3D Multi-Object Tracking of *All Movable* Objects

Neehar Peri
University of Maryland
College Park, MD
peri@umiacs.umd.edu

Achal Dave
Carnegie Mellon University
Pittsburgh, PA
achald@andrew.cmu.edu

Shu Kong
Carnegie Mellon University
Pittsburgh, PA
shuk@andrew.cmu.edu

Deva Ramanan
Carnegie Mellon University
Pittsburgh, PA
deva@andrew.cmu.edu

Abstract

3D multi-object tracking (MOT) plays a crucial role in safety-critical systems such as autonomous vehicles (AVs). Modern datasets have greatly fostered 3D-MOT research, but evaluate tracking performance only on a few common classes, such as `pedestrians` *and* `vehicles`*. As any movable object can potentially affect motion planning, AV perception should reliably track* **all** *movable objects such as* `animals` *and* `strollers`*. Motivated by this, we explore an extended 3D-MOT which requires tracking not only objects from common classes, but also all* `other` *movable objects. In practice, these objects are from some "other" rare classes, which have too few examples for reliable benchmarking. As a result, despite having annotated all these "other" objects, modern datasets ignore them in their final benchmark. We repurpose these historically ignored objects by aggregating them into a single catch-all* `other` *class that can be reliably benchmarked. Furthermore, inspired by open-set recognition, we study how 3D-MOT methods can track movable objects from novel classes unseen in training. Extensive experiments show consistent and notable improvements in tracking all movable objects by grouping classes during training and applying non-maximum suppression on the predicted tracks.*

1. Introduction

3D multi-object tracking (3D-MOT) plays a crucial role in safety-critical systems such as autonomous vehicles (AVs) [13, 35, 4]. The emergence of modern datasets, especially those based on LiDAR data, have greatly advanced 3D-MOT research [13, 4, 5, 31]. However, these datasets only benchmark a few common classes such as `pedestrians` and `cars`. In the open-world, AV perception must reliably track *all movable* objects[1] in the real open world, because they can critically affect navigation behavior [32, 37].

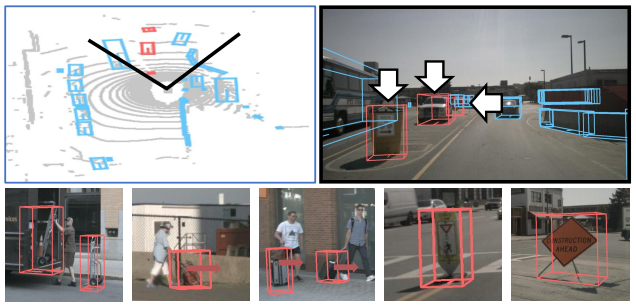

Figure 1: 3D multi-object tracking (3D-MOT) over LiDAR point clouds requires tracking objects from K common classes, *e.g.*, `bus` and `car` colored blue. However, safety-critical systems such as autonomous vehicles must reliably track **A**ll **M**ovable **O**bjects that can critically affect navigation behavior. Motivated by this, we study 3D-MOT-AMO, which requires tracking not only objects from K common classes, but also `other` movable objects, which are primarily from many rare classes, *e.g.*, `dolly` and `stroller` as colored red in the RGB images. Intuitively, the complexity of detecting and tracking `other` movable objects, due to the sparsity of training data and diverse 3D shapes, poses new challenges.

Recognizing this, nuScenes [4], a popular large-scale AV dataset, has already annotated *all movable* objects (Fig. 1). However, since many movable objects come from rare classes (cf. long-tail distribution [22, 14, 4]) which have too few examples for reliable training and benchmarking, nuScenes has only focused on K common classes and ignored (until now!) the remaining annotated objects in the

[1] In autonomous vehicles, *movable* objects are objects relevant to driving safety that are moving, or could move in the future (the next day/hour/minute/second) [7, 4]. Movable objects include those that can move with self-propelled locomotion (*e.g.*, `cars` and `animals`) and those that can be moved by external forces (*e.g.*, `strollers` and `wheelchair`).

final benchmark. In this work, we explore 3D-MOT of *all movable* objects in a meaningful way and make three contributions: problem definition, evaluation protocol and technical insights.

Problem Definition. Motivated by improving safety in AVs, we explore 3D-MOT of *All Movable Objects* on LiDAR input (Fig. 1), so termed as **3D-MOT-AMO**, which requires tracking not only objects from K common classes (*e.g.*, `pedestrians` and `vehicles`), but also `other` movable objects (*e.g.*, `animals`, `wheelchairs` and `strollers`). These `other` movable objects are primarily found in rare classes, which have too few examples to be used for training or benchmarking in a per-class fashion. Fortunately, in the real world, recognizing the class labels of `other` objects may not be critical: *e.g.*, discriminating a `stroller` from a `wheelchair` would not change the motion plan of "slow-down" or "yield" for collision avoidance in AVs. Therefore, we propose aggregating the `other` movable objects that are outside the K common classes into a single catch-all `other` class. Because the *collective* size of the aggregate is sufficiently large, we can now train on and reliably evaluate these movable objects.

Evaluation Protocols. We introduce two setups to explore 3D-MOT-AMO. First, we simply extend the typical K-way 3D-MOT evaluation protocol by including all `other` movable objects as a $(K{+}1)^{th}$ `other` class. We call this the *Default* setup. It is important to note that the `other` class is considerably more complex than the K common classes due to its diversity in size, and the sparsity of data examples. Second, `other` objects encountered at test time might come from novel rare classes unseen in training. Inspired by open-set literature [28, 10], we introduce a special case of 3D-MOT-AMO for the *Open-Set* setup which explicitly evaluates how well a 3D-MOT model tracks `other` movable objects sampled from novel classes unseen during training (Section 3). Through our baseline experiments, we show that the complexity of the `other` class makes this problem a non-trivial extension of 3D-MOT.

Technical insights. Under the above setups, we explore various approaches to 3D-MOT-AMO (Section 4). First and foremost, extensive experiments reveal that tracking `other` movable objects is significantly more difficult than tracking those from the K common classes, because of the complexity of the $(K{+}1)^{th}$ `other` class, which contains diverse 3D objects of various sizes and from many rare classes. This suggests that 3D-MOT-AMO is a non-trivial extension of 3D-MOT, requiring explicit future attention to solve it. Moreover, we find it useful to carefully group the training examples for simultaneously training for tracking both the K common classes and the $(K{+}1)$ `other` class, owing to a regularization from data itself. Further, we find that between-class non-maximum suppression, which is largely neglected in tracking, further improves overall tracking performance, owing to the removal of overlapping tracks.

2. Related Work

3D-MOT aims to track objects from K (common) classes of interest [35, 6]. Recently, it has seen significant advances, driven by deep neural networks trained on large-scale benchmark datasets [13, 5, 4]. In the context of AVs, 3D-MOT takes LiDAR point clouds as input, which accurately measures the 3D world [4]. In the real open-world, AV perception must track all objects that can potentially affect navigation behavior [32, 37]. However, current benchmarks only focus on a few common classes such as `pedestrians` and `cars`. As a result, 3D-MOT models trained with such labels cannot properly handle `other` objects outside the of K classes [18, 34, 39]. Motivated by the safe operation of AVs, we formulate the problem of *3D-MOT-AMO*, which requires tracking objects from both the typical K common classes and the $(K{+}1)^{th}$ `other` class.

3D-MOT methods commonly follow a tracking-by-detection paradigm [2, 35, 39] that first detects objects from the K classes and links the detections across time to form tracks [36, 35]. For 3D detection, many methods adopt an anchor-based model architecture that defines per-class shape anchors to guide class-aware object detection [20, 43, 18]. These methods show that class anchors robustly measure per-class 3D shapes that help LiDAR-based 3D detection [38]. To link detections into tracks, [3, 35] adopt established data association methods such as the Kalman Filter and Hungarian algorithm. Notably, CenterPoint [39] learns an *anchor-free* 3D-MOT model without per-class anchors for 3D object detection. Specifically, it learns to predict an object's center and estimates the 3D shape for each detected object's center. Moreover, CenterPoint learns its own tracker simultaneously by predicting each object's velocity offset between frames. CenterPoint achieves near real-time inference speed and state-of-the-art 3D-MOT performance.

Open-set recognition requires recognition of diverse open-set testing examples that are outside K classes of interest during K-way classification in testing [28, 22]. Typically, classifiers are trained with some outlier data as open-set training examples [15, 10]. During evaluation, the classifier is exposed to never-before-seen examples that are from novel classes, and is required to recognize them without decreasing the classification accuracy on the K classes. Inspired by open-set recognition, we explore 3D-MOT-AMO with an additional requirement of detecting and tracking novel movable objects outside the K classes.

3. 3D-MOT-AMO Protocol

Conceptually, 3D-MOT-AMO extends the traditional 3D-MOT problem, which focuses on tracking objects from

the K common classes, by further requiring tracking all `other` movable objects as the $(K{+}1)^{th}$ class. Since the `other` class encapsulates diverse 3D objects from many rare classes, learning to track them is more challenging than learning to track objects from any K common classes. Importantly, during testing, there might be unknown movable objects from novel (rare) classes outside all the classes seen during training. In this sense, 3D-MOT-AMO is a non-trivial extension of 3D-MOT.

3.1. Evaluation Setups

Recall that the `other` class contains diverse 3D objects of various sizes and from many rare classes. This complexity requires careful setups to study 3D-MOT-AMO. We introduce two setups below.

Default Setup. As introduced earlier, we aggregate the historically-ignored classes as a single $(K{+}1)^{th}$ `other` class. Therefore, by default, we study a $(K{+}1)$-way 3D-MOT problem. While this setup seems straightforward, our experiments show that the complexity of the $(K{+}1)^{th}$ `other` class makes 3D-MOT-AMO a non-trivial extension of 3D-MOT. It is worth noting that this setup assumes that the training and testing data distributions are the same, *i.e.*, all rare classes are seen in both training and testing. For brevity, we report the average performance across all K common classes for both detection and tracking metrics. We compute the overall detection and tracking accuracies as the average of all $(K{+}1)$ classes.

Open-Set Setup. The above setup does not consider the real-world case that `other` movable objects at test time might be from novel classes unseen during training. This is equivalent to an open-set recognition problem that requires recognizing data not belonging to any of the known classes used for training [28, 10]. To study this special case, following the practice of open-set recognition [28], we hold out a few rare classes (and their LiDAR sweeps) during training and only include them for evaluation in testing. We further analyze detection and tracking performance on subgroups of testing examples including all the `other` examples (`other`-all), those from the `other` classes used in training (`other`-known) and those from novel classes *only* seen in testing (`other`-unknown) in the supplemental material.

3.2. Evaluation Metrics

For 3D-MOT-AMO, methods must first detect all movable objects. Therefore, we evaluate both detection and tracking performance using the established metrics briefly introduced below.

Detection Metric. Mean average precision (mAP) is an established metric for object detection [12, 13, 21]. For 3D detection on LiDAR sweeps, a true positive (TP) is defined as a detection that has a center distance within a distance threshold on the ground-plane to a ground-truth annotation [4]. mAP computes the mean of AP over classes, where per-class AP is the area under the precision-recall curve for recall and precision over 10%, and distance thresholds of [0.5, 1, 2, 4] meters.

Tracking Metric. Identity F1 (IDF_1) is a well-known tracking metric introduced in [27]. Another equally well-known and commonly used metric is MOTA [1]. However, MOTA overemphasizes detection over association [29, 8, 23]. Since we also report detection performance using mAP as introduced above, in this paper, we choose to evaluate tracking performance with IDF_1, which focuses on measuring association accuracy over detection accuracy [24, 25, 33, 23]. IDF_1 is calculated as follows.

Given predicted tracks and ground-truth tracks, IDF_1 applies a *global min-cost matching* that assigns predicted tracks to ground-truth tracks so as to minimize the number of false positive and false negative frames induced by the matching. Within every predicted track, at each frame, a detection that sufficiently overlaps a bounding box from the matched ground-truth track is counted as an IDTP (ID true positive); otherwise it is an IDFP (ID false positive). All unmatched ground-truth boxes are IDFNs (ID false negatives). Finally, IDF_1 is computed below:

$$\text{IDF}_1 = \frac{2\ \text{IDTP}}{2\ \text{IDTP} + \text{IDFP} + \text{IDFN}}$$

4. 3D-MOT-AMO Methodology

Recall that 3D-MOT-AMO requires tracking objects not only from the K common classes but also many `other` classes, *i.e.*, the $(K{+}1)^{th}$ `other` class. In particular, unlike any of the K common classes, the `other` class contains many diverse movable objects that have various 3D shapes and are drawn from many rare categories. Therefore, approaches to 3D-MOT-AMO should consider the complexity of the `other` class. To better understand this new problem, rather than rushing to sophisticated solutions, we study several meta-approaches, *i.e.*, using traditional and simple classification schemes such as post-hoc modeling [15, 41] and $(K{+}1)$-way classification [12], as depicted in Fig. 2.

$(K+0)$-way model. Naively, we can simply post-process predictions from a K-way 3D-MOT model (trained only on the K common classes) to obtain "pseudo" tracks for `other`. To generate predictions for the `other` objects, we can threshold the predictions of the K classes w.r.t their confidence scores [15]. Intuitively, a `bicycle` detector might detect `strollers` with low-confidence [9]. For each of the K common classes, We perform a parameter sweep on a val-set to find the maximum confidence score that does not decrease the average mAP for all classes. We further test a learning-based method that trains atop the $(K{+}0)$-way model using the generated predictions and ground-truth annotations [41], with the goal of identifying the predictions

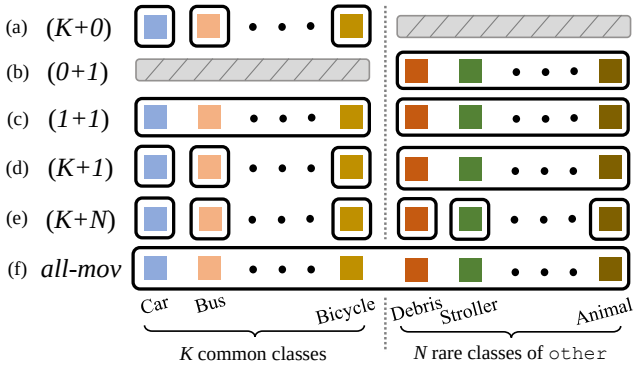

Figure 2: **Different meta-approaches** to 3D-MOT-AMO. **(a)** Given a 3D-MOT model trained on K common classes, we explore methods to post-process its predictions to obtain "pseudo" tracks for `other` objects, *e.g.*, by simply thresholding the results w.r.t the confidence scores. **(b)** Another post-hoc method is to train a (0+1)-way model that just tracks `other` movable objects trained with the aggregate of all N `other` rare classes. Conceptually, jointly using this (0+1)-way model and the (K+0)-way model in conjunction is a viable approach to 3D-MOT-AMO. **(c)** Instead of using two separate models for 3D-MOT-AMO, we can directly train a (K+1)-way model using all the training classes: the K common classes and the $(K{+}1)^{th}$ `other` class. **(e)** Since we have all the $K{+}N$ labels for all movable objects, we explore training a (K+N)-way model and merge the results corresponding to the N classes to track the `other` class. **(f)** We aggregate all the training examples to train a single "movable" model that tracks all the movable objects (so called *all-mov*). We combine *all-mov* and the (K+0)-way model as a viable approach.

of `other` movable objects and refining the predictions of the K common classes.

$(0+1)$**-way model**. Another simple post-hoc approach is to train an independent model just for the `other` movable objects. Then, we apply both the (0+1)-way model and the (K+0)-way model for 3D-MOT-AMO, as illustrated by Fig. 3. Note that the outputs from the two models might have overlapping tracks. We use non-maximum suppression (NMS) to remove lower-scoring tracks, as described later in this section.

$(1+1)$**-way model**. We further exploit the K common classes to train a $(1+1)$-way model for tracking `other` movable objects. We hypothesize that this model performs better than the (0+1)-way model because the data from different classes can regularize model training. Intuitively, the (0+1)-way model may overly generalize to movable objects belonging to the K common classes due to inductive bias. For 3D-MOT-AMO, we use this model along with the (K+0)-way model to obtain tracks of all movable objects.

$(K+1)$**-way model**. The previous methods adopt two independent models for 3D-MOT-AMO. We now directly train a $(K+1)$-way model using K common classes and the $(K{+}1)^{th}$ `other` class that aggregates the remaining annotated examples. We might expect training on `other` objects, in addition to the typical K common classes, to decrease overall tracking performance. However, our experiments show jointly training improves performance, verifying the benefit from data regularization by using all the classes.

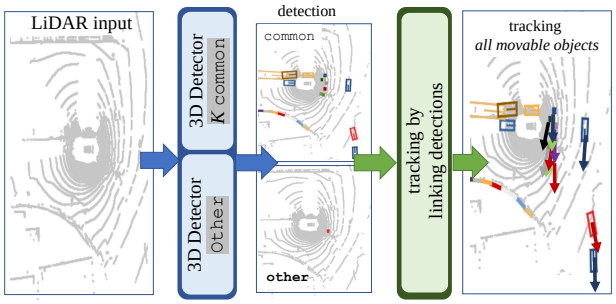

Figure 3: We explore several post-hoc methods that combine two independent models for 3D-MOT-AMO: (1) the typical (K+0)-way model is particularly trained for tracking objects from the K common classes, and (2) a post-hoc model for tracking `other` movable objects. In merging results from the two models, we apply NMS on tracks to remove overlapping tracks, leading to better tracking performance.

$(K+N)$**-way model**. As the training set has already annotated all movable objects with $K{+}N$ class labels, we explore a straightforward method that trains a $(K+N)$-way model by treating all classes independently. In testing, we merge the predictions from the N rare classes as the tracks of the `other` movable objects. Since objects from these N rare classes are few in number and are sparsely sampled, we carefully perform class-balanced sampling to stablize training [43].

All-Movable **model**. Lastly, we consider a model that trains on *All-Movable* (*all-mov* for short) annotated objects (from all the $K{+}N$ classes). This model is directly trained to track all movable objects and does not predict class labels. To evaluate under the 3D-MOT-AMO protocol, we jointly use this model and the (K+0)-way model to obtain predicted labels of the tracks, which requires non-maximal suppression of tracks as detailed below.

Non-Maximal Suppression on Tracks. Some of the previously introduced methods use two independent models for 3D-MOT-AMO. When merging the outputs from two independent models, there are likely to be overlapping tracks. To suppress such overlapping tracks, we use non-maximal suppression (NMS), which is a standard technique in detection [11, 44, 17, 20, 39] but largely undermphasized in tracking. We call this approach track-NMS to distinguish it from the typical NMS on detections, which is already adopted in the detection step (Fig. 3). We find that applying track-NMS between tracks of the K common classes and tracks of `other` objects leads to notabe improvement, while applying it to remove tracks among K common classes does not help. We conjecture that the K common classes are sufficiently distinct in the 3D LiDAR space that a trained

model does not generate many overlapping tracks among them (so there is no need to suppress); but the `other` class contains diverse 3D objects which can be easily misclassified as one of the K common classes and hence benefits from track suppression. To properly apply track-NMS, we tune the threshold to filter out overlapping tracks on the val-set with the goal of improving tracking performance.

5. Experiments

We carry out extensive experiments to better understand the 3D-MOT-AMO problem, and gain insights by validating the meta-approaches as elaborated in Section 4. Specifically, we aim to answer the following questions:[2]

1. Is the $(K{+}1)^{th}$ `other` class more difficult to detect and track than the K common classes?
2. Does a typical K-way 3D-MOT model (trained on only K common classes) frequently track `other` movable objects as false positives (FPs)?
3. If yes to the above, can we use a simple method (*e.g.*, thresholding low-scoring predictions or learning a classifier to rescore) to post-process the predictions of the K-way model to obtain meaningful tracks on the `other`?
4. Do different grouping methods presented in Section 4 significantly impact performance w.r.t tracking `other` movable objects?
5. Does jointly learning to track both K common classes and the `other` class harm the tracking performance compared to training individual models?

We start this section by introducing the model architecture, implemetnation and dataset.

Model Architecture. Section 4 describes general methods that are agnostic to model architectures. While many architectures can be used for the exploration of 3D-MOT-AMO, in this paper, we use CenterPoint [39], which is a recently published 3D-MOT model. It learns to detect 3D objects by treating them as points, and simultanenously learns to regress the 3D shapes and velocities for all the detected points. For tracking, it simply uses the predicted velocities to greedily link detections into tracks. Despite its simplicity, CenterPoint achieves near-realtime inference speed and the state-of-the-art on various 3D-MOT benchmarks [4, 31]. We have also tested other model architectures and find that our conclusions hold regardless the underlying architecture (details in the supplemental).

Implementation. We use the PyTorch toolbox [26] to train all models for 20 epoch with the Adam optimizer [19] and a one-cycle learning rate scheduler [30]. In training, we adop data augmentation techniques introduced by [39]. As a sanity check, we re-train the models (e.g., CenterPoint) and have verified that our re-trained models match the performance as reproted in the literature. We will release our code to foster the research of 3D-MOT-AMO.

5.1. Dataset

A number of LiDAR-based 3D-MOT datasets have been released in recent years for AV research such as nuScenes [4], Argoverse [5], KITTI [13], and Waymo [31]. We choose to work with nuScenes because *it is the only public large-scale 3D-MOT dataset that has annotated all movable objects*, allowing for the exploration of 3D-MOT-AMO.[3] The nuScenes dataset annotates 23 classes of movable objects, some of which are merged into superclasses, resulting into 16 classes in its final benchmark (Fig. 4). Other datasets tend to annotate only a few classes, making them difficult to repurpose for 3D-MOT-AMO exploration.

3D-MOT-AMO using nuScenes. In the nuScenes benchmark, only a subset of its annotated classes are evaluated because they have a large number of examples (Fig. 4). Particularly, nuScenes ignores many rare classes (*e.g.*, `wheelchair` and `stroller`, and ambiguously defined superclasses (*e.g.*, `debris` and `pushable-pullable`) which encapsulate more rare classes. We evaluate these historically ignored classes as the $(K{+}1)^{th}$ `other` class. In sum, to study 3D-MOT-AMO, that requires tracking all movable objects, we evaluate all annotated classes, which covers all movable objects. Please refer to the supplemental for our inspection and verification that nuScenes does annotate all movable objects. For the Open-Set tracking setup, we hold out `stroller`, `animal`, `police`, `ambulance`, `wheelchair` and `personal-mobility` as open-set, unseen during training but used for evaluation in testing. Concetely, we remove the LiDAR sweeps which contain any objects from these held-out classes for training.

Subsets for training, validation, and testing. We use the nuScenes official train-set to train all the models. We further split a subset of 100 scenes from this train-set as a val-set for model tuning. We test all models on the nuScenes official validation set.

5.2. Post-Processing K-way 3D-MOT Model

A 3D-MOT model trained on only the K common classes will surely produce false positives (FPs). But do these FPs contain legitimate tracks for `other` movable objects? We start our experiments by answering this question.

We first explore this by evaluating an *oracle* that optimally matches all predicted bounding boxes from the $(K{+}0)$ model to ground truth bounding boxes and discards unmatched predictions. We link these detections using the

[2] Answers: yes, yes, no, yes, no.

[3] While the nuScene dataset [4] does not explicitly state that it has annotated all movable objects, its annotation guide implies the requirement of labeling all movable objects. We further verify this through our manual inspection. Please refer to the supplemental for details.

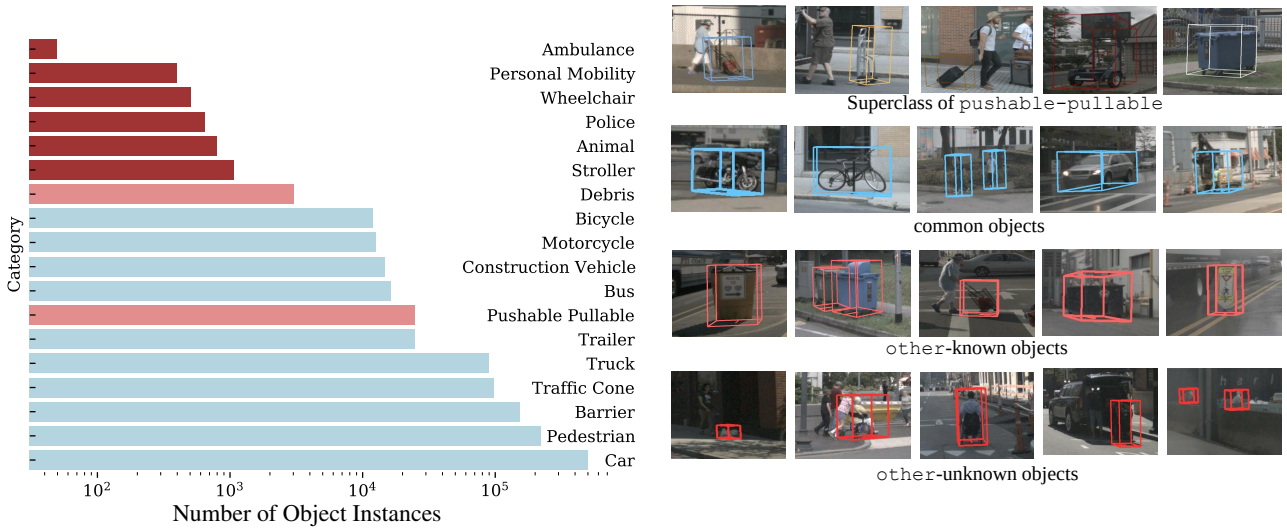

Figure 4: **Left:** Histogram of per-class object counts in nuScenes. The classes highlighted in red are ignored in the standard benchmark. The ignored classes either have very few examples (*e.g.*, `wheelchair`) or are superclasses which encapsulate many other fine-grained categories, *e.g.*, `pushable-pullable` that contains `shopping-cart`, `dolly`, `luggage` and `trash-can` as shown in the top row on the right pane. To explore 3D-MOT-AMO, we repurpose these historically ignored classes as the $(K{+}1)^{th}$ `other` class of other movable objects outside the K common classes (in blue). In particular, for the *Open-Set* setup, we use the two superclasses `debris` and `pushable-pullable` in light red as the `other` training examples (termed as `other`-known) [16, 10], and hold out the rest in dark red as novel classes only seen in testing (termed as `other`-unknown). **Right:** We show some selected objects from the `pushable-pullable` superclass (1st row), K common classes (2nd row), the `other`-known used in training (3rd-row), and `other`-unknown only seen in testing under the *Open-Set* setup (4th-row).

Table 1: We study several methods for post-processing predictions from a K-way model trained only on the K common classes. An *oracle* optimally matches the predictions from the K-way model to the ground truth and discards unmatched predictions. Unsurprisingly, the *oracle* produces significantly better performance on detection and tracking for both the K common classes and the `other` class. This implies that our K-way model already detects, but misclassifies `other` objects. Post-processing the existing set of detection results could be a viable baseline. Therefore, we test two simple methods: 1) thresholding low-scoring per-class predictions as `other` objects [15], and 2) learning an MLP based classifer atop the K-way model using the ground-truth labels [41]. However, neither works well in practice. This suggests fundamental changes are required in model training to solve 3D-MOT-AMO.

Method	`Common`		`Other`		`All`	
	mAP	IDF_1	mAP	IDF_1	mAP	IDF_1
(K+0)	59.1	71.1	0	0	53.7	64.7
oracle	70.9	78.6	25.6	41.8	66.8	75.2
thresh.	59.1	71.3	0	1.0	53.7	64.9
classifier	57.2	68.3	0	0	52.0	62.0

learned tracker from CenterPoint. We find that the *oracle* significantly improves performance on the K common classes by removing FPs (not surprising), and surprisingly produces strong results for `other`, as shown in Table 1. Clearly, many predictions from the (K+0) model are simply misclassified `other` objects. This motivates us to test two simple baselines to improve upon the results of the K-way 3D-MOT model: (1) thresholding per-class predictions w.r.t the confidence scores and using the low-scoring results as the `other` objects [15], and (2) learning a classifier (multi-layer perceptron, MLP) atop the K-way model to predict the class label given the predicted attributes about a particular bounding box [41]. However, neither works. Even worse, learning an MLP classifier notably degrades performance, presumably because we train it on the same training data that serverely overfits to the train-set and generalizes poorly. This suggests that misclassifications between K common classes and the `other` class cannot be easily resolved via simply post-processing the predictions of the K-way model; instead we should approach 3D-MOT-AMO "from scratch", *e.g.*, training with the `other` examples.

5.3. Default Setup for 3D-MOT-AMO

We now study 3D-MOT-AMO under the *Default* setup, which allows for training on all movable objects including the `other` classes. We assume that the models have seen all testing classes at train time. Table 2 lists the detailed results for both 3D detection and tracking.

Training with the historically-ignored `other` classes

Figure 5: **Qualitative results**. We visualize tracking results on some testing examples that contain `other` objects trained under the *Default* setup. As LiDAR point clouds provide a 360-degree input, our (K+1)-way model approach detects and tracks `other` objects in all views, as highlighted by the big white arrows. For better visual inspection, we attach the corresponding RGB images for tracked `other` objects next to the LiDAR visuals. Clearly, the `other` objects detected and tracked by our model, such as the `strollers` and `carts` (left-column), are critical to safe AV operation. We note that such objects are movable and can move when unattended on their own (*e.g.*, a runaway stroller). This demonstrates the importance of studying 3D-MOT-AMO.

boosts tracking performance for the `other`, cf. (K+1) vs. (K+0). However, compared to the performance for the K common classes, the metrics for the `other` class is much lower, cf. mAP and IDF_1 on `other` vs. `common`. This suggests that the `other` class is more challenging than the K common classes. However, for both detecting and tracking `other` objects, simultaneously learning with the K common classes boosts performance, cf. (1+1) vs. (0+1), and (K+1) vs. (0+1) on IDF_1. We conjecture the K common classes improves `other` detector and tracker training by defining what are not `others`. Lastly, careful grouping the classes in training notably improves detection and tracking performance for both K common classes and the `other` class, cf. (K+1) vs. (K+N). Fig. 5 shows visual results for 3D-MOT-AMO under this setup.

5.4. Open-Set Setup for 3D-MOT-AMO

For 3D-MOT-AMO under the *Open-Set* setup, we compare 3D detection and tracking in Table 3. We summarize the salient conclusions below.

As expected, the (K+0)-way model achieves "zero" performance in detecting and tracking `others`, because it is not designed for 3D-MOT-AMO. When explicitly trained on `other` objects provided by the nuScenes dataset, all models can detect and track `other` movable objects. Moreover, grouping the N `other` classes into a single group modestly improves tracking `other` objects, cf. from (K+N) to (K+1) on IDF_1 in Table 3, which is further significantly improved by grouping the K common classes together, cf. from (K+1) to (1+1). This demonstrates that carefully grouping `other` objects allows for better tracking of diverse objects and reasoning about the `other` movable objects compared to the K common classes.

Comparing between the Default setup (Table 2) and the Open-Set setup(Table 3), we can see the tracking performance on the `other` class is much lower under the *Open-Set* than that under the *Default* setup, e.g., 9.3 vs. 36.1 by (K+1) on IDF_1-`other`. Clearly, the novel movable objects

Table 2: 3D-MOT-AMO under the *Default* setup. We compare all methods for 3D detection and tracking of all movable objects. All models, except for the (K+0)-model, are trained with both the K common classes and the $(K+1)^{th}$ `other` class. Unsurprisingly, the (K+0) model achieves "zero" performance on `other` objects, while training with `other` examples boosts performance for the `other` class. We combine the (0+1), (1+1) and *all-mov* with the (K+0)-way model in order to fully address 3D-MOT-AMO. We find it helpful to incorporate the K common classes to train a model for tracking `other` objects, cf. (1+1) improves over the (0+1) on IDF$_1$, presumably because to the K common classes clearly define what are not `other` objects, helping train a better model for tracking `others`. Lastly, careful grouping of classes greatly improve tracking performance, cf. (K+1) vs. (K+N), and (K+1) vs. (1+1).

Method	Common		Other		All	
	mAP	IDF$_1$	mAP	IDF$_1$	mAP	IDF$_1$
(K+0)	59.1	71.1	0	0	53.7	64.7
(K+N)	56.9	70.2	15.5	28.2	53.2	66.4
(K+1)	58.7	71.4	20.4	**34.9**	55.2	**68.0**
(1+1)	59.0	70.7	**20.7**	9.7	**55.5**	65.1
(0+1)	58.8	70.6	16.3	5.6	55.0	64.7
all-mov	56.9	70.0	0	2.7	51.7	63.8

Table 3: 3D-MOT-AMO under the *Open-Set* setup. The same conclusion holds here as drawn from Table 2: (1) training with `other` examples boosts performance on the `other` class, cf. (K+1) vs (K+0); (2) training with both K common classes and the `other` class helps tracking `others`, cf. (1+1) vs. (0+1); (3) careful grouping greatly improves detection and tracking performance, cf. (1+1) vs. *all-mov* and (1+1) vs. (K+N), implying an effective grouping strategy should neither be too coarse nor too fine-grained, but just right. Interestingly, comparing the results of this table and Table 2, we find the models acheive similar performance on the K common classes across the two setups, but quite low tracking performance on `other` in this *Open-Set* setup. We believe this is due to many novel objects seen in the testing but unseen in training.

Method	Common		Other		All	
	mAP	IDF$_1$	mAP	IDF$_1$	mAP	IDF$_1$
(K+0)	59.1	71.1	0	0	53.7	64.7
(K+N)	57.7	70.4	19.2	6.5	54.2	64.6
(K+1)	57.8	70.1	19.0	7.5	54.2	64.4
(1+1)	59.0	70.6	**20.7**	**10.2**	**55.5**	**65.1**
(0+1)	58.9	70.7	17.6	4.7	55.1	64.7
all-mov	56.9	70.1	0	2.5	51.7	63.9

Table 4: We study the impact of track-NMS with the best models under the two setups, *i.e.*, (K+1) from the *Default* setup and (1+1) from the *Open-Set* setup, w.r.t tracking performance (IDF$_1$). For each model,we run track-NMS to remove overlapping tracks from the `other` class; this guarantees the same performance on the K common classes. Clearly, track-NMS improves tracking performance on the `other` (cf. numbers in bold), implying the removed `other` tracks are likely belong to K common classes.

Method	Default			Open-Set		
	Common	Other	All	Common	Other	All
(K+1)	71.4	34.9	68.0	70.1	7.5	64.4
+ *Track-NMS*	71.4	**36.1**	68.1	70.1	**9.2**	64.5
(1+1)	70.7	9.7	65.1	70.6	10.2	65.1
+ *Track-NMS*	70.7	**11.5**	65.3	70.6	**11.3**	65.2

pose further difficulties in tracking the `others` as a whole, further motivating the need to study this special case.

Another subtle, yet interesting observation is that detection performance on `others` is similar under both setups, but tracking performance is not. Presumably this is because the models under the Open-Set setup have not learned to generalize to the motion of novel objects. This suggests that data association can be further improved for better tracking of `others`, *e.g.*, by exploiting RGB for re-identification of the detections to form better tracks [42, 40].

5.5. Study of NMS on Tracks

We study track-level NMS to remove overlapping tracks corresponding to the same object (Table 4). For this study, we turn to the best models from the two setups, *i.e.*, (K+1) from the Default setup and (1+1) from the Open-Setsetup. We find that applying track-NMS to remove overlapping tracks consistently helps when we remove the tracks from the `other` class while keeping the K common classes intact. This gaurantees the same tracking performance on the K common classes. Clearly, track-NMS modestly improves tracking performance on `other` objects, suggesting that overlapping tracks of the `other` class are likely from the K common classes.

6. Conclusion and Future Work

We propose 3D-MOT-AMO, which requires tracking 3D objects not only from K common classes, but also from many other *movable* classes that are oftentimes rare occurences in the real world. This new problem is motivated by the operation safety of autonomous stacks in the real open world. Autonomous vehicles must reliably track *all movable objects* for appropriate motion planning and collision avoidance. To study 3D-MOT-AMO, we establish rigorous evaluation protocols and study various simple approaches to understand the problem. Our extensive

study shows that this new problem is very challenging and worth further exploration in the community. We envision that better solutions should make fundamental changes over our explored meta-approaches. We expect that successful approaches will leverage insights from other areas including few-shot learning, long-tail learning, synthetic data adaptation, RGB & LiDAR multi-modal learning, and re-identification for tracking.

References

[1] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008. 3

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016. 2

[3] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Tozeto Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing, ICIP*, pages 3464–3468. IEEE, 2016. 2

[4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020. 1, 2, 3, 5

[5] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 1, 2, 5

[6] Hsu-kuang Chiu, Antonio Prioletti, Jie Li, and Jeannette Bohg. Probabilistic 3d multi-object tracking for autonomous driving. *CoRR*, abs/2001.05673, 2020. 2

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 1

[8] Achal Dave, Tarasha Khurana, Pavel Tokmakov, Cordelia Schmid, and Deva Ramanan. Tao: A large-scale benchmark for tracking any object. In *European conference on computer vision*, pages 436–454. Springer, 2020. 3

[9] Akshay Dhamija, Manuel Gunther, Jonathan Ventura, and Terrance Boult. The overlooked elephant of object detection: Open set. In *The IEEE Winter Conference on Applications of Computer Vision*, 2020. 3

[10] Akshay Raj Dhamija, Manuel Günther, and Terrance Boult. Reducing network agnostophobia. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2, 3, 6

[11] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311. IEEE, 2009. 4

[12] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision (IJCV)*, 2015. 3

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 1, 2, 3, 5

[14] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 1

[15] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017. 2, 3, 6

[16] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *ICLR*, 2019. 6

[17] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4507–4515, 2017. 4

[18] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. *CoRR*, abs/1912.04986, 2019. 2

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015*, 2015. 5

[20] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 12697–12705. Computer Vision Foundation / IEEE, 2019. 2, 4

[21] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014. 3

[22] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, pages 2537–2546, 2019. 1, 2

[23] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, 2020. 3

[24] Andrii Maksai and Pascal Fua. Eliminating exposure bias and metric mismatch in multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4639–4648, 2019. 3

[25] Andrii Maksai, Xinchao Wang, Francois Fleuret, and Pascal Fua. Non-markovian globally consistent multi-object tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 2544–2554, 2017. 3

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. 5

[27] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In Gang Hua and Hervé Jégou, editors, *Computer Vision - ECCV 2016 Workshops*, volume 9914 of *Lecture Notes in Computer Science*, pages 17–35, 2016. 3

[28] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *TPAMI*, 35(7):1757–1772, 2012. 2, 3

[29] Horesh Ben Shitrit, Jerome Berclaz, Francois Fleuret, and Pascal Fua. Tracking multiple people under global appearance constraints. In *2011 International conference on computer vision*, pages 137–144. IEEE, 2011. 3

[30] Leslie N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision, WACV*, pages 464–472. IEEE Computer Society, 2017. 5

[31] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 2443–2451. IEEE, 2020. 1, 5

[32] Araz Taeihagh and Hazel Si Min Lim. Governing autonomous vehicles: emerging responses for safety, liability, privacy, cybersecurity, and industry risks. *Transport Reviews*, 39(1):103–128, 2019. 1, 2

[33] Gaoang Wang, Yizhou Wang, Haotian Zhang, Renshu Gu, and Jenq-Neng Hwang. Exploit the connectivity: Multi-object tracking with trackletnet. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 482–490, 2019. 3

[34] Jun Wang, Shiyi Lan, Mingfei Gao, and Larry S. Davis. Infofocus: 3d object detection for autonomous driving with dynamic information modeling. *CoRR*, abs/2007.08556, 2020. 2

[35] Xinshuo Weng and Kris Kitani. A baseline for 3d multi-object tracking. *arXiv preprint arXiv:1907.03961*, 2019. 1, 2

[36] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017. 2

[37] Kelvin Wong, Shenlong Wang, Mengye Ren, Ming Liang, and Raquel Urtasun. Identifying unknown instances for autonomous driving. In *CoRL*, 2020. 1, 2

[38] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2

[39] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *arXiv preprint arXiv:2006.11275*, 2020. 2, 4, 5

[40] Fengwei Yu, Wenbo Li, Quanquan Li, Yu Liu, Xiaohua Shi, and Junjie Yan. Poi: Multiple object tracking with high performance detection and appearance feature. In *European Conference on Computer Vision*, pages 36–42. Springer, 2016. 8

[41] Peng Zhang, Jiuling Wang, Ali Farhadi, Martial Hebert, and Devi Parikh. Predicting failures of vision systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3566–3573, 2014. 3, 6

[42] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015. 8

[43] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019. 2, 4

[44] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014. 4

3D Multi-Object Tracking of *All Movable* Objects
(Supplementary Material)

Outline

In this supplementary document, we examine the annotation protocol used in nuScenes to demonstrate that nuScenes has annotated all movable objects, provide breakdown analysis under the Open-Set setup, and verify the generalization for our meta-approaches. Lastly, we provide video demos on selected scenes showcasing the difficulty of 3D-MOT-AMO and the effectiveness of our methods.

1. Examining nuScenes Annotations

As we noted in the main paper, the nuScenes dataset does not explicitly state that it aims to annote all movable objects. We examine its annotation guide, and perform manual inspection to verify that it annotates almost all movable objects.

nuScenes Annotator Guide. In the supplementary material of the nuScenes paper [3], Figure 8 studies the portion of "moving" objects within each class, and concludes that some classes are "rarely moving" such as `construction-vehicles`. Further, Figure 11 studies the velocity of moving objects. These figures suggest that "movable" (either moving or temporarily static) is an important attribute in the annotated objects, or generally in the context of autonomous vehicle research. Moreover, the main paper explicitly explains the scene selection process [3]:

> After collecting the raw sensor data, we manually select 1000 interesting scenes of 20s duration each. Such scenes include high traffic density (e.g. intersections, construction sites), rare classes (e.g. `ambulances`, `animals`), potentially dangerous traffic situations (e.g., jaywalkers, incorrect behavior), maneuvers (e.g., lane change, turning, stopping) and situations that may be difficult for an AV.

This implies that the annotation is a botton-up, heuristic procedure, where the annotation experts first discover interesting objects which are movable (e.g., jaywalkers and animals), and then uses this to create *a list of class labels*.

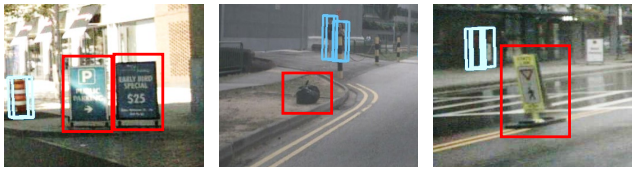

Figure 1: We manually inspect 50 random RGB frames from 50 videos in nuScenes to validate if the dataset actually annotates all movable objects. There are 455 annotated objects within the 50 frames. In addition to these labeled objects, we found 14 unannotated objects (12 are temporary parking signs are 2 are debris). We show 3 frames and highlight the unannotated objects with red boxes. Over the 50 inspected frames, nuScenes annotates 97% of all movable objects. Furthermore, we note that nuScenes does not annotate objects when LiDAR sweeps do not have any points corresponding to an object. Therefore, our RGB-based inspection potentially annotates objects that are nearly invisible in LiDAR. Overall, our inspection suggests that nuScenes annotates all movable objects.

From the list of class labels, the annotation guide [2] requests annotators to label all objects belonging to these classes, which not only contain many common movable objects such as `pedestrians` and `cars`, but also collapses many (rarely seen) movable objects into superclasses including:

- `Portable Personal Mobility Vehicle`: A small electric or self-propelled vehicle, e.g. skateboard, segway, or scooters, on which the person typically travels in a upright position. Driver and (if applicable) rider should be included in the bounding box along with the vehicle.
- `Animal`: All animals, e.g. cats, rats, dogs, deer, birds.
- `Pushable Pullable Object`: Objects that a pedestrian may push or pull. For example dolleys, wheel barrows, garbage-bins with wheels, or shopping carts. Typically not designed to carry humans.
- `Debris`: Debris or movable object that is too large to be driven over safely. Includes misc. things like trash

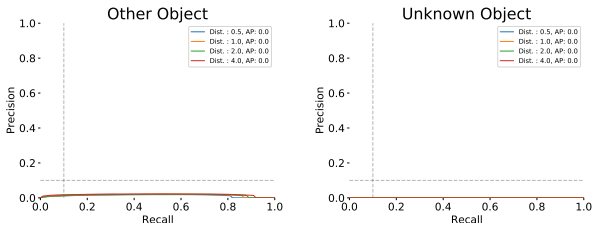

Figure 2: We show that mAP is not suitable to evaluate detection performance on `other` objects with precision-recall (PR) curves. **Left**: PR curve for detection of `other-all` objects. Note that the four lines correspond to four distance thresholds for mAP metric (details in the paper). **Right**: PR curve for detection of `other-unknown` objects. We use the *all-mov* model under the *Open-Set* setup. Clearly, mAP is zero on `other-unknown`, while the recall remains high. This prevents us from using mAP (and more generally precision based metrics) to measure detection and tracking performance on `other-unknown` objects for breakdown analysis.

bags, temporary road-signs, objects around construction zones, and trash cans.

Missing Annoations In addition to examining the annotators instructions, we also manually validate that the nuScenes dataset has annotated all movable objects through visual inspection. For a randomly selected set of 50 RGB frames from 50 distinct scenes, we annotate all movable objects not already annotated according to the guidelines. From these 50 frames, we find 12 unannotated `temporary-signs` and 2 unannotated `debris` out of 455 annotated objects, indicating a miss-rate of 3%. Furthermore, we note that nuScenes does not annotate objects when LiDAR sweeps do not have points on them. Therefore, our RGB-based inspection potentially annotates invisible objects in LiDAR. Our inspection suggests that nuScenes annotates all movable objects. Fig. 1 shows the typical visual appearance of objects within these unannotated cateogries.

2. Breakdown Evaluation

Recall that the *Open-Set* setup evaluates tracking never-before-seen objects belonging to unknown classes. This allows us to analyze detection and tracking performance on subgroups of testing examples, including all the `other` examples (`other`-all), those from the `other` classes used in training (`other`-known) and those from novel classes *only* seen in testing (`other`-unknown). To compute the metrics on a particular subgroup, we remove true positive matches for all other subgroups, and evaluate the remaining predictions. Importantly, all predictions that are unmatched to any particular subgroup are evaluated as false positives for all subgroups. This evaluation method is similar to the COCO benchmark that analyzes detection performance on different groups of object sizes [8].

Auxiliary Metrics. We now perform breakdown analysis under the *Open-Set* setup. Note that we find precision-based metrics for tracking novel movable objects to be particularly low (Fig. 2), which makes it difficult to draw reliable conclusions. While this suggests the difficulty of tracking novel objects and the need for future research on 3D-MOT-AMO, we now use two recall-based metrics to measure the performance on the movable objects from *novel* classes unseen in training.

Concretely, for detection, we report mean average recall (mAR). While mAP evaluates class-specific detection performance, mAR is useful in the evaluation of class-agnostic detection or proposal detection [6, 7, 8], hence it is suitable for measuring detection performance on the `other`-unknown class. For tracking, we use Identity Recall (IDR) [11] to evaluate data association performance on the novel `other-unknown` objects. Loosely speaking, both recall-based metrics can be computed similarly to their precision-based counterpart (mAP and IDF1 respectively) without penalizing false positives; we refer readers to the respective papers [7, 11] for their mathematical formulations.

Discussion on the gamability of recall-based metrics. As widely acknowledged, recall-based metrics can produce near-perfect measurement if the detector trivially outputs an "infinite" number of detections. To resolve this issue, a common practice is to restrict the number of per-image detections by only evaluating top-k proposals for each image w.r.t each bounding boxes' confidence score [4, 9, 10]. However, since we use precision-based metrics as the primary metric, we select models based on the overall precision rather than a high overall recall on `other`-unknown objects. In other words, we do not exploit the gamability of recall-based metrics, but simply use them to diagnose performance on subclasses of testing examples.

Sub-class evaluation for the *Open-Set* setup. We apply the sub-class evaluation protocol as described above to the *Open-Set* setup results from the main paper. We use subscripts to denote metrics for `other-known` by $_K$ (e.g., mAP_K and IDF1_K), `other-unknown` by $_U$ (e.g., mAR_U and IDR_U), and all the `other` movable objects by $_A$ (e.g., mAP_A and IDF1_A).

Table 1 and 2 list detailed results for detection and tracking under the *Open-Set* setup. For brevity, we do not repeat the conclusions shown in the paper. However, the subclass evaluation reveals that while all methods achieve similar overall detection performance (mAP) in Table 1, they perform quite differently on tracking `other` objects. Especially, (0+1) and *all-mov* models appear to perform quite well in detecting `other-unknown` and achieve high recall at the cost of detection precisions on `other-known` for the mAP_A and mAP_K metrics. This suggests that careful grouping of training examples and model design is crucial for detecting `other` objects with high precision and

Table 1: **Breakdown analysis on 3D detection** in 3D-MOT-AMO under the *Open-Set* setup. We do not repeat conclusions which are already shown in Table 2 of the main paper. We summarize one additional salient observation: while all methods achieve similar overall detection performance (mAP) in Table 1, they perform quite differently in tracking `other` objects. Specifically, (0+1) and *all-mov* models appear to perform quite well in detecting `other-unknown` and achieve high recall, at the cost of decreased detection precision on `other-known`. This suggests that careful grouping of training examples and the model design is crucial for detecting `other` objects with high precision and high recall. Future research should focus on new methods for solving this challenging problem.

Method	`Common`	`Other`			`All`
	mAP	mAP_A	mAP_K	mAR_U	mAP
(K+0)	**59.1**	0	0	0	53.7
(K+N)	57.7	19.2	20.5	33.9	54.2
(K+1)	57.8	19.0	20.3	34.2	54.2
(1+1)	59.0	**20.7**	**20.7**	32.6	**55.5**
(0+1)	58.9	17.6	17.6	47.3	55.1
all-mov	56.9	0	0	**91.7**	51.7

Table 2: **Breakdown analysis on 3D tracking** in 3D-MOT-AMO under the *Open-Set* setup. As the same conclusions hold in detection metrics, we refer readers to the caption of Table 1 for details.

Method	Common	`Other`			All
	IDF_1	IDF_{1A}	IDF_{1K}	IDR_U	IDF_1
(K+0)	**71.1**	0	0	0	64.7
(K+N)	70.4	6.5	8.4	27.1	64.6
(K+1)	70.1	7.5	8.9	27.7	64.4
(1+1)	70.6	**10.2**	12.0	30.9	**65.1**
(0+1)	70.7	4.7	**23.4**	42.0	64.7
all mov	70.1	2.5	2.8	**91.0**	63.9

high recall. We observe the same trend for tracking in Table 2. Admittedly, recall-based metrics inflate performance as it is poorly correlated with precision, while precision-based metrics do not effectively measure the performance on the detection and tracking of `other-unknown` objects. From this clear gap, we posit that 3D-MOT-AMO under the *Open-Set* is more challenging than one would have thought. Therefore, we solicit future work on solving this problem.

3. Exploration of 3D-MOT-AMO with an Anchor-Based Detector

To validate the generalaity of our proposed meta-approaches, we repeat the grouping experiments using CBGS [14], an effective anchor-based 3D detector. We associate the predicted bounding boxes with the greedy aggregation techinque based on predicted velocity offsets described in the main paper. We study other trackers in the next section. Tables 3, 4, and 5 list the comprehensive results, showing that our primary conclusions still hold: (1) careful grouping is essential in order to effectively detect and track `other` objects, and (2) jointly training `common` objects with `other` objects improves overall performance. (cf. (1+1) vs. (0+1)).

Table 3: **Using CBGS-detector** for 3D-MOT-AMO under the *Default* setup. We compare different grouping methods for 3D detection and tracking of all movable objects. All models, except for the (K+0) and (0+1) models, are trained with both the K common classes and the $(K{+}1)^{th}$ `other` class. We combine the (0+1), (1+1) and *all-mov* with the (K+0)-way CBGS model in order to fully address 3D-MOT-AMO. Surprisingly, (K+N) model achieves "zero" performance on `other` objects. We posit that this is because of the class balanced sampling in training, which forces the training to oversample the N rare `other` classes to train a model that can fairly generalize. Consistent with the results reported in the main paper, incorporating the K common classes helps train a better tracker for `other` objects, cf. (1+1) improves over the (0+1) on IDF_1, presumably because the K common classes clearly define negative examples for `other` objects. Moreover, careful grouping of classes greatly improves tracking performance, cf. (1+1) vs. (K+1), and (1+1) vs. *all-mov*

Method	`Common`		`Other`		`All`	
	mAP	IDF_1	mAP	IDF_1	mAP	IDF_1
(K+0)	**52.0**	67.0	0	0	47.3	60.9
(K+N)	51.5	**67.3**	0	0	46.8	61.2
(K+1)	51.3	66.1	14.1	15.8	47.9	61.5
(1+1)	51.9	66.8	**15.6**	**37.1**	**48.6**	**64.1**
(0+1)	51.9	66.6	12.7	7.8	48.3	61.2
all-mov	50.2	65.2	0	3.2	45.7	59.6

From Table 3, surprisingly, the (K+N) model is unable to detect and track `other` objects. We posit that this is because of the class balanced sampling in training, which heavily weights the N rare `other` classes. Moreover, the overall performance of CBGS is slightly lower than CenterPoint, e.g., (1+1) CBGS achieves 48.6 mAP and 64.1 IDF1, as opposed to (1+1) CenterPoint that acheives 55.5 mAP and 65.1 IDF1, for the *Default* setup. Lastly, under the *Open-Set* setup (Table 3), CGBS largely underperforms CenterPoint, e.g., (1+1) CBGS achieves 16.3 mAP and 48.7 IDF1, as opposed to (1+1) CenterPoint that acheives 20.7 mAP and 65.1 IDF1.

Table 4: **Breakdown analysis on 3D detection by using CBGS-detector** for 3D-MOT-AMO under the *Open-Set* setup. Here we reiterate the same conclusions as reported in the main paper: (1) training with `other` examples boosts performance on the `other` class, cf. (K+1) vs (K+0); (2) training with both K common classes and the `other` class helps tracking `others`, cf. (1+1) vs. (0+1); (3) careful grouping has a significant impact on overall performance, cf. (1+1) vs. *all-mov*. Similar to Table 2, *all-move* achieves high recall on `other-unknown` but very low precision. This implies that `all-move` overly generalizes and detects many false positives.

Method	`Common`	`Other`			`All`
	mAP	mAP_A	mAP_K	mAR_U	mAP
(K+0)	**52.0**	0	0	0	47.3
(K+N)	40.7	8.9	9.5	13.4	37.8
(K+1)	51.1	14.5	15.6	13.2	47.8
(1+1)	51.9	**16.3**	**17.4**	21.0	**48.7**
(0+1)	51.9	12.3	13.2	35.0	48.3
all-mov	50.3	0	0	**88.6**	45.7

Table 5: **Breakdown analysis on 3D tracking by using CBGS-detector** for 3D-MOT-AMO under the *Open-Set* setup. Please refer to the caption of Table 4 for the salient conclusions.

Method	Common	`Other`			All
	IDF_1	IDF_{1A}	IDF_{1K}	IDR_U	IDF_1
(K+0)	**67.3**	0	0	0	61.2
(K+N)	55.5	13.2	14.7	15.4	51.2
(K+1)	65.7	**34.5**	**36.4**	11.2	62.8
(1+1)	66.7	33.5	33.9	16.0	**63.7**
(0+1)	66.6	23.9	24.2	**31.4**	62.8
all mov	65.1	3.1	2.3	84.0	59.5

4. Exploration of Other Trackers

Although our prior approaches have exclusively focused on greedy point based track linking, we explore two other data association methods.

- **AB3DMOT** [12] is a simple extension of 2D Kalman Filter based tracking algorithms like SORT [1] for 3D tracking applications. It serves as a simple, yet effective baseline for tracking-by-detection algorithms.
- **Chiu *et al.*** [5] improves upon the simple Kalman Filter approach proposed in [12]. Specifically, it uses the statistics of the training data to estimate the initial state covariance matrix and the observation noise covariance. This method is particularly effective because it models the variance in ground truth accelerations, better linking detections corresponding to the same object.

Table 6: **Study of using other trackers** for 3D-MOT-AMO under the *Default* setup. We explore two more data association methods, AB3DMOT [12] and Chiu *et al*. [5], in addition to CenterPoint [13] to determine which approach maximizes IDF_1. Generally, in terms of overall performance (cf. numbers for `all`), Chiu *et al*. performs better than AB3DMOT, and CenterPoint works the best.

Detections	Default		
w/ Tracker	`Common`	`Other`	`All`
(K + 0)			
w/ AB3DMOT [12]	68.1	0	61.9
w/ Chiu *et al*. [5]	69.1	0	62.8
w/ CenterPoint [13]	**71.1**	0	**64.7**
(K + N)			
w/ AB3DMOT [12]	66.8	19.3	62.5
w/ Chiu *et al*. [5]	68.7	18.7	64.2
w/ CenterPoint [13]	**70.2**	**28.2**	**66.4**
(K + 1)			
w/ AB3DMOT [12]	67.2	9.5	61.9
w/ Chiu *et al*. [5]	69.5	25.5	65.5
w/ CenterPoint [13]	**71.4**	**34.9**	**68.0**
(1 + 1)			
w/ AB3DMOT [12]	66.6	**10.6**	61.5
w/ Chiu *et al*. [5]	68.5	9.8	63.1
w/ CenterPoint [13]	**70.7**	9.7	**65.1**
(0 + 1)			
w/ AB3DMOT [12]	66.6	20.1	62.3
w/ Chiu *et al*. [5]	68.7	**22.5**	64.5
w/ CenterPoint [13]	**70.6**	5.6	**64.7**
all-mov			
w/ AB3DMOT [12]	65.8	**3.0**	60.1
w/ Chiu *et al*. [5]	67.9	2.7	62.0
w/ CenterPoint [13]	**70.0**	2.7	**63.8**

In general, Table 6 shows that among these two methods, Chiu *et al*. consistently performs better than AB3DMOT, and CenterPoint outperforms Chiu *et al*. Part of this improvement can be attributed to how each method leverages motion information. AB3DMOT adapts a standard Kalman Filter to 3D bounding boxes, and does not explicitly leverage motion information. Chiu *et al*. initializes the Kalman filter with per-class motion-information. CenterPoint, regresses the velocity offset at each timestep, and utilizes per-instance motion information. This implies that explicitly leveraging motion cues is essential for 3D-MOT-AMO.

5. Further Study of Track-NMS

Let us first walk through the implementation details of Track-NMS and analyze key hyperparameters in the effectiveness of the algorithm.

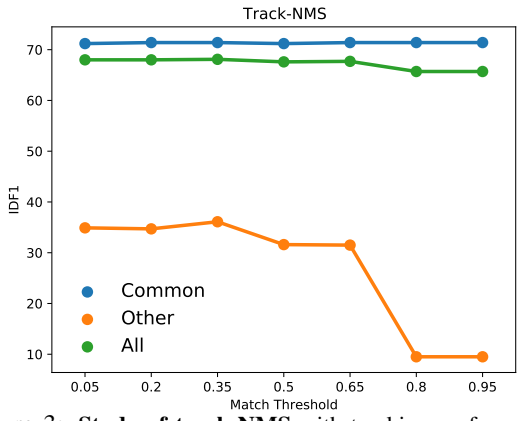

Figure 3: **Study of track-NMS** with tracking performance vs. matching threshold, which is defined as the center-distance of the bounding boxes in meters in the BEV perspective. Bounding boxes with center-distances less than the matching threshold are considered to be overlapping. To study this, we use the (K+1) model in the *Default* setup. Using different match thresholds for the NMS algorithm shows the sensetivity of the proposed approch. Track-NMS has little effect on `common` object tracking performance, but has a modest impact on the tracking performance of `others`. Arbitrarily increasing the match threshold will cause the algorithm to compare two distinct, but spatially close objects that should not be suppressed, causing the significant decrease in IDF_1.

Track-NMS can be thought of as applying NMS on overlapping detections after linking the detections into tracks. Concretely, we first find overlapping detections between classes and suppress the boxes belonging to the `other` class. To find overlapping detections, we measure the ground-plane center distance of boxes, i.e. distance in the birds-eye view (BEV) perspective. If the distance is less than a *matching threshold*, we group these as overlapping detections and apply NMS. Two key insights helped improve the tracking performance on `other` objects. First, we perform this inter-class NMS at the track-level rather than the detection level because suppressing bounding boxes without temporal context can potentially lead to track fragmentation. Second, we only allow `common` bounding boxes to suppress `other` bounding boxes because `common` objects are distinct and unambiguously defined, and hence are less likely to result in overlappped bounding boxes.

Track overlap between `common` and `other` classes is an important problem to consider when grouping classes and combining models in a two-stream fashion, e.g., jointly using (1+1) and (K+0) for 3D-MOT-AMO. When grouping classes together, the inductive bias learned for `other` objects causes overlapping detections between `common` and `other` objects. To reduce the number of false positives, we apply track-level NMS on overlapping bounding boxes in tracks from different classes to appropriately remove the incorrect class bounding box. If tuned effectively, Figure 3 highlights that Track-NMS improves both tracking performance on `other` objects, as well as overall tracking performance.

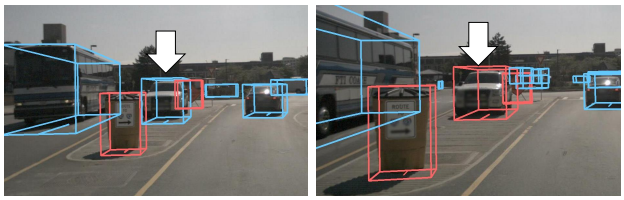

Figure 4: From our video demo, we find that the ground-truth annotations change over time. We show a concrete example above, where the `police-car` is labeled as a `car` (a common class) at a distance, but is relabeled as a `police-car` (a rare class) when the ego car drives closer. This is surprising w.r.t the annotation but understandable in the real world because we human drivers may also have difficulty distinguishing between a `police-car` and a normal `car` at a distance.

6. Video Demo

We include demo videos that display tracking results on an interesting scene from our test-set. We use the (K+1)-way model under the *Default* setup. Specifically, we include three videos as below. We are interested in showing tracking results by differentiating `common` and `other` objects, without showing K labels within the `common` class. Therefore, we color the tracked `common` and `other` objects with shades of blue and red, respectively.

- (K+1).mp4 shows tracking results from the (K+1) model. Note that we suppress tracks with a score less than 0.3 to remove low-confidence tracks, and do not use other techniques to post-process the tracking results.
- (K+0).mp4 shows tracking results from a (K+0) model, which is not trained for 3D-MOT-AMO. Obviously, this model does not detect and track `other` objects.
- GroundTruth.mp4 shows ground-truth annotations on all frames. Interestingly, we find the `police-car` is labeled as a `car` (one of the `common` class) at a distance, but changes to a `police-car` when the ego car gets closer to it, as illustrated by Fig. 4. This is surprising because of the changed ground-truth labels, yet understandable because human drivers also need closer inspection to distinguish `police-car` from normal `cars`. This also explains why our (K+1) model's predicted labels for this vehicle switches between `common` and `other`.

References

[1] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016*

IEEE International Conference on Image Processing (ICIP), pages 3464–3468. IEEE, 2016. 4

[2] Holger Caesar. https://github.com/nutonomy/nuscenes-devkit/blob/master/docs/instructions_nuscenes.md#instructions. 1

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020. 1

[4] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3286–3293, 2014. 2

[5] Hsu-kuang Chiu, Antonio Prioletti, Jie Li, and Jeannette Bohg. Probabilistic 3d multi-object tracking for autonomous driving. *CoRR*, abs/2001.05673, 2020. 4

[6] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *European conference on computer vision*, pages 340–353. Springer, 2012. 2

[7] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What makes for effective detection proposals? *IEEE transactions on pattern analysis and machine intelligence*, 38(4):814–830, 2015. 2

[8] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014. 2

[9] Pedro OO Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015. 2

[10] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European conference on computer vision*, pages 75–91. Springer, 2016. 2

[11] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In Gang Hua and Hervé Jégou, editors, *Computer Vision - ECCV 2016 Workshops*, volume 9914 of *Lecture Notes in Computer Science*, pages 17–35, 2016. 2

[12] Xinshuo Weng and Kris Kitani. A baseline for 3d multi-object tracking. *arXiv preprint arXiv:1907.03961*, 2019. 4

[13] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *arXiv preprint arXiv:2006.11275*, 2020. 4

[14] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019. 3