Supplementary Material for
Object Re-Identification from Point Clouds

A. Extended experimental details

The following section contains further details of the models used to conduct our experiments.

A.1. 3D object detectors

We generate our ReID datasets using point cloud samples cropped from predicted 3D bounding boxes and image samples cropped from their projections onto images. Here, we describe the details of the object detectors used to predict these bounding boxes.

Class	Level 1		Level 2	
	mAP	mAPH	mAP	mAPH
Car	68.3	67.8	60.7	60.2
Truck	34.2	33.5	32.2	31.6
Bus	49.0	48.6	42.6	42.3
Motorcycle	53.1	52.2	38.0	37.4
Cyclist	68.4	67.0	65.8	64.5
Pedestrian	65.9	59.7	58.0	52.4

Table 5. **Performance of the CenterPoint model on the WOD with enhanced vehicle labels.** The metrics are only calculated on samples with an enhanced label.

Class	AP	ATE	ASE	AOE	AVE	AAE
Car	89.2	0.170	0.148	0.061	0.274	0.185
Truck	64.6	0.326	0.181	0.093	0.247	0.217
Bus	75.3	0.338	0.189	0.069	0.430	0.274
Trailer	42.5	0.520	0.201	0.610	0.214	0.140
Const. Veh.	30.4	0.735	0.431	0.797	0.118	0.295
Pedestrian	88.2	0.134	0.288	0.387	0.217	0.101
Motorcycle	78.6	0.184	0.249	0.216	0.348	0.271
Bicycle	65.1	0.169	0.257	0.411	0.190	0.015
Traffic Cone	79.5	0.121	0.317	-	-	-
Barrier	72.0	0.178	0.277	0.054	-	-
NDS: 0.714	68.5	0.288	0.254	0.300	0.255	0.187

Table 6. **Performance of the BEVfusion C+L model on the nuScenes validation set.** This model was used to generate the nuScenes detections for tracking and to create the nuScenes ReID dataset.

nuScenes On the nuScenes dataset, we use BEVfusion C+L [28] to extract detections for constructing our ReID dataset. Table 6 shows its performance on the nuScenes validation set.

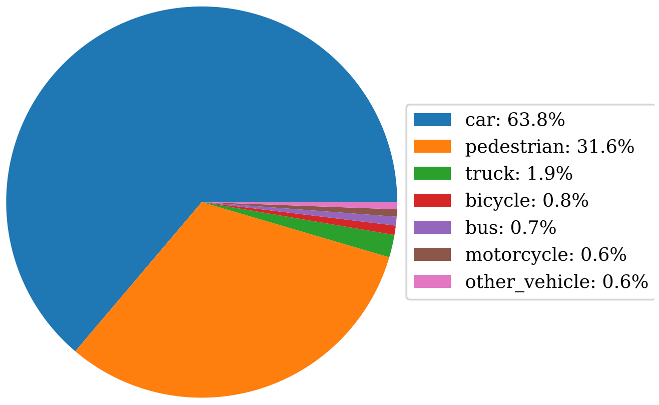


Figure 6. **Waymo class label split after propagating segmentation annotations.** The plot shows the proportions of different types of objects in the Waymo training set. We note that those samples belonging to the "other_vehicle" class are ignored during the creation of our ReID dataset since we do not have label information for them.

Waymo On the Waymo dataset, we train a CenterPoint [52] object detector using the implementation from [7]. Our CenterPoint model is trained for 20 epochs, where each epoch constitutes a pass over 20% of the frames in WOD (sampled uniformly at random). Since the Waymo dataset's Vehicle class is heterogeneous and contains various sub-types of objects that are annotated separately in the nuScenes [2] dataset, we utilize the point-wise semantic segmentation label to automatically divide the Vehicle class into {Car, Truck, Bus, Motorcycle}. Figure 6 shows the proportion of object classes after propagating segmentation labels. Our model's evaluation results on the Waymo validation set with segmentation labels are reported in table 5.

While results between datasets are difficult to compare, we hypothesize that the nuScenes model attains relatively stronger performance. This is to be expected as the model incorporates both camera and LiDAR information, while our Waymo model does not. We note that this performance difference should have an impact on the level of noise in predicted bounding boxes and consequently may impact our ReID network's performance.

A.2. Hyperparameters, training details, and inference speed.

Table 7 reports the hyperparameters of our ReID networks in greater detail. We note that the difference in training epochs (400 vs. 500) for the Waymo and nuScenes datasets was intentional to have approximately the same number of gradient descent steps for each (± 3 epochs).

Parameter	Explanation	Value
Point model hyperparameters		
B	Batch Size	256×4
N	Number of input points	128
D_p	Point input dimension	3
LR	Learning Rate	$3 \cdot 10^{-4}$
WD	weight decay	0.01
G_c	Gradient Clipping	1
Image model hyperparameters		
D_I	Image input dimension	$3 \times 224 \times 224$
B	Batch Size	60×4
LR	Learning Rate	$3 \cdot 10^{-5}$
WD	weight decay	0.01
G_c	Gradient Clipping	1
Randomly initialized Waymo models		
E	Training epochs	500
Randomly initialized nuScenes models		
E	Training epochs	400
Pretrained Models		
E	Training epochs	200

Table 7. **Hyperparameters of our different architectures.** Both image and point models follow a cyclic learning rate schedule with target_ratio $(10, 1e-4)$, cyclic_times 1, and step_ratio_up 0.4 implemented by the software package MMCV [6]

Model	Par.	Batch Size	Inference Time
DeiT-Tiny	5910K	100	32.4 ms $\pm$ 1.75ms
DeiT-Base	87,338K	100	173 ms $\pm$ 11.8ms
Point Transformer	529K	100	15 ms $\pm$ 0.605ms
DGCNN	617K	100	18.7 ms $\pm$ 3.25 ms
PointNet	2800K	100	8.04 ms $\pm$ 0.0996 ms
C2FCN [17]	182.5k	512	92 $\pm$ 7.73 ms
C2FCN no EFA	91.3k	512	6.27 $\pm$ 1.43 ms
RTMM	91.3k	512	13.2 $\pm$ 1.48 ms

Table 8. **Inference speed of different point and image backbones.** All models were tested on a single RTX 3090 GPU.

Table 8 reports the inference speed of all models used in our empirical evaluation. We select batch sizes to simulate the needs of practitioners in a multi-object tracking context. Backbone models are tested for a batch size of 100, an approximate upper bound on the number of detections a 3D object detector might make at a given timestep. We test RTMM with a larger batch size of 512, since pairwise comparisons scale quadratically with the input size. We note, however, that it is possible to considerably reduce the comparisons needed by only comparing within the same class or imposing other such rules. At the batch sizes shown, all our models, backbone + RTMM, run in real-time (< 10 Hz), except for the DeiT-Base model.

B. Additional dataset details

Table 10 reports training set statistics for both our ReID datasets, while table 9 report test set statistics. Algorithms 1 and 2 provide pseudocode for the even and uniform sampling procedures introduces in section 4 and evaluated in table 3. In algorithm 1, for a given object, the frequency distribution over power-two buckets is computed by calculating the number of observations that fall within each bucket and normalizing by the total number of observations of that object, creating a probability distribution over buckets.

Dataset	Positive Pairs	Negative Pairs	Total
nuScenes Eval	53,787	53,733	107,520
Waymo Eval	120,859	120,805	241,664
Waymo Eval All	145,421	145,287	290708

Table 9. **Evaluation set statistics.**

Classes	# of Obj.	# of Obs.	Pos. Pairs	Neg. Pairs
FP bicycle	$--$	$13,100$	$--$	$--$
FP bus	$--$	$2,069$	$--$	$--$
FP car	$--$	$125,223$	$--$	$--$
FP motorcycle	$--$	$7,222$	$--$	$--$
FP pedestrian	$--$	$106,206$	$--$	$--$
FP trailer	$--$	$8,023$	$--$	$--$
FP truck	$--$	$21,792$	$--$	$--$
bicycle	554	$7,575$	$7.76e+4$	$1.62e+12$
bus	422	$8,375$	$1.07e+5$	$4.54e+11$
car	$20,830$	$326,967$	$3.77e+6$	$3.34e+16$
motorcycle	588	$8,201$	$8.67e+4$	$9.73e+11$
pedestrian	$9,112$	$156,852$	$1.83e+6$	$5.43e+15$
trailer	773	$13,921$	$1.66e+5$	$3.34e+12$
truck	$2,933$	$50,487$	$5.98e+5$	$1.32e+14$
Total	$35,212$	$856,013$	$6.63e+06$	$3.90e+16$

Classes	# of Obj.	# of Obs.	Pos. Pairs	Neg. Pairs
FP bicycle	$--$	$27,043$	$--$	$--$
FP bus	$--$	$2,378$	$--$	$--$
FP car	$--$	$24,971$	$--$	$--$
FP motorcycle	$--$	$46,111$	$--$	$--$
FP pedestrian	$--$	$604,021$	$--$	$--$
FP truck	$--$	$5,987$	$--$	$--$
bicycle	497	$46,204$	$3.02e+6$	$1.23e+14$
bus	378	$43,639$	$3.35e+6$	$4.59e+13$
car	$43,305$	$4,002,934$	$2.78e+8$	$3.25e+19$
motorcycle	355	$32,317$	$2.18e+6$	$9.90e+13$
pedestrian	$18,107$	$1,954,970$	$1.41e+8$	$6.40e+18$
truck	$1,114$	$108,722$	$7.77e+6$	$7.13e+14$
Total	$63,756$	$6,899,297$	$4.35e+08$	$3.89e+19$

Table 10. **Training set statistics for nuScenes (left) and WOD (right).** From left to right, the columns contain the number of unique objects of each class, the number of observations of these objects, the number of positive pairs of each class, and the number of negative pairs of each class. False positives are used to create negative pairs with observations of the corresponding true positive class. Positive and negative pairs are created from objects of the same predicted class.

Algorithm 1: Even Data Sampling Algorithm for one Epoch of Training

```
Data: Dataset $D$ of all objects and their observations.
Result: A set of sampled pairs for one epoch of training.
// Initialize sample
$S \leftarrow \emptyset$;
foreach object $O$ in $D$ do
    $o_1 \leftarrow$ random observation of $O$;
    $c \leftarrow$ class of $O$;
    $p_1 \leftarrow$ random number between 0 and 1;
    if $p_1 \leq 0.5$ then
        // Sample positive pair
        $o_2 \leftarrow$ random observation of $O$ other than $o_1$;
    end
    else
        // Get distribution of object O
        $D \leftarrow$ frequency distribution of power-two buckets
            for O's observations;
        $b \leftarrow$ sample a point density bucket from $D$;
        // Sample negative observation
        $p_2 \leftarrow$ random number between 0 and 1;
        if $p_2 \leq 0.5$ then
            // Sample false positive
            $fp_o \leftarrow$
             random false positive of density $b$ with predicted class $c$;
            $S \leftarrow S \cup fp_o$;
        end
        else
            // Sample true positive
            $o_2 \leftarrow$ random observation of density b with class $c$
                other than those associated with $O$;
        end
    end
    $S \leftarrow S \cup (o_1, o_2)$;
end
return $S$
```

Algorithm 2: Uniform Data Sampling Algorithm for one Epoch of Training

```
Data: Dataset $D$ of all objects and their observations.
Result: A set of sampled pairs for one epoch of training.
// Initialize sample
$S \leftarrow \emptyset$;
foreach object $O$ in $D$ do
    $o_1 \leftarrow$ random observation of $O$;
    $c \leftarrow$ class of $O$;
    $p_1 \leftarrow$ random number between 0 and 1;
    if $p_1 \leq 0.5$ then
        // Sample positive pair
        $o_2 \leftarrow$ random observation of $O$ other than $o_1$;
    end
    else
        // Sample negative observation
        $p_2 \leftarrow$ random number between 0 and 1;
        if $p_2 \leq 0.5$ then
            // Sample false positive
            $fp_o \leftarrow$ random false positive of predicted class $c$;
            $S \leftarrow S \cup fp_o$;
        end
        else
            // Sample true positive
            $o_2 \leftarrow$ random observation of class $c$
                other than those associated with $O$;
        end
    end
    $S \leftarrow S \cup (o_1, o_2)$;
end
return $S$
```

C. A reasonable assumption

Section 5.2 analyses the results from table 2 and claims that the performance improvements from nuScenes to Waymo are due in part to the increase in sensor resolution between the datasets. However, Waymo is also much more diverse than nuScenes featuring $63,756$ unique objects vs. $35,212$. This is a potential confounder since a more diverse dataset can also lead to improved performance. We claim, in the main manuscript, that under a reasonable assumption, we can obtain a lower bound on the performance improvement from nuScenes to Waymo that can be attributed to the increase in sensor resolution. The assumption is that the increased diversity from nuScenes to Waymo causes the same performance improvement (in terms of matching accuracy) for image and point cloud models alike. We believe it is reasonable to assume this since we have paired data (i.e., the point clouds and image crops are observations of the same objects). Assuming this, we can then take the performance difference from nuScenes to Waymo of the randomly initialized DeiT-Tiny model to be the increase caused by dataset diversity (as it can only be smaller than or equal to this quantity). Then, assuming no other confounders are present, we can account for the effect of a more diverse dataset by subtracting the performance improvement of the randomly initialized DeiT-Tiny model from the improvement of the point models. This is the number we report in section 5.2.

D. Additional results analysis

While the main manuscript highlights the results that are central to our contributions, some secondary figures could not be included, but still provide additional information. We include them and a corresponding discussion here.

D.1. Scaling compute on nuScenes

Table 11 reports results for scaling compute on nuScenes. Specifically, we train four Point-Transformer models for $500 \cdot 2^i$ epochs with $i \in \{0, 1, 2, 3\}$. All models noticeably improve from longer training, as they do on WOD.

	Backbone	Epochs	Acc.	F1 Pos.	F1 Neg.	Car	Pedestrian	Bicycle	Bus	Motorcycle	Truck	FP
nuScenes Eval	**Point-Transformer**^{L}	4000	77.89%	77.92%	77.85%	82.03%	67.63%	71.82%	84.15%	71.35%	84.77%	83.18%
	Point-Transformer^L	2000	76.91%	77.09%	76.72%	81.02%	66.68%	70.25%	83.01%	71.3%	83.78%	82.21%
	Point-Transformer^L	1000	75.67%	75.92%	75.41%	79.45%	65.79%	69.82%	82.27%	68.97%	82.94%	81.43%
	Point-Transformer^L	500	74.54%	74.72%	74.35%	78.36%	64.39%	67.24%	82.62%	68.08%	82.48%	81.04%

^{L}: using LiDAR data

Table 11. **Scaling compute improves performance for all classes on nuScenes.**

D.2. Additional analysis comparing accuracy at different point densities

Figure 7 is the twin of figure 1 from the main manuscript. Each figure plots the performance of our ReID networks on nuScenes (left) and Waymo (right) at different point densities. The difference between the two figures is seen on the x-axis. Figure 7 plots accuracy on pairs of observations with at least one observation containing x points or more, while figure 1 requires that both observations in the pair have at least x points. We observe that the increase in accuracy is steeper when restricting to cases where both observations have more than x points. However, figure 7 shows the same trend but with a smaller slope and more samples for every x value.

Similarly, figure 8 is the twin of figure 4 in the main manuscript. We see a similar trend of increased performance relative to the number of points but with a smaller slope compared to the plot which filters for both observations having more than x points. Notably, performance improves faster on cars in figure 8 than it does for pedestrians, while the opposite was true in figure 4, suggesting that pairs of dense observations are needed for strong performance on deformable objects, while rigid objects can perform well with only one dense observation.

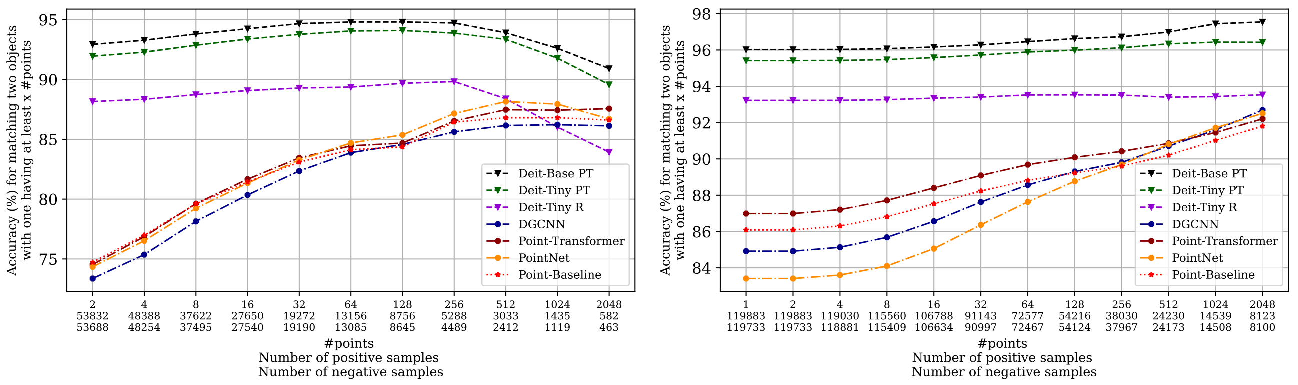


Figure 7. **The performance of point cloud ReID approaches image ReID with sufficient points.** The plot shows the performance of the image and point cloud ReID networks as a function of the number of points in at least one observation of the pair. Left plots models trained on nuScenes and evaluated on the *nuScenes Eval* set, while right plots models trained on Waymo and evaluated on *Waymo Eval*.

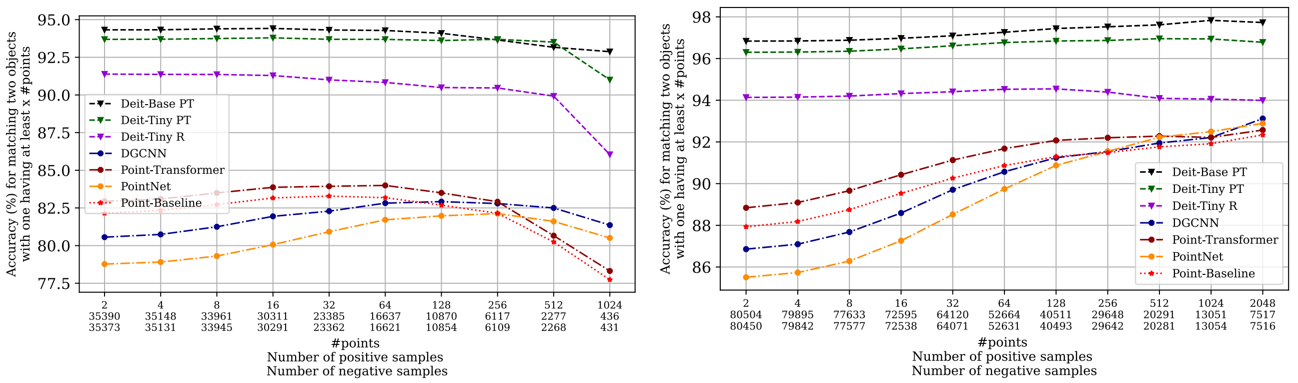


Figure 8. **Deformable vs. Non-deformable objects.** The plot shows the performance of image and point cloud ReID networks for classes car and pedestrian as pairs of observations without at least one observation containing x points are filtered out. We observe a similar trend for deformable and non-deformable objects: both increase with more points. However, we note that the increase in ReID performance for cars is steeper than for pedestrians. Left is evaluated on pedestrians from the *Waymo Eval* set, while right is evaluated on cars from *Waymo Eval*.

D.3. Does point ReID performance plateau at higher point densities?

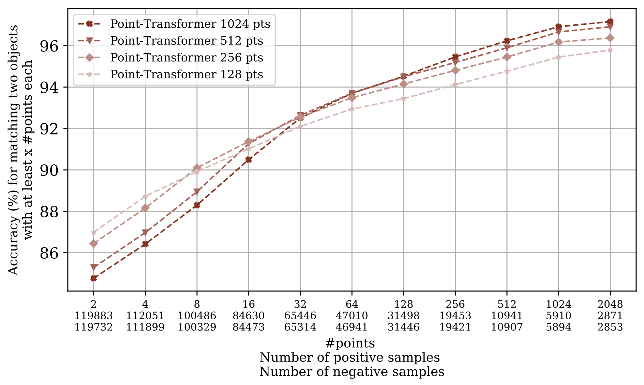
Figure 1 from the main manuscript plots the performance of image and point cloud ReID networks as a function of point density. The plots show that ReID performance improves as point density increases. However, the performance improvements appear to plateau (the slope decreases) as the number of points is increased, suggesting that there may be a ceiling to point

ReID performance. That is, the problem of re-identification from point cloud observations may have large irreducible error. If this were the case, it would imply that there is little performance still to gain from further increasing sensor resolution. However, we believe that this is not the case as there are other explanations for the *plateau* in performance.

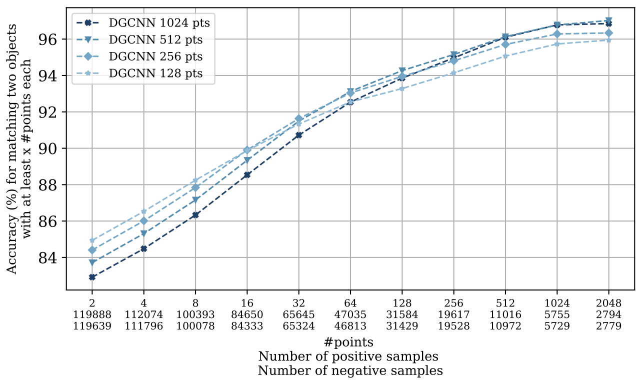
Firstly, we observe for all point models that the *plateau* is more pronounced on the nuScenes dataset than WOD. This is to be expected as the nuScenes datasets is significantly sparser than WOD, meaning that many fewer samples of high point density are seen during training. Therefore, the nuScenes models reach their maximum performance at fewer points than on WOD and have difficulty with the highest point densities as these samples are in the tail of the training distribution.

Secondly, our networks only accept $n = 128$ points as input, subsampling or resampling the input point cloud as needed. This means that the same amount of information is provided to the network for any observation with greater or equal to 128 points. We believe that this along with the network architecture has an influence on the change in slope observed. To verify our hypothesis we conduct experiments training different point models on WOD that accept $n \in \{128, 256, 512, 1024\}$ points. Figure 9 reports the performance of these models on *Waymo Eval* as a function of point density, mirroring figure 1 (right). We do not train models for this experiment on nuScenes as we are concerned with performance at high point density and the nuScenes training set is not dense enough to provide meaningful results. Subplot (a) shows the performance of the Point-Transformer for different numbers of input points. We observe that as the number of input points are increased, the slope improves and higher performance is achieved. Similar trends overall are observed for DGCNN and PointNet (Subplots (b) and (c), respectively), though there is some variance between the three architectures. For instance, Point-Transformer and DGCNN experience poorer performance at lower point densities for models that accept more points, whereas for PointNet the models that accept more points remain strongest throughout. These experiments show that both the architecture and the number of input points affect the slope of curves, confirming our hypothesis.

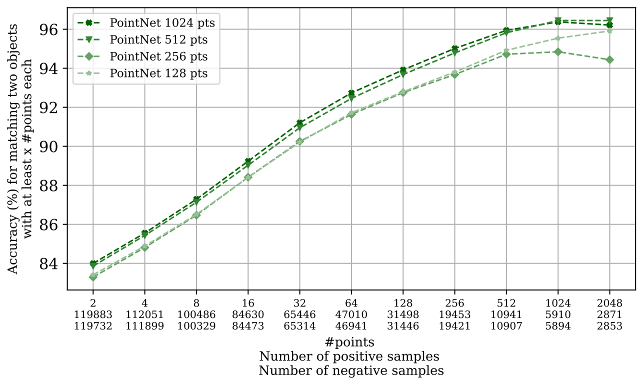
In conclusion, after further investigation, we find that a multitude of factors, including, dataset point density distribution, architecture, and the number of input points, can impact the slope of point cloud ReID models in figure 1. This suggests that appropriately modifying these factors can lead to increased performance at high point density, showing that the results in figure 1 are not indicative of a fundamental ceiling in performance for point ReID.



(a) **Point-Transformer**



(b) **DGCNN**



(c) **PointNet**

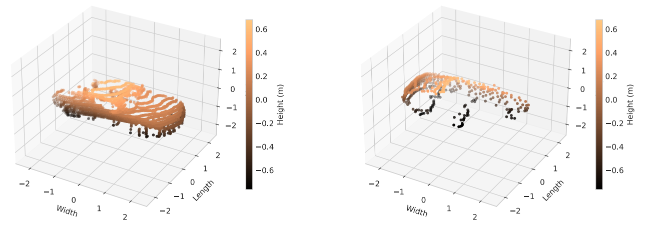
Figure 9. **Increasing the number of input points for various architectures on WOD.** We plot the performance of point cloud ReID networks as a function of point density for networks trained with varying numbers of points. For $n \in \{128, 256, 512, 1024\}$, inputs are subsamples or resampled uniformly at random to match the desired number of points, n.

D.4. A qualitative study of Point ReID success and failure cases

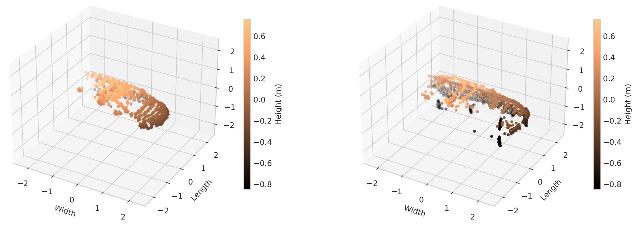
Figures 10, 11, 14, 15, 16, and 17 visualize success and failure cases of our Point-Transformer + RTMM model trained for 400 epochs on WOD. These observations were collected from *Waymo Eval* and were all filtered to contain more than 200 points to make them easier for humans to interpret. For each class, we show a True Positive (TP), True Negative (TN), False Positive (FP), and False Negative(FN). We note that correctly rotated and translated observations, i.e. those without or with minimal bounding box noise, will be centered at $(0, 0, 0)^\top$ and facing the axis labeled "Length". In the following paragraphs, we select classes car and bicycle to analyze in-depth but still provide success and failure cases for other classes in figures 14, 15, 16, and 17.

Class car Subfigure (a) of figure 10 provides an interesting true positive example. Both observations seem to contain points from different parts of the vehicle, yet RTMM is able to infer that they are observations of the same vehicle. This suggests that the model may have some concept of vehicle symmetry, enabling this inference. Subfigure (b) shows a True Negative. This pair seems to have LiDAR points on the same side of the vehicle making the model's predictions easier. The vehicle on the right appears to be larger than the vehicle on the left, which may be how RTMM makes the correct decision. Subfigure (c) shows the first failure case, a false positive. This example is particularly glaring as it is apparent that the vehicle on the left has two elevated pillars which the vehicle on the right does not. It is possible, however, that these were missed when the point cloud is initially subsampled, suggesting that accepting more input points could lead to a more robust model. Subfigure (d) shows a false negative failure case. Although both examples have LiDAR points on the same sections of the vehicle, RTMM makes an incorrect prediction. This could, again, be caused by the uniform subsampling of the input point cloud to $n = 128$ points that can introduce variance into the perceived shapes.

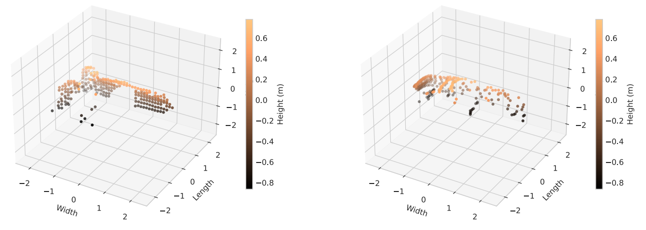
Class bicycle Subfigure (a) of figure 11 shows a true positive pair. Although the rightmost observation lacks LiDAR points on the front wheel of the bicycle and the head of the cyclist, RTMM makes a correct prediction. We hypothesize that this is due to the distinctive shape of the rider's back which is visible in both observations. Subfigure (b) shows a false negative pair. The cyclist in the rightmost observation appears to be wearing a backpack, while the other cyclist is not. This may be how RTMM was capable of making a correct prediction. Subfigure (c) shows the first failure case, a false positive. Although both bicycles are almost completely covered by LiDAR points, RTMM makes a mistake. This may be attributable to the front part of the bicycle being more visible in the right image, while the back is more visible on the left. In such a situation, RTMM may be unable to compare rigid components of the observation. Finally, subfigure (d) shows a false negative. It appears that both observations feature LiDAR points capturing the cyclist from different angles. Given that the object is also deformable, this adds increased difficulty, possibly leading RTMM to incorrectly classify this pair as not matching.



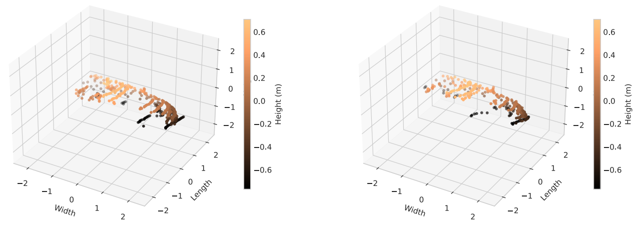
(a) True Positive



(b) True Negative

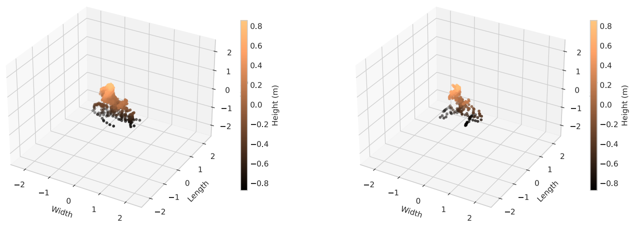


(c) False Positive

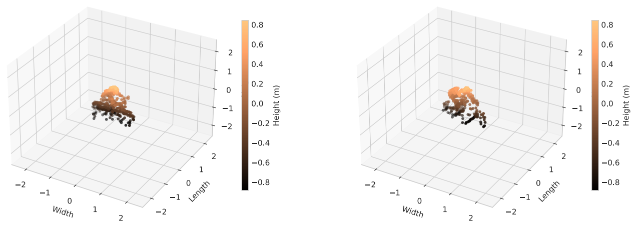


(d) False Negative

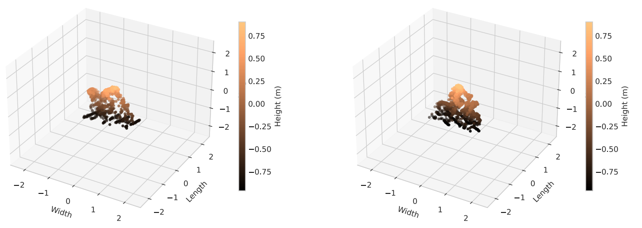
Figure 10. **Qualitative success and failure cases for class car on WOD.** We use the Point-Transformer model trained for 400 epochs. We select pairs of TP, TN, FP, and FN examples that have more than 200 points from *Waymo Eval*.



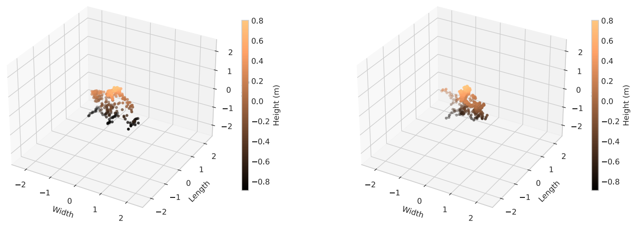
(a) True Positive



(b) True Negative



(c) False Positive



(d) False Negative

Figure 11. **Qualitative success and failure cases for class bicycle on WOD.** We use the Point-Transformer model trained for 400 epochs. We select pairs of TP, TN, FP, and FN examples that have more than 200 points from *Waymo Eval*.

E. Dataset visualization

In figures 12 and 13, we provide visual examples of samples from our Waymo ReID dataset. The images are cropped from projected 3D bounding boxes predicted by our centerpoint model (see sec. A.1 for details). The center plot shows the LiDAR point associated to the same observation. We selected observations with more than 200 points, so that it is possible for a human to make out the underlying object. However, most of the observations contain fewer than 200 points. The leftmost plot shows complete point clouds created by aggregating the points from all observations using ground truth bounding boxes. Aggregated deformable objects (pedestrian, bicycle, and motorcycle) have a blob-like appearance, while rigid objects retain a more detailed shape. This is due to their deformability, causing them to take on many different poses over a sequence.

F. CFA block structure

Our RTMM, illustrated in Fig. 2, compares two point clouds by symmetrically applying CFA blocks [17] between them. These are linear attention blocks, but with a few modifications; here we detail the exact structure used. CFA blocks receive two sets of points $\{\boldsymbol{x}_1^{(i)}\}_i^{n_1}, \{\boldsymbol{x}_2^{(i)}\}_i^{n_2}$, which we designate in stacked matrix form $\boldsymbol{X}_1, \boldsymbol{X}_2 \in \mathbb{R}^{n \times 3}$ henceforth, where the points are subsampled or resampled to size n (we use $n = 128$ for all our models), and their corresponding representations $f_\theta(\boldsymbol{X}_1), f_\theta(\boldsymbol{X}_2)$, where f_θ can be any set or sequence processing network. The block first computes linear cross-attention (LCA):

$$\boldsymbol{L} = \mathrm{LCA}(f_\theta(\boldsymbol{X}_1), f_\theta(\boldsymbol{X}_2) + \boldsymbol{P}, f_\theta(\boldsymbol{X}_2) + \boldsymbol{P}) \tag{5}$$

where $\boldsymbol{P} = \mathrm{MLP}_{pos}(\boldsymbol{X}_2)$ is a positional encoding computed from the point cloud. Next, a layer normalization (LN) is applied followed by an MLP applied to the channel-wise concatenation of $\mathrm{LN}(\boldsymbol{L})$ and $f_\theta(\boldsymbol{X}_1)$) and another layer normalization,

$$\boldsymbol{L}' = \mathrm{LN}(\ \mathrm{MLP}(\ \mathrm{LN}(\boldsymbol{L}) \oplus f_\theta(\boldsymbol{X}_1))). \tag{6}$$

Finally, a residual connection is applied to complete the CFA block:

$$\mathrm{CFA}(f_\theta(\boldsymbol{X}_1), \boldsymbol{X}_1, f_\theta(\boldsymbol{X}_2), \boldsymbol{X}_2) = \boldsymbol{L}' + f_\theta(\boldsymbol{X}_1). \tag{7}$$

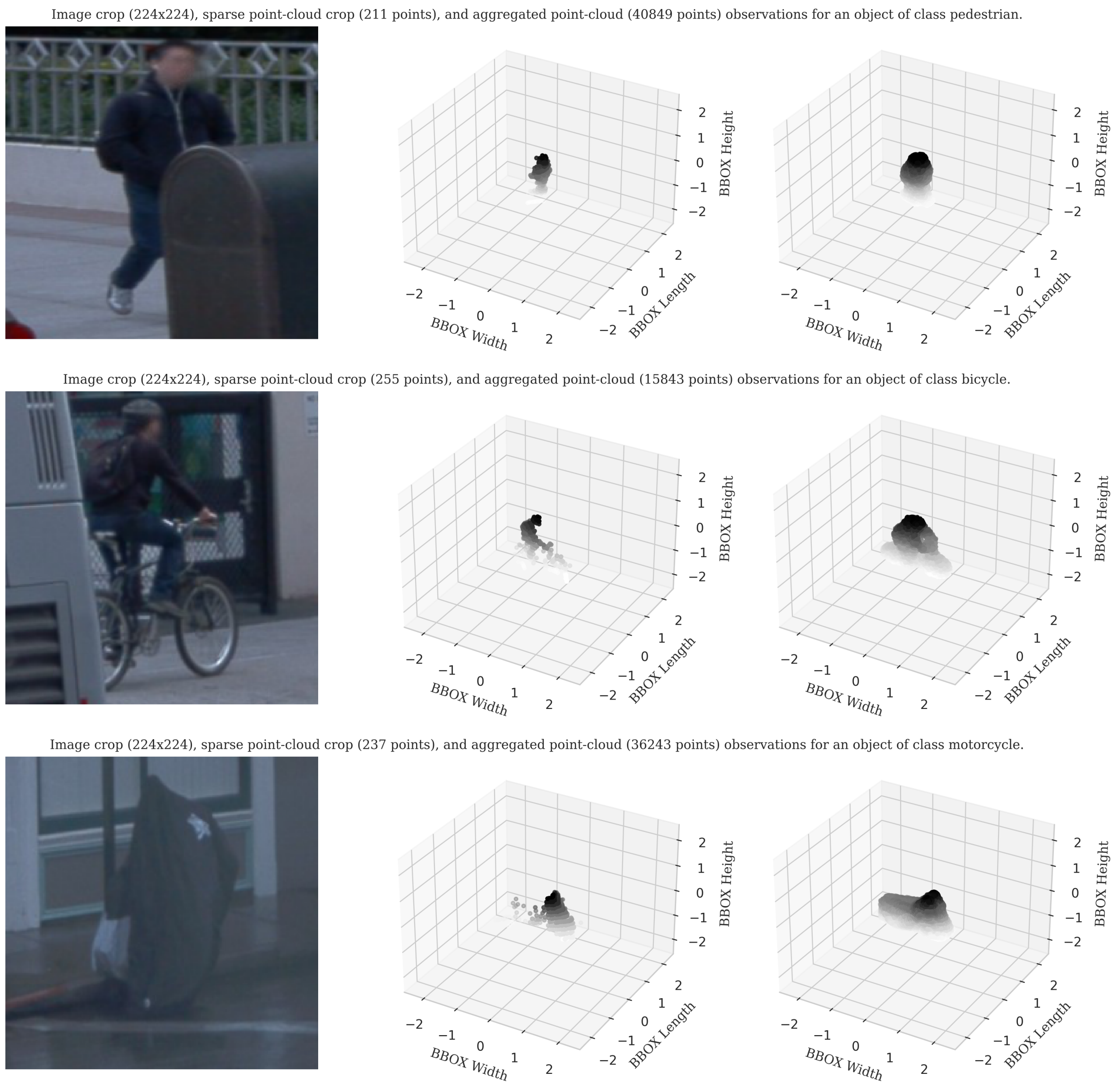


Figure 12. **Samples from our Waymo ReID dataset for deformable objects.** The plot shows the cropped image (left) and cropped sparse point cloud (center) for the same predicted 3D bounding box (output by our centerpoint model). We also include the corresponding complete version of the point cloud (right), created by aggregating LiDAR scans over different observations and mirroring them about the object's center.

Image crop (224x224), sparse point-cloud crop (388 points), and aggregated point-cloud (33860 points) observations for an object of class truck.

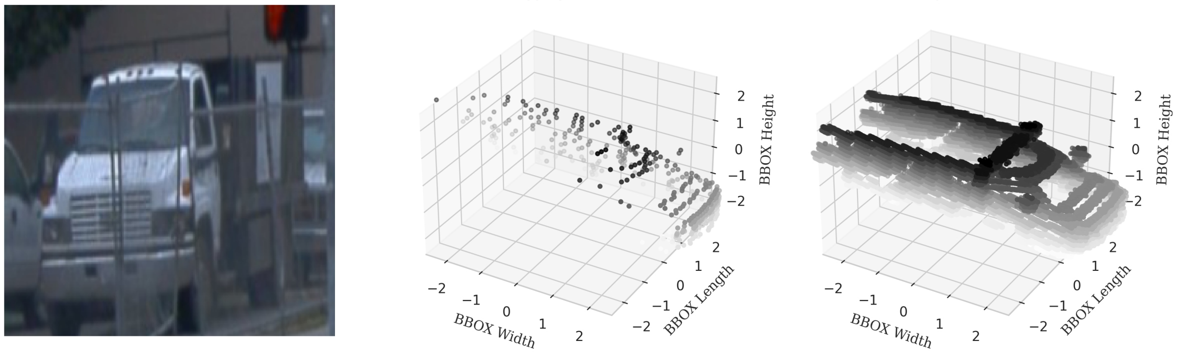


Image crop (224x224), sparse point-cloud crop (201 points), and aggregated point-cloud (18583 points) observations for an object of class car.

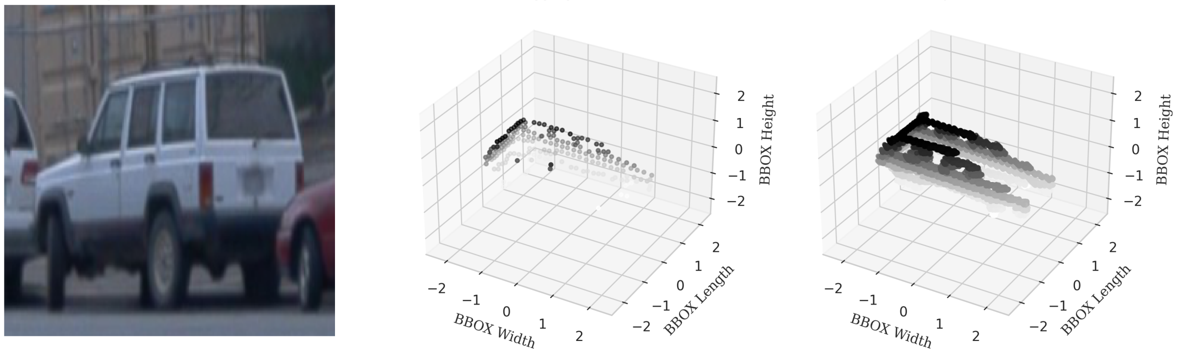


Image crop (224x224), sparse point-cloud crop (11818 points), and aggregated point-cloud (906761 points) observations for an object of class bus.

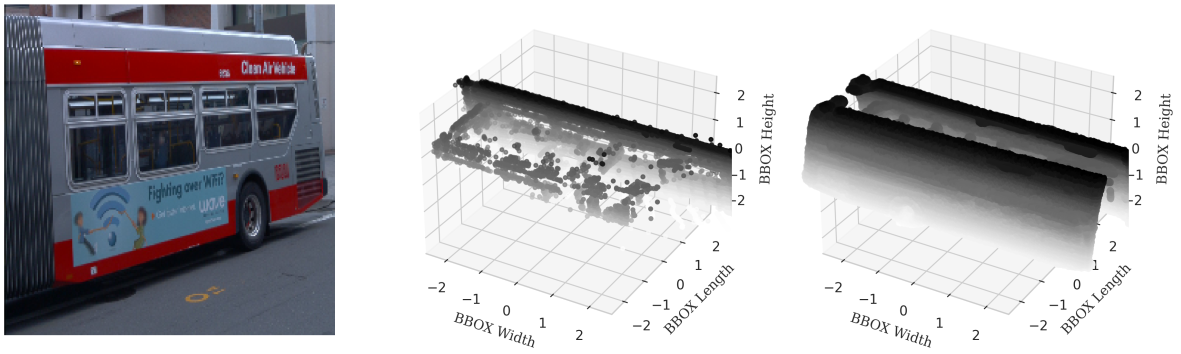
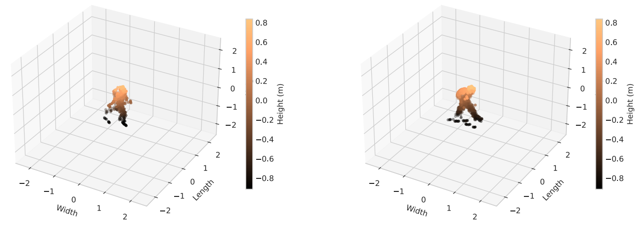
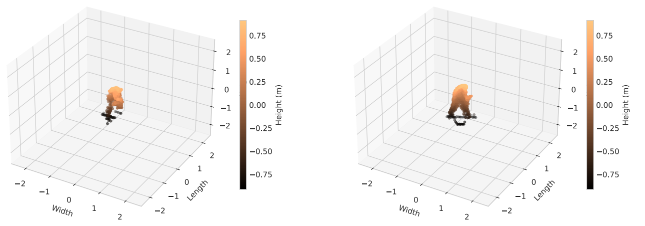


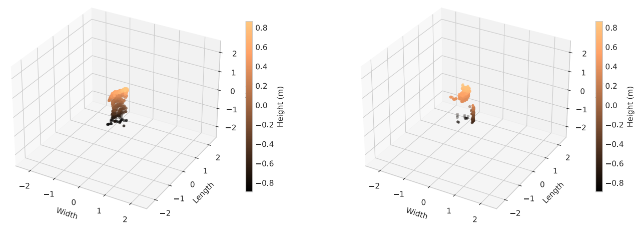
Figure 13. **Samples from our Waymo ReID dataset continued for rigid objects.** The plot shows the cropped image (left) and cropped sparse point cloud (center) for the same predicted 3D bounding box (output by our centerpoint model). We also include the corresponding complete version of the point cloud (right), created by aggregating LiDAR scans over different observations and mirroring them about the object's center.



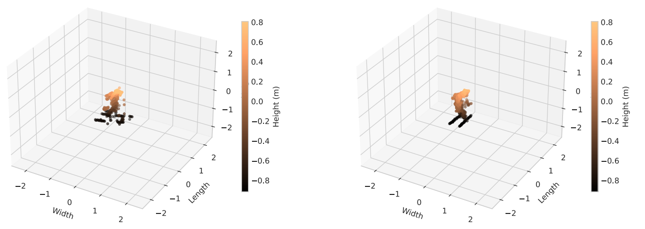
(a) True Positive



(b) True Negative

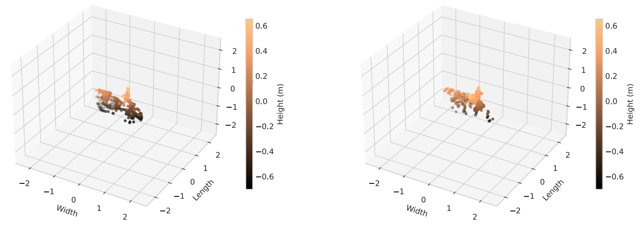


(c) False Positive

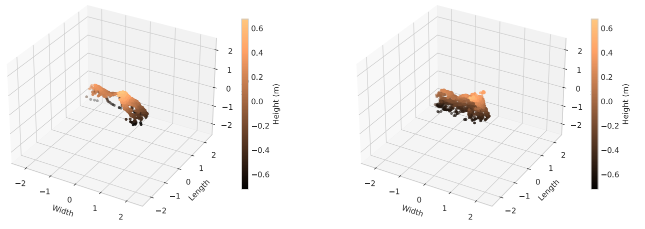


(d) False Negative

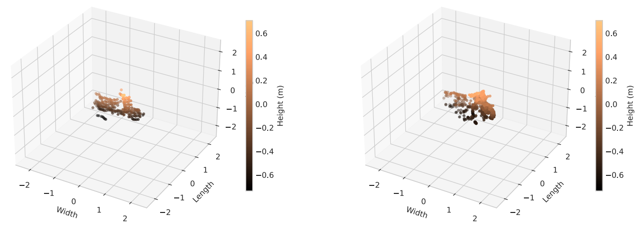
Figure 14. **Qualitative success and failure cases for class pedestrian on WOD.** We use the Point-Transformer model trained for 400 epochs. We select pairs of TP, TN, FP, FN examples that have more than 200 points from *Waymo Eval*.



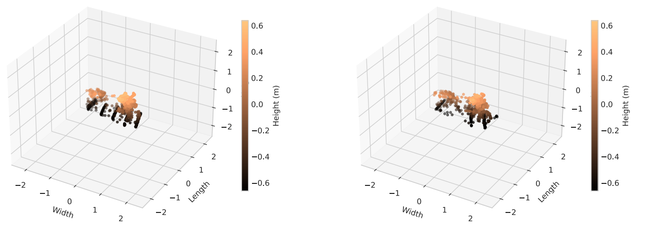
(a) True Positive



(b) True Negative



(c) False Positive



(d) False Negative

Figure 15. **Qualitative success and failure cases for class motorcycle on WOD.** We use the Point-Transformer model trained for 400 epochs. We select pairs of TP, TN, FP, FN examples that have more than 200 points from *Waymo Eval*.

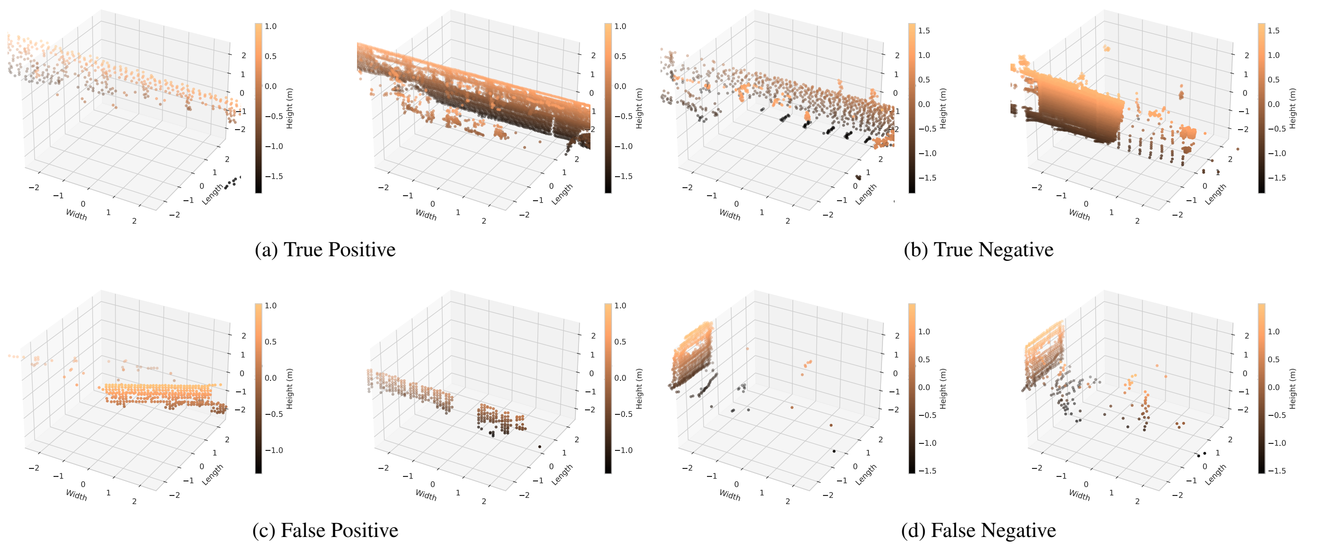


Figure 16. **Qualitative success and failure cases for class bus on WOD.** We use the Point-Transformer model trained for 400 epochs. We select pairs of TP, TN, FP, FN examples that have more than 200 points from *Waymo Eval*.

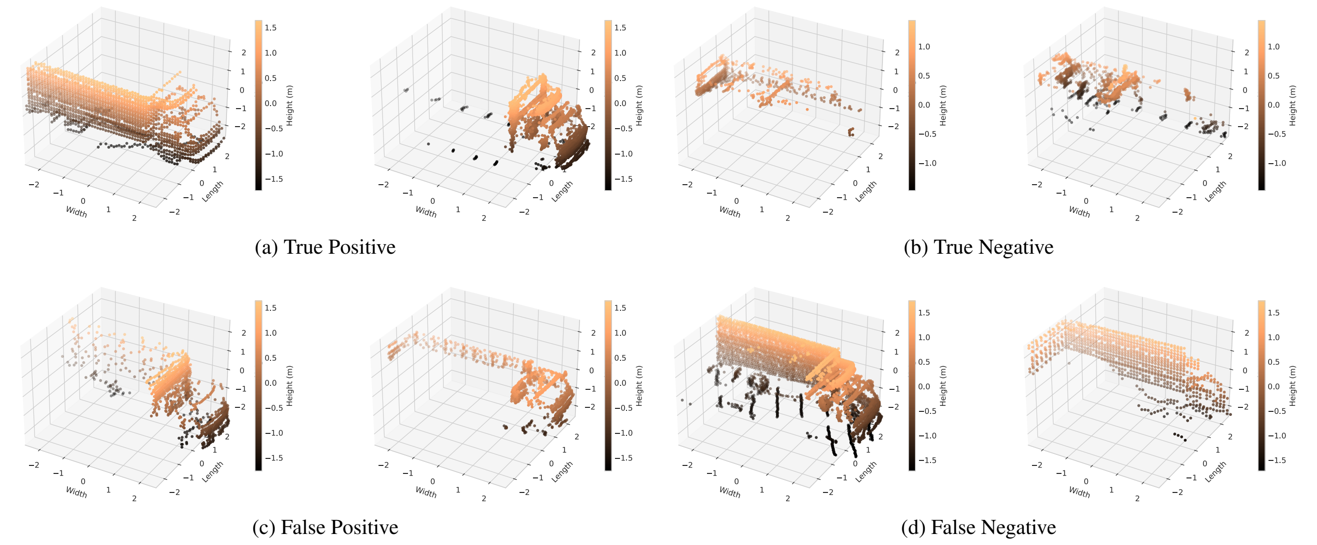


Figure 17. **Qualitative success and failure cases for class truck on WOD.** We use the Point-Transformer model trained for 400 epochs. We select pairs of TP, TN, FP, FN examples that have more than 200 points from *Waymo Eval*.