-shelf CenterPoint model to perform significantly better than off-the-shelf PointPillars model, yet this was not the case. One possible explanation for this are issues related to the pre-trained base for the CenterPoint model. As mentioned, the model was pre-trained on the nuScenes [42] dataset, which has a 32 layer point cloud. This is in comparison to VoD which has a 64 layer point cloud.

Another possible explanation is that the model is learning to reduce its loss by simply making higher confidence predictions as opposed to making impactful improvements. Evidence for this explanation can be seen in the evolution of the F1 score profiles during training (see Fig. 3). As opposed to the peak F1 score improving, the area under the curve is instead increasing and there are more predictions made with a higher score.

Tracking performance is largely influenced by detection quality [7]. We believe significant improvements could be made by simply recreating the detector performance shown in the paper it was presented in.

Transition to probabilistic model: Currently, the auto labelling framework presented in this paper treats all predictions

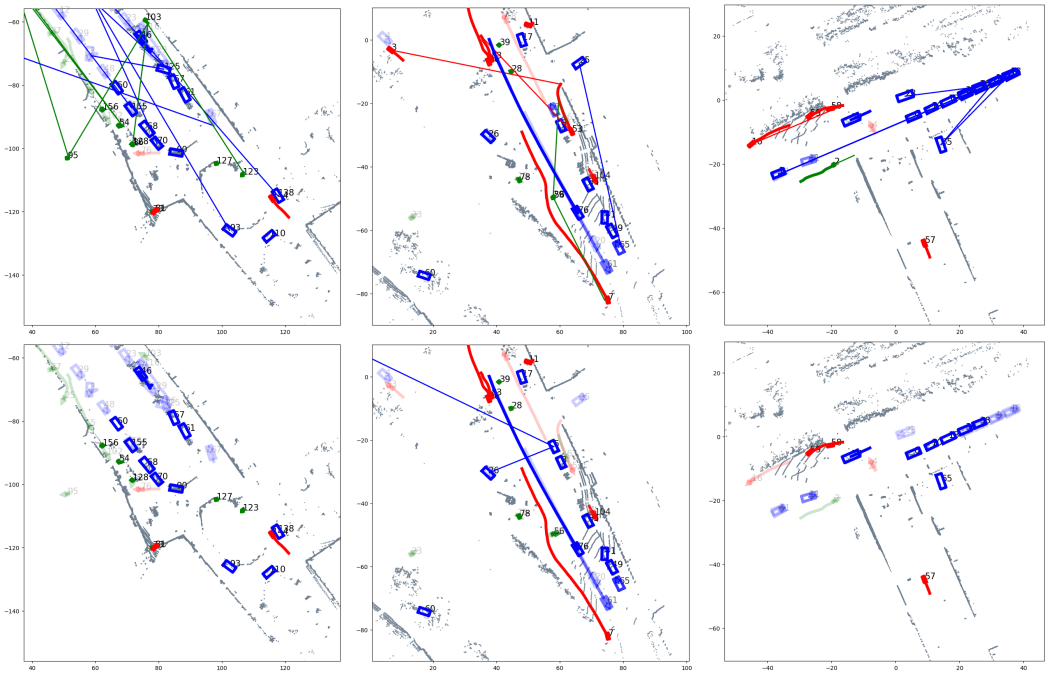

Fig. 5. Tracking with the motion affinity network [6] (top row) and with the AFLink network [7] (bottom row). Pedestrians are displayed in green, cyclists in red and cars in blue. Bright colours represent active tracks. Faded colours represent inactive tracks. Numbers represent the associated track ids. Straight lines between tracks indicate matches made by the matching network, at a confidence threshold of 0.5 and 0.9 for the motion affinity network and AFLink network, respectively. Grey points represent the point cloud rendered in 2D, w/ ground removal. Left to Right: Frames 2480, 4835, 9855.

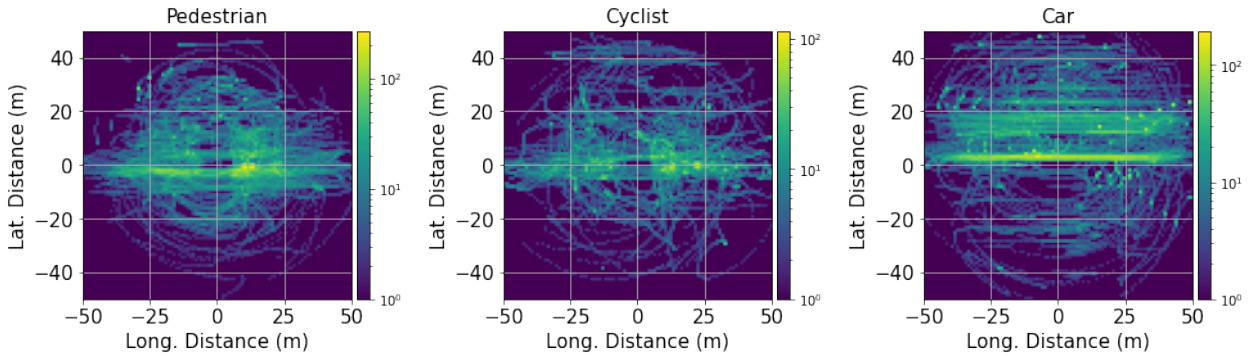

Fig. 6. Spatial distribution of tracked objects around the vehicle. The ego-vehicle is centred at (0,0). The positive x-axis is the direction the vehicle is facing. The positive y-axis is the direction left of the vehicle. Brighter regions indicate areas with a lot of occurrences. The darkest blue regions indicate areas with zero occurrences.

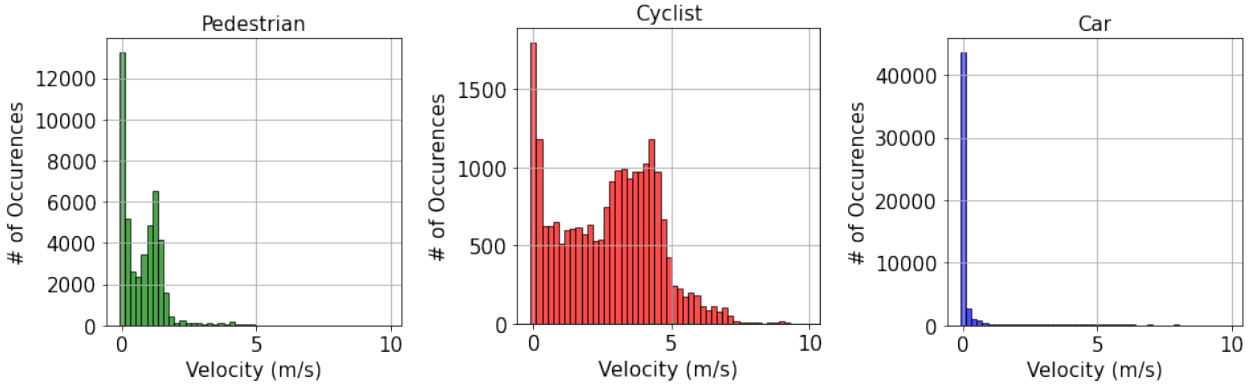

Fig. 7. Velocity profile of tracked objects around the vehicle. The height of each bar represents the number of tracked object states travelling at the corresponding velocity.

received by the tracker as true and of the same confidence, after discarding those below the detection threshold. This leads to some avoidable issues particularly related to object misclassification and how that issue interacts with the ImmortalTracker [34] framework. In typical tracking frameworks [27], false positive tracks caused by misclassification can be filtered out given their short track length. However due to ImmortalTracker's design, the lifespan of these tracks are extended indefinitely. An example of a typical misclassification in our experiment is the Cyclist class being incorrectly predicted as the Pedestrian class. An example of this can be seen in Fig. 2, CenterPoint, frame 490.

Our belief is that transitioning our tracker from plainly using the predictions to a more probabilistic model would significantly improve on this issues. To this end, we have collected statistics on the detection position and velocity profile of the tracked objects (see Figs. 6 and 7). This information could be used to generate a model which can deduce the likelihood of an object based only on it's motion, augmenting the frameworks ability to correct false positives and misclassifications. These statistics were collected on the filtered, smoothed and post processing tracks. No manual matching or pruning was performed.

Smarter initialisation: The initialisation of the object Kalman filters is another area in which employing the use of track statistics could be useful. Currently, all objects are initialised with a velocity of 0 meters per second. This causes an issue for annotating dynamic objects as the initial frames the object is observed in will have a lower velocity than is correct. Object velocity is also not included in the measurement model, rather it is inferred in the Kalman filter based on the difference in position over time. Smoothing also partially remedies this issue as the initial estimates are corrected. We believe a better strategy for initialising the Kalman filters would be to use the statistics shown in Fig. 7. A model could be initialised with two Kalman filters, representing both a static and dynamic hypothesis. We then transition to one hypothesis as object dynamics become more clear. This is not something we have investigated, however we would like to explore this in future.

Orientation issues: Expanding on this, another area in which the auto labelling performance could be improved is in its handling of the orientation of objects. The 3D object detection network is poor at predicting accurate orientation of objects. This behaviour is not evident in the evaluation metrics we used in Table II, however may be observed in Fig. 2. A common error prediction error is for the bounding boxes of objects to be inverted 180 degrees, especially for the Car class.

Currently, this issue is handled by using a set of knowledge based rules which corrects objects oriented in the opposite direction than they are travelling. This however is not a robust solution to the issue, particularly for static object, and should be addressed in further works.

Tracklet matching: Regarding the tracklet matching section of the pipeline, we used pre-trained weights for both models. The motion affinity network proposed in [6] was trained on data recorded at 2 Hz, this is compared to VoD which is recorded at 10 Hz. This was our initial guess as to why the network was failing. However the model still failed even after down sampling the input to 2 Hz. Our hypothesis as to why the model is unsuccessful is that the dynamics of objects in the dataset in which it was trained on [42] are too different from the dynamics of objects in VoD. The motion affinity network was trained only on tracking dynamic vehicles, whereas the majority of dynamic objects in VoD are pedestrians and cyclists (See Table I). Retraining the model on VoD would be the obvious next step. However due to the time constraints of the project, this was not possible.

Poor VoD dynamic performance: The proposal of the use of VoD dynamic labels is a topic we hope to pursue in future works. We believe the models performance (see Table IV) can be partially explained by the number of occurrences of each class in the training and validation sets (See Table V). The truck class for example occurs only 219 times in the whole training and validation sequence, with only 28 of those occurrences being in the validation set. This is a likely explanation as to the model initial strategy of not predicting the truck class in any frames. It also provides a possible explanation as to the model poor performance at predicting this class in the validation set.

CenterPoints poor performance at detecting the motor/moped/scooter class is more difficult to explain. This class occurs with half the frequency of the cyclist class, meaning its presence in the dataset it likely not a strong argument.

Another practical explanation worth mentioning is the possibility that the CenterPoint model was configured incorrectly. The detection head configuration for the truck and motor/moped/scooter classes were copied from the car and cyclist classes, respectively. This was done due to there being no off-the-shelf configurations for these classes. We don't believe this is a strong argument, but it is a point to consider in future.

VIII. Conclusion

In conclusion, the auto labelling framework presented in this work provides researchers with a solution for obtaining rough MOT labels in the area 360 degrees around the vehicle in the View-of-Delft dataset, and similar driving environments. We have shown that our framework can significantly increase the amount of tracked objects and unique bounding boxes in the dataset, increasing by 148.9% and 71.9% respectively. The primary contributions of this work consist of our method for extending predictions from front to surround view and our exploration into viable detection and tracking methods for the driving environment presented.

Appendix A
Manual Tracklet Matching and Pruning

In order to provide a solution for remedying the amount of false positives tracks, we decided to build a tool for manual tracklet matching and pruning. This tool works on a per sequence basis. The user can input a set of valid matching tracks (i.e. if tracks A and B are requested to be joined they occur in the sequence, begin and terminate within the frame range of the sequence and have no overlap in the times the tracks are active). The unseen tracks are created by linearly

interpolating the unseen states between the last state of the earlier occurring track and the first state of the later occurring track. All unseen states are calculated this way bar orientation, where we take the orientation of the later occurring track and set the orientation of all unseen states to that value.

To prune tracks, the user inputs a set of valid tracks and these tracks are removed from all frames in the sequence.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[2] B. Yang, M. Bai, M. Liang, W. Zeng, and R. Urtasun, "Auto4d: Learning to label 4d objects from sequential point clouds," *arXiv preprint arXiv:2101.06586*, 2021.

[3] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov, "Offboard 3d object detection from point cloud sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6134–6144.

[4] T. Ma, X. Yang, H. Zhou, X. Li, B. Shi, J. Liu, Y. Yang, Z. Liu, L. He, Y. Qiao *et al.*, "Detzero: Rethinking offboard 3d object detection with long-term sequential point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 6736–6747.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[6] X. Liu and H. Caesar, "Offline tracking with object permanence," *arXiv preprint arXiv:2310.01288*, 2023.

[7] Y. Du, Z. Zhao, Y. Song, Y. Zhao, F. Su, T. Gong, and H. Meng, "Strongsort: Make deepsort great again," *IEEE Transactions on Multimedia*, 2023.

[8] T. Likhomanenko, Q. Xu, J. Kahn, G. Synnaeve, and R. Collobert, "slimipl: Language-model-free iterative pseudo-labeling," *arXiv preprint arXiv:2010.11524*, 2020.

[9] H. Wu and S. Prasad, "Semi-supervised deep learning using pseudo labels for hyperspectral image classification," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1259–1270, 2017.

[10] B. Caine, R. Roelofs, V. Vasudevan, J. Ngiam, Y. Chai, Z. Chen, and J. Shlens, "Pseudo-labeling for scalable 3d object detection," *arXiv preprint arXiv:2103.02093*, 2021.

[11] A. Palffy, E. Pool, S. Baratam, J. F. Kooij, and D. M. Gavrila, "Multi-class road user detection with 3+ 1d radar in the view-of-delft dataset," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4961–4968, 2022.

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

[13] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu, "Voxel transformer for 3d object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 3164–3173.

[14] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel r-cnn: Towards high performance voxel-based 3d object detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 2, 2021, pp. 1201–1209.

[15] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 529–10 538.

[16] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

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

[18] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.

[19] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.

[20] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[24] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[25] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.

[26] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[27] X. Weng, J. Wang, D. Held, and K. Kitani, "Ab3dmot: A baseline for 3d multi-object tracking and new evaluation metrics," *arXiv preprint arXiv:2008.08063*, 2020.

[28] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[29] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[30] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[31] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[32] N. Benbarka, J. Schröder, and A. Zell, "Score refinement for confidence-based 3d multi-object tracking," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8083–8090.

[33] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[34] Q. Wang, Y. Chen, Z. Pang, N. Wang, and Z. Zhang, "Immortal tracker: Tracklet never dies," *arXiv preprint arXiv:2111.13672*, 2021.

[35] Y.-H. Chen, C.-Y. Wang, C.-Y. Yang, H.-S. Chang, Y.-L. Lin, Y.-Y. Chuang, and H.-Y. M. Liao, "Neighbortrack: Single object tracking by bipartite matching with neighbor tracklets and its applications to sports," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5138–5147.

[36] L. Fan, Y. Yang, Y. Mao, F. Wang, Y. Chen, N. Wang, and Z. Zhang, "Once detected, never lost: Surpassing human performance in offline lidar based 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 820–19 829.

[37] J. Lee, S. Walsh, A. Harakeh, and S. L. Waslander, "Leveraging pre-trained 3d object detection models for fast ground truth generation," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2504–2510.

[38] H. Ling, J. Gao, A. Kar, W. Chen, and S. Fidler, "Fast interactive object annotation with curve-gcn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5257–5266.

[39] D. Acuna, H. Ling, A. Kar, and S. Fidler, "Efficient interactive annotation of segmentation datasets with polygon-rnn++," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 859–868.

[40] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler, "Annotating object instances with a polygon-rnn," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5230–5238.

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

[42] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the*

IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 11 621–11 631.

[43] M. Contributors, "MMDetection3D: OpenMMLab next-generation platform for general 3D object detection," https://github.com/open-mmlab/mmdetection3d, 2020.

[44] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "Hota: A higher order metric for evaluating multi-object tracking," *International journal of computer vision*, vol. 129, pp. 548–578, 2021.

[45] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, pp. 1–10, 2008.