nuScenes: A multimodal dataset for autonomous driving Supplementary Material

A. The nuScenes dataset

In this section we provide more details on the nuScenes dataset, the sensor calibration, privacy protection approach, data format, class mapping and annotation statistics.

Sensor calibration. To achieve a high quality multi-sensor dataset, careful calibration of sensor intrinsic and extrinsic parameters is required. These calibration parameters are updated around twice per week over the data collection period of 6 months. Here we describe how we perform sensor calibration for our data collection platform to achieve a high-quality multimodal dataset. Specifically, we carefully calibrate the extrinsics and intrinsics of every sensor. We express extrinsic coordinates of each sensor to be relative to the *ego frame*, i.e. the midpoint of the rear vehicle axle. The most relevant steps are described below:

- Lidar extrinsics: We use a laser liner to accurately measure the relative location of the lidar to the ego frame.
- Camera extrinsics: We place a cube-shaped calibration target in front of the camera and lidar sensors. The calibration target consists of three orthogonal planes with known patterns. After detecting the patterns we compute the transformation matrix from camera to lidar by aligning the planes of the calibration target. Given the lidar to ego frame transformation computed above, we compute the camera to ego frame transformation.
- Radar extrinsics: We mount the radar in a horizontal position. Then we collect radar measurements by driving on public roads. After filtering radar returns for moving objects, we calibrate the yaw angle using a brute force approach to minimize the compensated range rates for static objects.
- Camera intrinsic calibration: We use a calibration target board with a known set of patterns to infer the intrinsic and distortion parameters of the camera.

Privacy protection. It is our priority to protect the privacy of third parties. As manual labeling of faces and license plates is prohibitively expensive for 1.4M images, we use state-of-the-art object detection techniques. Specifically for plate detection, we use Faster R-CNN [67] with ResNet-101 backbone [39] trained on Cityscapes [19][7]. For face detection, we use [87][8]. We set the classification threshold to achieve an extremely high recall (similar to [31]). To increase the precision, we remove predictions that do not overlap with the reprojections of the known *pedestrian* and *vehicle* boxes in the image. Eventually we use the predicted boxes to blur faces and license plates in the images.

[7] https://github.com/bourdakos1/Custom-Object-Detection

[8] https://github.com/TropComplique/mtcnn-pytorch

General nuScenes class	Detection class	Tracking class
animal	void	void
debris	void	void
pushable_pullable	void	void
bicycle_rack	void	void
ambulance	void	void
police	void	void
barrier	barrier	void
bicycle	bicycle	bicycle
bus.bendy	bus	bus
bus.rigid	bus	bus
car	car	car
construction	construction_vehicle	void
motorcycle	motorcycle	motorcycle
adult	pedestrian	pedestrian
child	pedestrian	pedestrian
construction_worker	pedestrian	pedestrian
police_officer	pedestrian	pedestrian
personal_mobility	void	void
stroller	void	void
wheelchair	void	void
trafficcone	traffic_ cone	void
trailer	trailer	trailer
truck	truck	truck

Table 5. Mapping from general classes in nuScenes to the classes used in the detection and tracking challenges. Note that for brevity we omit most prefixes for the general nuScenes classes.

Data format. Contrary to most existing datasets [32, 61, 41], we store the annotations and metadata (e.g. localization, timestamps, calibration data) in a relational database which avoids redundancy and allows for efficient access. The nuScenes devkit, taxonomy and annotation instructions are available online[9].

Class mapping. The nuScenes dataset comes with annotations for 23 classes. Since some of these only have a handful of annotations, we merge similar classes and remove classes that have less than 10000 annotations. This results in 10 classes for our detection task. Out of these, we omit 3 classes that are mostly static for the tracking task. Table 5-SM shows the detection classes and tracking classes and their counterpart in the general nuScenes dataset.

Annotation statistics. We present more statistics on the annotations of nuScenes. Absolute velocities are shown in Figure 11-SM. The average speed for moving *car*, *pedestrian* and *bicycle* categories are 6.6, 1.3 and 4 m/s. Note that our data was gathered from urban areas which shows reasonable velocity range for these three categories.

[9] https://github.com/nutonomy/nuscenes-devkit

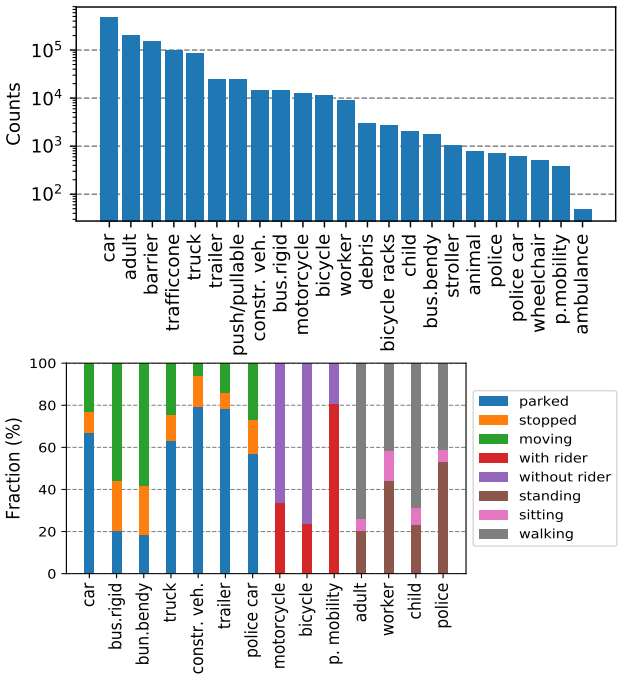

Figure 8. Top: Number of annotations per category. Bottom: Attributes distribution for selected categories. Cars and adults are the most frequent categories in our dataset, while ambulance is the least frequent. The attribute plot also shows some expected patterns: construction vehicles are rarely moving, pedestrians are rarely sitting while buses are commonly moving.

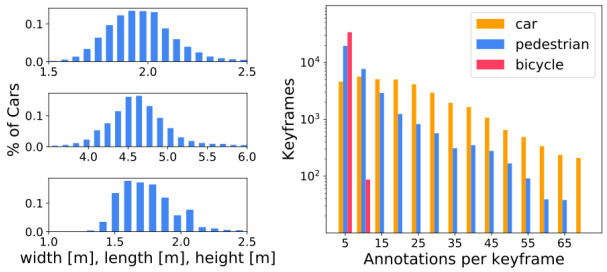

Figure 9. Left: Bounding box size distributions for *car*. Right: Category count in each keyframe for *car*, *pedestrian*, and *bicycle*.

We analyze the distribution of box annotations around the ego-vehicle for *car*, *pedestrian* and *bicycle* categories through a polar range density map as shown in Figure 12-SM. Here, the occurrence bins are log-scaled. Generally, the annotations are well-distributed surrounding the ego-vehicle. The annotations are also denser when they are nearer to the ego-vehicle. However, the *pedestrian* and *bicycle* have less annotations above the 100m range. It can also be seen that the *car* category is denser in the front and back of the ego-vehicle, since most vehicles are following the same lane as the ego-vehicle.

In Section 2 we discussed the number of lidar points inside a box for all categories through a hexbin density plot, but here we present the number of lidar points of each category as shown in Figure 13-SM. Similarly, the occurrence bins are log-scaled. As can be seen, there are more lidar points found inside the box annotations for *car* at varying distances from the ego-vehicle as compared to *pedestrian* and *bicycle*. This is expected as cars have larger and more reflective surface area than the other two categories, hence more lidar points are reflected back to the sensor.

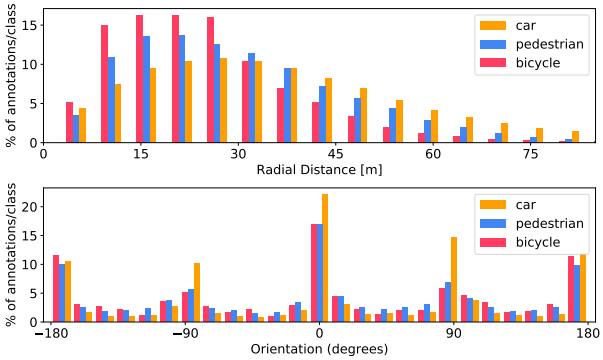

Figure 10. Top: radial distance of objects from the ego vehicle. Bottom: orientation of boxes in box coordinate frame.

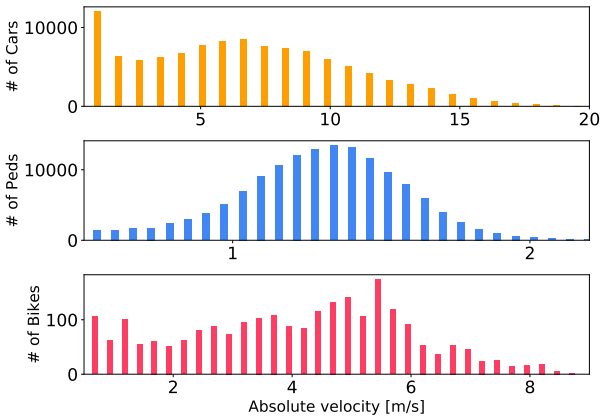

Figure 11. Absolute velocities. We only look at moving objects with speed > 0.5m/s.

Scene reconstruction. nuScenes uses an accurate lidar based localization algorithm (Section 2). It is however difficult to quantify the localization quality, as we do not have ground truth localization data and generally cannot perform loop closure in our scenes. To analyze our localization qualitatively, we compute the merged pointcloud of an entire scene by registering approximately 800 pointclouds in global coordinates. We remove points corresponding to the ego vehicle and assign to each point the mean color value of the closest camera pixel that the point is reprojected to. The result of the scene reconstruction can be seen in Figure 15, which demonstrates accurate synchronization and localization.

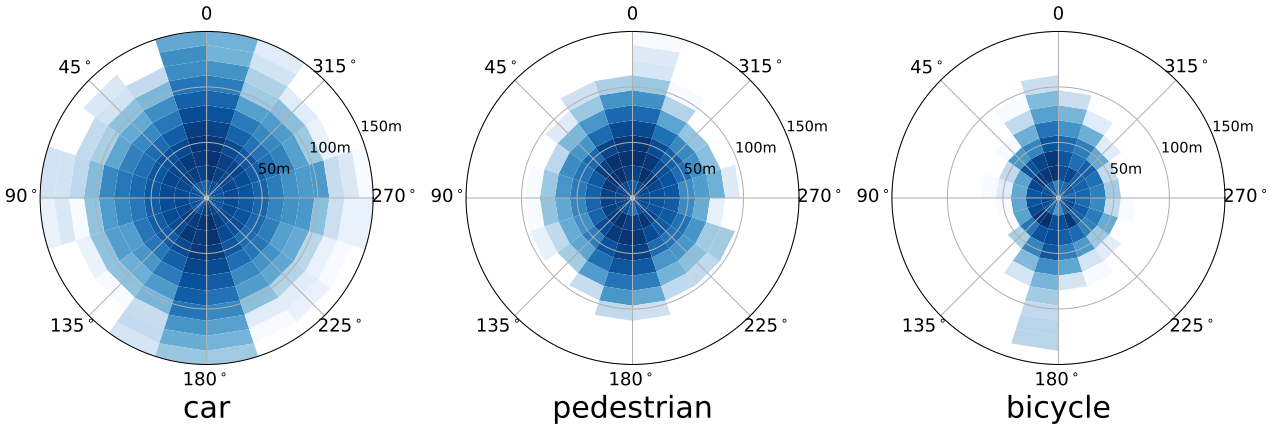

Figure 12. Polar log-scaled density map for box annotations where the radial axis is the distance from the ego-vehicle in meters and the polar axis is the yaw angle wrt to the ego-vehicle. The darker the bin is, the more box annotations in that area. Here, we only show the density up to 150m radial distance for all maps, but *car* would have annotations up to 200m.

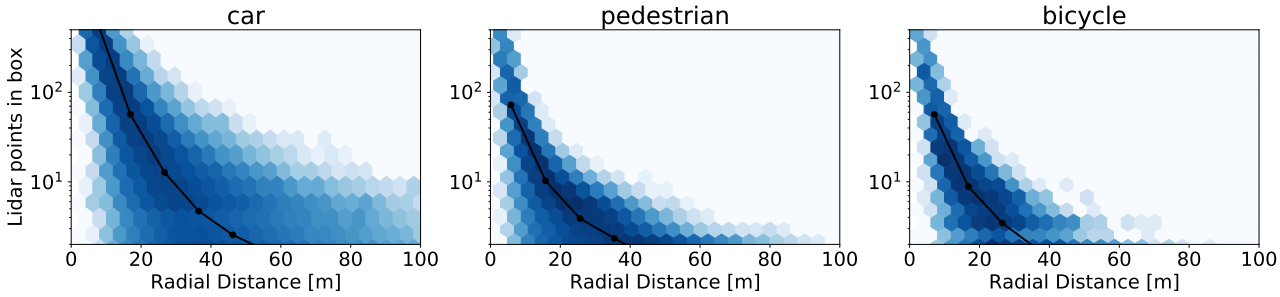

Figure 13. Hexbin log-scaled density plots of the number of lidar points inside a box annotation stratified by categories (*car*, *pedestrian* and *bicycle*.

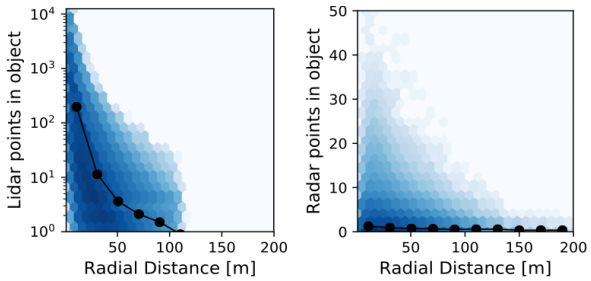

Figure 14. Hexbin log-scaled density plots of the number of lidar and radar points inside a box annotation. The black line represents the mean number of points for a given distance wrt the ego-vehicle.

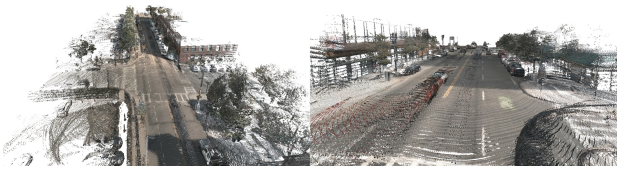

Figure 15. Sample scene reconstruction given lidar points and camera images. We project the lidar points in an image plane with colors assigned based on the pixel color from the camera data.

B. Implementation details

Here we provide additional details on training the lidar and image based 3D object detection baselines.

PointPillars implementation details. For all experiments, our PointPillars [51] networks were trained using a pillar xy resolution of 0.25 meters and an x and y range of $[-50, 50]$ meters. The max number of pillars and batch size was varied with the number of lidar sweeps. For 1, 5, and 10 sweeps, we set the maximum number of pillars to 10000, 22000, and 30000 respectively and the batch size to 64, 64, and 48. All experiments were trained for 750 epochs. The initial learning rate was set to 10^{-3} and was reduced by a factor of 10 at epoch 600 and again at 700. Only ground truth annotations with one or more lidar points in the accumulated pointcloud were used as positive training examples. Since bikes inside of bike racks are not annotated individually and the evaluation metrics ignore bike racks, all lidar points inside bike racks were filtered out during training.

OFT implementation details. For each camera, the Orthographic Feature Transform [69] (OFT) baseline was trained on a voxel grid in each camera's frame with an lateral range of $[-40, 40]$ meters, a longitudinal range of $[0.1, 50.1]$ meters and a vertical range of $(-3, 1)$ meters.

Method	Singapore	Rain	Night
OFT [69]†	6%	10%	**55%**
MDIS [70]†	8%	-3%	**58%**
PP [51]	1%	6%	**36%**

Table 6. Object detection performance drop evaluated on subsets of the nuScenes val set. Performance is reported as the relative drop in mAP compared to evaluating on the entire val set. We evaluate the performance on Singapore data, rain data and night data for three object detection methods. Note that the MDIS results are not directly comparable to other sections of this work, as a ResNet34 [39] backbone and a different training protocol are used. (†) use only monocular camera images as input. PP uses only lidar.

We trained only on annotations that were within 50 meters of the car's ego frame coordinate system's origin. Using the 'visibility' attribute in the nuScenes dataset, we also filtered out annotations that had visibility less than 40%. The network was trained for 60 epochs using a learning rate of 2×10^{-3} and used random initialization for the network weights (no ImageNet pretraining).

C. Experiments

In this section we present more detailed result analysis on nuScenes. We look at the performance on rain and night data, per-class performance and semantic map filtering. We also analyze the results of the tracking challenge.

Performance on rain and night data. As described in Section 2, nuScenes contains data from 2 countries, as well as rain and night data. The dataset splits (train, val, test) follow the same data distribution with respect to these criteria. In Table 6 we analyze the performance of three object detection baselines on the relevant subset of the val set. We can see a small performance drop for Singapore as compared to the overall val set (USA and Singapore), particularly for vision based methods. This is likely due to different object appearance in the different countries, as well as different label distributions. For rain data we see only a small decrease in performance on average, with worse performance for OFT and PP, and slightly better performance for MDIS. One reason is that the nuScenes dataset annotates any scene with raindrops on the windshield as rainy, regardless of whether there is ongoing rainfall. Finally, night data shows a drastic performance relative drop of 36% for the lidar based method and 55% and 58% for the vision based methods. This may indicate that vision based methods are more affected by worse lighting. We also note that night scenes have very few objects and it is harder to annotate objects with bad visibility. For annotating data, it is essential to use camera *and* lidar data, as described in Section 2.

Per-class analysis. The per class performance of PointPillars [51] is shown in Table 7-SM (top) and Figure 17-SM. The network performed best overall on cars and pedestrians which are the two most common categories. The worst performing categories were bicycles and construction vehicles, two of the rarest categories that also present additional challenges. Construction vehicles pose a unique challenge due to their high variation in size and shape. While the translational error is similar for cars and pedestrians, the orientation error for pedestrians (21°) is higher than that of cars (11°). This smaller orientation error for cars is expected since cars have a greater distinction between their front and side profile relative to pedestrians. The vehicle velocity estimates are promising (e.g. 0.24 m/s AVE for the *car* class) considering the typical speed of a vehicle in the city would be 10 to 15 m/s.

PointPillars						
Class	**AP**	**ATE**	**ASE**	**AOE**	**AVE**	**AAE**
Barrier	38.9	0.71	0.30	0.08	N/A	N/A
Bicycle	1.1	0.31	0.32	0.54	0.43	0.68
Bus	28.2	0.56	0.20	0.25	0.42	0.34
Car	68.4	0.28	0.16	0.20	0.24	0.36
Constr. Veh.	4.1	0.89	0.49	1.26	0.11	0.15
Motorcycle	27.4	0.36	0.29	0.79	0.63	0.64
Pedestrian	59.7	0.28	0.31	0.37	0.25	0.16
Traffic Cone	30.8	0.40	0.39	N/A	N/A	N/A
Trailer	23.4	0.89	0.20	0.83	0.20	0.21
Truck	23.0	0.49	0.23	0.18	0.25	0.41
Mean	30.5	0.52	0.29	0.50	0.32	0.37
MonoDIS						
Class	**AP**	**ATE**	**ASE**	**AOE**	**AVE**	**AAE**
Barrier	51.1	0.53	0.29	0.15	N/A	N/A
Bicycle	24.5	0.71	0.30	1.04	0.93	0.01
Bus	18.8	0.84	0.19	0.12	2.86	0.30
Car	47.8	0.61	0.15	0.07	1.78	0.12
Constr. Veh.	7.4	1.03	0.39	0.89	0.38	0.15
Motorcycle	29.0	0.66	0.24	0.51	3.15	0.02
Pedestrian	37.0	0.70	0.31	1.27	0.89	0.18
Traffic Cone	48.7	0.50	0.36	N/A	N/A	N/A
Trailer	17.6	1.03	0.20	0.78	0.64	0.15
Truck	22.0	0.78	0.20	0.08	1.80	0.14
Mean	30.4	0.74	0.26	0.55	1.55	0.13

Table 7. Detailed detection performance for PointPillars [51] (top) and MonoDIS [70] (bottom) on the test set. AP: average precision averaged over distance thresholds (%), ATE: average translation error (m), ASE: average scale error (1-IOU), AOE: average orientation error (rad), AVE: average velocity error (m/s), AAE: average attribute error ($1 - acc.$), N/A: not applicable (Section 3.1). nuScenes Detection Score (NDS) = 45.3% (PointPillars) and 38.4% (MonoDIS).

Semantic map filtering. In Section 4.2 and Table 7-SM we show that the PointPillars baseline achieves only an AP of 1% on the *bicycle* class. However, when filtering both the predictions and ground truth to only include boxes on the semantic map prior[10], the AP increases to 30%. This observation can be seen in Figure 16-SM, where we plot the AP at different distances of the ground truth to the semantic map prior. As seen, the AP drops when the matched GT is

[10]Defined here as the union of roads and sidewalks.

Method	sAMOTA (%)	AMOTP (m)	$sMOTA_r$ (%)	MOTA (%)	MOTP (m)	TID (s)	LGD (s)
Stan [16]	55.0	0.80	76.8	45.9	0.35	0.96	1.38
VVte	37.1	1.11	68.4	30.8	0.41	0.94	1.58
Megvii [90]	15.1	1.50	55.2	15.4	0.40	1.97	3.74
CeOp	10.8	0.99	26.7	8.5	0.35	1.72	3.18
CeVi†	4.6	1.54	23.1	4.3	0.75	2.06	3.82
PP [51]	2.9	1.70	24.3	4.5	0.82	4.57	5.93
MDIS [70]†	1.8	1.79	9.1	2.0	0.90	1.41	3.35

Table 8. Tracking results on the test set of nuScenes. PointPillars, MonoDIS (MaAB) and Megvii (MeAB) are submissions from the detection challenge, each using the AB3DMOT [77] tracking baseline. StanfordIPRL-TRI (Stan), VVte (VV-team), CenterTrack-Open (CeOp) and CenterTrack-Vision (CeVi) are the top submissions to the nuScenes tracking challenge leaderboard. (†) use only monocular camera images as input. CeOp uses lidar and camera. All other methods use only lidar.

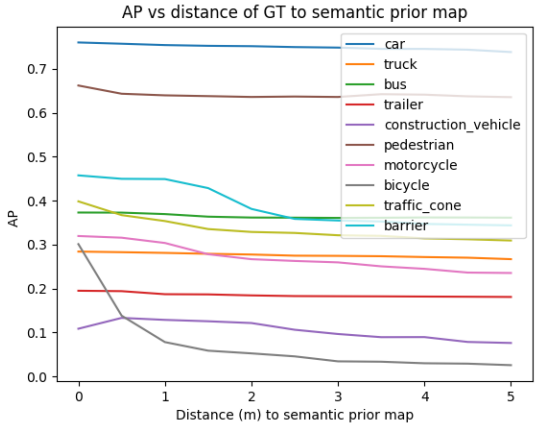

Figure 16. PointPillars [51] detection performance vs. semantic prior map location on the val set. For the best lidar network (10 lidar sweeps with ImageNet pretraining), the predictions and ground truth annotations were only included if within a given distance of the semantic prior map.

farther from the semantic map prior. Again, this is likely because bicycles away from the semantic map tend to be parked and occluded with low visibility.

Tracking challenge results. In Table 8 we present the results of the 2019 nuScenes tracking challenge. Stan [16] use the Mahalanobis distance for matching, significantly outperforming the strongest baseline (+40% sAMOTA) and setting a new state-of-the-art on the nuScenes tracking benchmark. As expected, the two methods using only monocular camera images perform poorly (CeVi and MDIS). Similar to Section 4, we observe that the metrics are highly correlated, with notable exceptions for MDIS LGD and CeOp AMOTP. Note that all methods use a tracking-by-detection approach. With the exception of CeOp and CeVi, all methods use a Kalman filter [44].

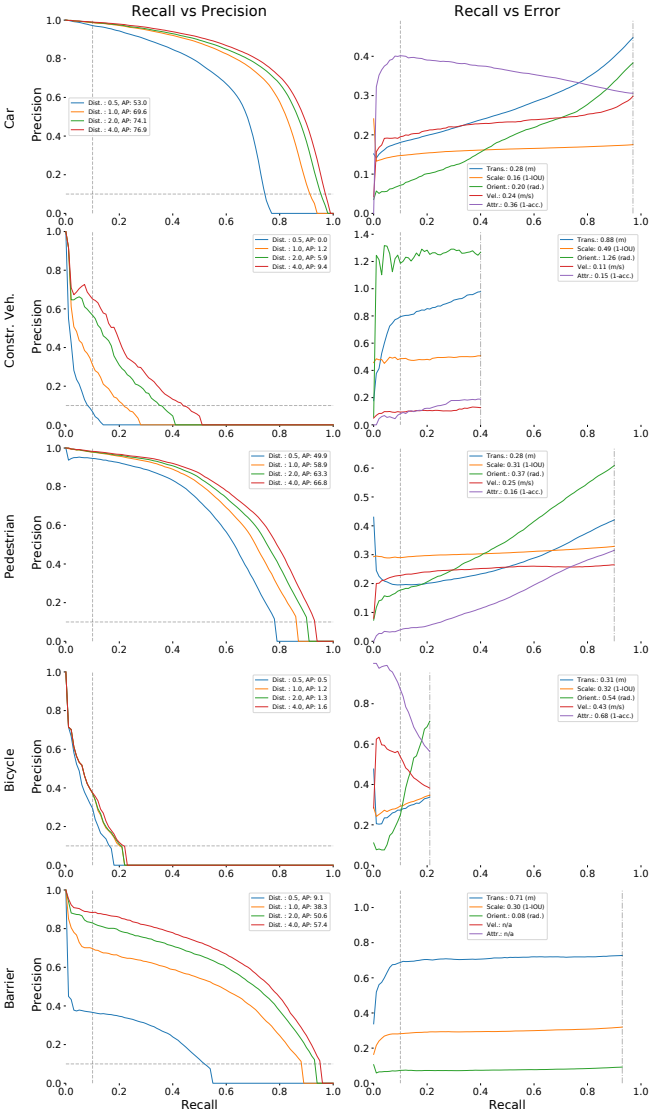

Figure 17. Per class results for PointPillars on the nuScenes test set taken from the detection leaderboard.