WiSAR3D - Aerial LiDAR dataset for 3D object detection

Supplementary Material

7. Data statistics

Strips statistics. We present additional statistics on our dataset. The collected strips had a wide range of flight times, from less than a second to over a minute. To balance the dataset in terms of point cloud size and to facilitate the computational requirements for training, we removed strips shorter than 1 second and truncated longer strips to a maximum of 10 seconds. Specifically, we divided strips longer than 10 seconds into equal segments, each between 1 and 10 seconds in length. Fig. 5(a) illustrates the flight times for the strips in both the training and validation sets. The flight times in the training set range from 1.59 to 9.99 seconds, with an average of 6.9 seconds, while in the validation set, the flight times range from 2.36 to 9.98 seconds, with an average of 7.1 seconds. The number of points per strip in both the training and validation sets is shown in Fig. 5(b). In the training set, the number of points ranges from $327K$ to $4845K$, with an average of $2130K$ points. In the validation set, the number of points ranges from $335K$ to $4880K$, with an average of $2226K$ points.

Annotation statistics. We provide some key statistics regarding the annotations. The number of LiDAR points within a box annotation for the *Standing person*, *Tent*, and *Cone* at different sampling rates (PRRs) is shown in Fig. 6. As anticipated, the average number of points for the *Standing person* and *Tent* increases with higher sampling rates. However, for the *Cone*, being a smaller object, the number of points does not increase much beyond a 600kHz sampling rate. The *Tent* consistently contains more points within its box annotations across various rates compared to the *Standing person* and *Cone*. This is expected since tents have a larger and more reflective surface area, resulting in more LiDAR points being reflected back to the sensor.

8. Implementation details

We provide more details on the training procedure for the experiments. As stated in the paper, we trained all the detectors on $8\times$ A100-SXM4-80GB GPUs, for 80 epochs with a batch size of 16 and employed standard augmentations. We used the same augmentations for all the detectors for a fair comparison. These augmentations include random flipping around the X-axis, random scaling drawn from the range $[0.95, 1.05]$, and random rotation around the Z-axis with angles drawn from $[-45°, 45°]$. Additionally, we applied the copy/paste augmentation [31], where GT objects from different training strips are inserted during training.

We used the version of [24] with the center head, and the version of [10] with the center head and dynamic voxels, as they have been shown to outperform their original counterparts [26]. For CenterPoint [32], we modified the 2D BEV backbone to have a single level instead of two, as this configuration demonstrated better results, as shown in Table 6. For PointRCNN [23], a point-based detector, we increased the number of points to better handle our denser point clouds. Table 7 illustrates that increasing the number of points for the input sampling phase, as well as the number of points used in the backbone's hierarchies, improves results, albeit with longer training times and higher memory consumption. For example, using the original number of points proposed for the KITTI dataset yields an mAP accuracy of 0.1117 and takes 1.38 hours to train on 8 GPUs in parallel, with each GPU consuming up to 8.72 GB of memory. However, when we increase the number of input points to $300K$ (from $16K$) and adjust the backbone's hierarchy of points to $[64,000, 8,000, 1,000, 512]$ (from $[4,096, 1,024, 256, 64]$), the mAP increases to 0.4163. This improvement comes at the cost of increased training time, which jumps to 11.63 hours, and each GPU's memory consumption rises to 35.18 GB.

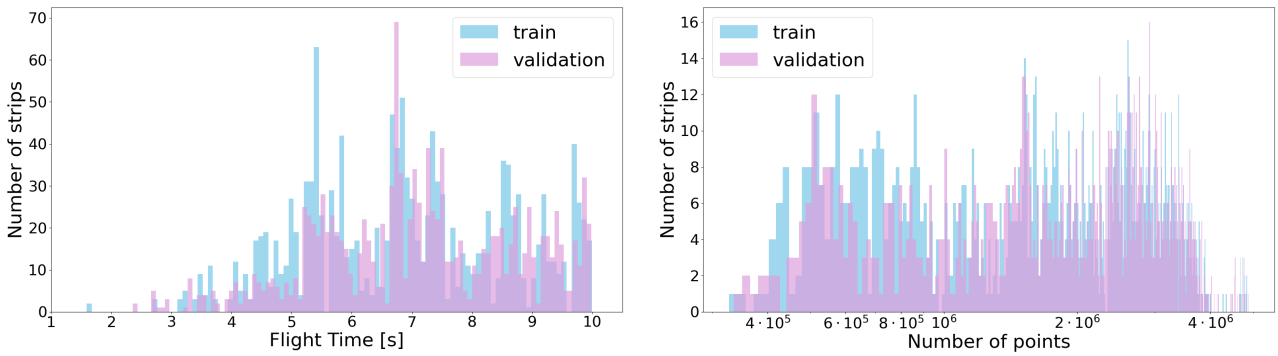
Additionally, for [9, 10, 23, 24, 32, 33] we set the voxel size to $0.1m, 0.1m, 0.15m$ and the grid range to $[-40m, 110m], [-75m, 75m], [-2m, 4m]$ along the X, Y, and Z axes respectively. This results in a 1500×1500 pixel Birds Eye View (BEV) pseudo-image for the voxel-based detectors. For [14] and [29] we set the voxel size to $0.32m, 0.32m, 6.0m$ and $0.32m, 0.32m, 0.1875m$ respectively, both with the respective grid range of $[-40.88m, 108.88m], [-74.88m, 74.88m], [-2m, 4m]$

BEV levels	mAP↑	NDS↑	ATE↓	ASE↓	AOE↓
1	0.6076	0.7146	0.0853	0.1570	0.2928
2	0.5799	0.6902	0.0943	0.1589	0.3454

Table 6. **CenterPoint BEV number of levels.** It is evident that using a single level yields better results compared to the original two levels.

Input #points	Backbone #points	Taken From	mAP↑	NDS↑	Training time (h)	Memory (GB)
16384	[4096, 1024, 256, 64]	[12]	0.1117	0.2233	1.38	8.72
60000	[32000, 4000, 500, 256]	[17]	0.3001	0.4374	2.98	16.12
300000	[64000, 8000, 1000, 512]	ours	0.4163	0.5276	11.63	35.18

Table 7. **PointRCNN number of points.** Increasing the number of points for the sampled input and the backbone hierarchy of PointNet++ [19] improves accuracy, albeit at the cost of longer training times and higher memory usage.



(a) Flight time (b) Number of points

Figure 5. **Strips statistics.** Strips were collected over varying flight durations. The flight time statistics for the strips are presented in (a), and the statistics for the number of points are shown in (b).

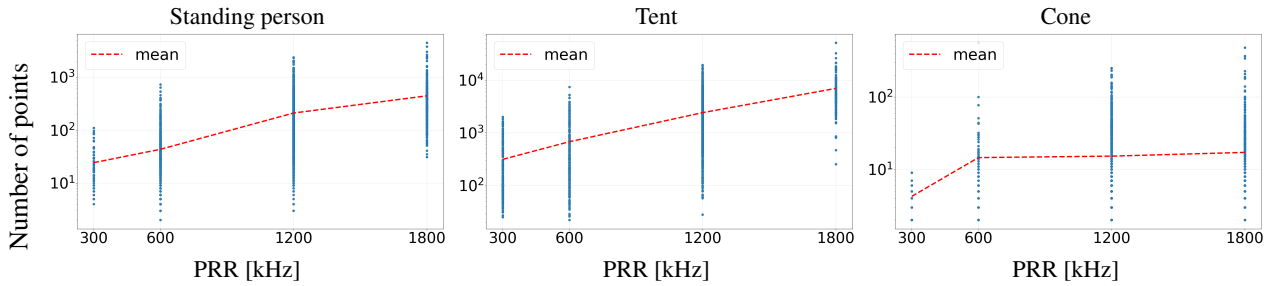


Figure 6. **Annotation statistics.** Number of points within a box annotation for the *Standing person*, *Tent*, and *Cone* at different sampling rates (PRRs).

along the X, Y, and Z axes, which gives a BEV pseudo-image of 468×468 pixels. To accommodate our denser point clouds, we raised the maximum voxel count from $150k$, as employed in Waymo, to $900k$.

9. Results of indoor models

While our dataset is more aligned with outdoor scenes due to the LiDAR sensor, categories, and application, outdoor methods primarily rely on BEV, which can struggle with overlapping targets (e.g., a person under foliage). However, overlapping targets are common in indoor methods. Thus, we also adapt two state-of-the-art indoor methods for our dataset: CAGroup3D [28] and TR3D [21].

Both detectors use RGB features as input. However, our data includes (x, y, z) coordinates and intensity. Thus, to make these models work on our data, we utilized the coordinates and intensity as features for the backbone instead of RGB. The models were trained for 80 epochs with a batch size of 16, using the AdamW optimizer with a learning rate of 0.001 and step decay at epochs 20, 50, and 70.

For TR3D, we used a voxel size of 0.075 along x, y, and z and removed the pooling layer in the backbone, which reduces spatial resolution and improves small-object detection. We also tried a voxel size of 0.1 (as in other models), but accuracy dropped.

Method	mAP	NDS	ATE	ASE	AOE
TR3D	0.6501	0.5765	0.1189	0.3723	1.2876
CAGroup3D	0.1688	0.2323	0.4655	0.6474	1.1371

Table 8. **Additional indoor methods detection result.** Indoor methods do not provide additional benefits to our dataset.

For CAGroup3D, we used a voxel size of 0.15 in the x, y, and z axes. Since the model easily exceeded 80 GB of GPU memory during training, we halved the number of channels in the backbone, reduced the kernel size to 3 in both the dense head classifier and the ROI head grid pooling layers (from 9 and 5, respectively), and used a shared weights classifier for all classes in the dense head.

The results are shown in Table 8. It shows that both models exhibit a high Average Orientation Error (AOE). We believe this is because the models are heavily optimized for indoor datasets like ScanNet, which has axis-aligned 3D boxes, differing from our natural scenes. Additionally, CAGroup3D shows a low mAP, which could be attributed to the model being designed for a smaller spatial range with significantly fewer points per scene compared to our dataset.