

Multi-Modality Task Cascade for 3D Object Detection (Supplementary Materials)

Jinhyung Park
jinhyun1@andrew.cmu.edu
Xinshuo Weng
xinshuow@cs.cmu.edu
Yunze Man
yman@cs.cmu.edu
Kris Kitani
kkitani@cs.cmu.edu

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA

1 Overview

In this supplementary, we provide more details and results that are organized as follows:

- Section 2 provides more details and visualizations on our 2D segmentation ground truth generation, as well as comparison between training our model on the provided SUN RGB-D 2D segmentation ground truth verses the 2D segmentation ground truth generated from 3D box annotations.
- Section 3 explains our implementation and training details in more depth. Further, we present training and inference times for our pipeline.
- Section 4 presents more experiments building off of the 2D→3D→2D→3D framework, but with a stronger initial 2D segmentation network.
- Section 5 contains per-category results for 3D detection.
- Section 6 contains additional visualizations.

2 2D Ground Truth

2.1 2D Ground Truth Generation From 3D Box Annotations

To generate 2D semantic segmentation ground truth to train our 2D modules, we elect to generate them from the 3D object detection labels instead of using the 2D segmentation labels provided with the SUN RGB-D dataset. This choice enables our method to impose no additional annotation burdens and allows MTC-RCNN to be used in scenarios where 2D segmentation labels are unavailable.

To obtain the 2D labels, we expand the 2D depth map into a 3D point cloud using the camera intrinsics. Then, using the 3D box labels, if a point is within a single 3D box, the

2D Labels Used	1st Stage 2D mIoU	2nd Stage 2D mIoU	3D mAP
Generated from 3D boxes	46.37	52.65	31.62
Human Annotations	49.61	54.22	31.17

Table 1: Comparison between using generated 2D segmentation labels vs human-annotated 2D segmentation labels from SUN RGB-D

Bathtub	Bed	Bookshelf	Chair	Desk	Dresser	Nightstand	Sofa	Table	Toilet	2D mIoU
62.93	72.5	38.62	55.39	33.39	47.91	28.28	58.00	38.10	71.06	50.62

Table 2: 2D mIoU between generated 2D segmentation labels and human-annotated 2D segmentation labels.

point (and its corresponding 2D pixel) is marked as the class of the 3D box it is in. If a point is not within any box, it is labeled as background. We note that some 3D points are in multiple boxes and that some 2D pixels do not have a corresponding depth value. For these latter two cases, the 2D pixel is marked to be ignored during training.

We present some example visualizations of the 2D labels in Figure 1. Background is shown in black and ignored pixels are shown in white. We see that for some scenes (the first three rows), the segmentation labels are quite accurate, with objects clearly labeled. However, in the last three rows, we see that the generated ground truth is not always perfect. In row 4, the floor is very reflective, causing many portions of the floor to not have corresponding depth values. In row 5, the chair and table 3D bounding boxes are greatly overlapped, causing many areas to be ignored during training. In the final row we see both problems together - much of the scene is marked white. Despite these incomplete 2D segmentation labels, we find that this is enough to well-supervise MTC-RCNN, which is able to generate high-quality 2D segmentation predictions to further benefit 3D object detection.

2.2 Additional Experiments Using SUN RGB-D 2D Human-Annotated Segmentation Ground Truth

The SUN RGB-D dataset additionally provides 2D segmentation ground truth annotations. Although our model does not require usage of these additional annotations for training, we present here the results of training our 2D→3D→2D→3D pipeline using these labels to compare with training on our generated 2D ground truth. The results are shown in Table 1. Please note that the 2D mIoUs are not directly comparable, as the segmentation predictions trained on different 2D labels are evaluated against their respective 2D label sources - the first row 2D predictions are evaluated with respect to the 2D labels generated from 3D boxes, and the second row predictions are evaluated against the SUN RGB-D 2D human annotations. However, the 3D mAP is directly comparable.

We notice that regardless of which type of 2D labels are used, our pipeline demonstrates a huge boost in 2D mIoU performance: +6.28 mIoU for row 1, +4.61 mIoU for row, showing that the 3D to 2D fusion is greatly beneficial.

However, curiously, we notice that the 3D mAP performance is worse when we use SUN RGB-D’s human-annotated 2D segmentation labels. To investigate why this may be the case, we computed the 2D mIoU between 2D segmentation labels generated from 3D box annotations vs the 2D human-annotated segmentation labels as shown in Table 2. Note that

we only computed mIoU at 2D locations where there was a valid depth value and the human 2D segmentation labels are not marked as "ignore." Surprisingly, we see that the mIoU is very low - for some classes, like nighstand, the mIoU is as low as 28.28. Looking at the visualizations in Figure 2, we find that this is due to a multitude of factors. First, we see that similar classes (table, desk, and nightstand) are often mislabeled as each other. In the first row, the object to the right of the bed is labeled as a table in the provided 2D segmentation labels but is labeled as a night stand in the 3D box annotations - this happens in row 3 as well. Another trend seems to be missing labels in either 2D or 3D - in the first second row, a chair is labeled in 2D but not labeled in 3D. Conversely, several dressers in the third row are labeled in 3D but not in 2D. From these visualizations as well as our results, we find that our method 1) is able to cope well with missing labels, 2) works better when 3D and 2D annotations are consistent, even if the 2D annotations are more sparse like they are for the 2D labels generated from 3D box annotations, 3) but still demonstrates significant performance improvement in both 2D and 3D even when 2D and 3D labels are not perfectly matched. As such, we further verify that our method is robust and able to work well without additional 2D segmentation annotations.

3 Implementation Details and Runtime

3.1 Implementation Details

Data Setup and Augmentation. Following the commonly used data processing strategy from [14], 20,000 points from each point cloud is sampled to be used as input to the network. Each point has XYZ coordinates as well as a height value (distance from the ground). The ground height is estimated from the 1% lowest percentile of heights of points. During training, each point cloud is augmented as follows: random flip, random rotation between $[-30, 30]$ degrees, and random scaling between $[.85, 1.15]$. For the 2D image, the augmentations are as follows: random resizing of image with smaller side between $[480, 600]$, photometric distortion, ImageNet normalization of RGB channels, and depth channel normalized to between 0 and 1. Consistent with previous works, no test-time augmentation is done. To account for the randomness in sampling 20,000 points, each model is evaluated 5 times on different seeds, with the results averaged.

Training Details. For training, We use the AdamW optimizer ($\beta_1=0.9$, $\beta_2=0.999$) with 240 epochs. The initial learning rate is $5e-4$ with a batch size of 8, and is decayed 10x at 160 and 210 epochs. Weight decay is set as 0.01 for 3D components and $1e-4$ for 2D components. Gradient normalized clipping is used, with maximum norm of 10.

We use dropout [14] at multiple places in our pipeline. Dropout of rate 0.1 is used before the final classification heads in the 2D model as well as before each box parameter prediction head in the second stage 3D model. Further, dropout of rate 0.5 is used when fusing 2D segmentation predictions into 3D modules, at both $2D \rightarrow 3D$ junctures in the full $2D \rightarrow 3D \rightarrow 2D \rightarrow 3D$ pipeline. Notably, instead of randomly dropping out individual scalars, we randomly dropout 2D segmentation predictions for entire samples. More specifically, for each batch, the 2D segmentation predictions for half of the samples are not fused into 3D. We find that including this dropout allows 3D modules to not over-rely on 2D segmentation predictions, which allow them to be more robust to mistakes in 2D segmentation.

Method	Throughput (ms/frame)	3D AP@0.25	3D AP@0.50	3D mAP (AP _{.25:.95})
VoteNet [10]	48 <i>ms</i>	58.7	35.1	23.8
H3DNet [14]	185 <i>ms</i>	61.1	39.0	-
Group-Free [8]	90 <i>ms</i>	63.0	45.2	-
Ours (3D→3D)	132 <i>ms</i>	60.2	46.0	29.8
Ours (3D→2D→3D)	156 <i>ms</i>	64.6	49.0	31.8
Ours (2D→3D→2D→3D)	167 <i>ms</i>	65.0	48.4	32.0

Table 3: Inference time of different methods.

3.2 Runtime Experiments

The inference time of our method in comparison with other methods is shown in Table 3. We acquiesce that although our method demonstrates better performance than previous works, it is slower than some. However, we believe our method cannot simply be considered a 3D object detection pipeline as it offers additional significant advantages.

First, our (3D→2D→3D) and (2D→3D→2D→3D) frameworks produce high-quality 2D segmentation outputs which are significantly better than single-modality 2D segmentation baselines. 2D segmentation detections are often a critical intermediate output for robotic manipulation or autonomous driving. For instance, segmentation is commonly used for free space estimation [8, 14], and some methods propose learning control policies using semantic segmentation predictions as an intermediary representation [8, 9, 8].

Second, what we propose in this work is not just a specific architecture but rather a flexible framework for cascaded fusion of 2D and 3D modalities. For instance, the first stage 3D network does not need to be VoteNet, nor does the second stage 3D network need to be fashioned from LiDAR-RCNN [9]. Any initial 3D network can general proposals, and any 3D refinement head will work with our proposed idea. In fact, looking at Table 3, we find that the majority of the runtime of our method is actually due to the choice of LiDAR-RCNN as the second stage 3D network (48ms to 132ms), while the addition of the 2D modules as well as the proposed multi-modal fusion is actually very lightweight. ResNet18 DeepLabV3+ just takes 18ms per frame on its own, which means the multi-modal fusion in (3D→2D→3D) just takes $156\text{ ms} - (132\text{ ms} + 18\text{ ms}) = 6\text{ ms}$. Pipelines for autonomous driving or robotic control with existing 2D and 3D modules can flexibly fit their networks into our pipeline for multi-modal improvement of their predictions. We show that improving the 2D network can yield benefits in both 2D and 3D predictions in Section 4.

Third, the recursive nature of our method, as well as the usage of dropout at multiple stages in our pipeline allows our method to be repeated fewer or more times as desired. For instance, one could decide to just take the first stage 3D predictions or the second stage 2D predictions as the final output and save runtime on lower resource environments. On the other hand, on stronger hardware, our pipeline can be recursive applied for further boost in performance. This flexibility allows our pipeline to be applied to a larger range of scenarios.

Finally, the runtime of our method can further be improved through improvements and optimization for edge devices. First, we find that simply replacing the multi-class NMS of the proposals to class-agnostic NMS can improve the runtime of our full pipeline from 167 *ms* to 125 *ms* while the 3D mAP performance is largely maintained (31.62 → 31.52, averaged over evaluation runs). The performance can be further optimized on edge devices using TensorRT, a deep-learning inference optimizer.

First 2D Backbone	Method	2D mIoU	3D mAP
-	Ours (3D→2D→3D)	51.03	31.46
ResNet18	Initial 2D-only predictions	46.37	-
	Ours (2D→3D→2D→3D)	52.65	31.62
	Ours (2D→3D→2D→3D→2D→3D)	52.93	31.79
	Ours (2D→3D→2D→3D→2D→3D→2D→3D)	52.91	31.80
ResNet50	Initial 2D-only predictions	50.22	-
	Ours (2D→3D→2D→3D)	53.38	32.05
	Ours (2D→3D→2D→3D→2D→3D)	53.49	32.23
	Ours (2D→3D→2D→3D→2D→3D→2D→3D)	53.34	32.19
ResNet101	Initial 2D-only predictions	52.92	-
	Ours (2D→3D→2D→3D)	53.60	32.39
	Ours (2D→3D→2D→3D→2D→3D)	53.53	32.45
	Ours (2D→3D→2D→3D→2D→3D→2D→3D)	53.38	32.39

Table 4: Effects of incorporating PointPainting into our framework with larger initial 2D backbones during inference.

4 2D→3D→2D→3D with Larger Initial 2D Backbone

In Table 4, we experiment with different backbones for the initial 2D network (that is fused into the 3D proposal generation stage) during inference. During training, 2D predictions from a baseline 2D-only ResNet18 DeepLabV3+ is used. We observe that our pipeline is able to improve upon 2D predictions of even a very large backbone like ResNet101 - the initial 2D predictions achieve 52.92 2D mIoU, and the second 2D module is able to improve it significantly to 53.60, despite the second 2D module having a much smaller ResNet18 backbone. Further, we also see improvements in both 2D mIoU and 3D mAP with larger initial 2D backbone. We also experiment with recursively applying our pipeline one and two additional times (3rd and 4th row in each section) and note that although the first recursive application often yields benefits, the gains saturate or decline with the second additional application.

5 More Quantitative Results

In Tables 5, 6, 7, 8, we report per-category evaluation results. The commonly used AP@0.25 metric is quite saturated by recent works, so we also report AP@0.50, AP@0.75, and mAP (AP_{.25:.95}). mAP is a tougher and more stable metric incorporating many IoU thresholds. We observe that our 3D-only (3D→3D) model is already a very strong state-of-the-art method. Further, including additional 2D segmentation between the two 3D modules boosts performance by a significant margin. Then, also including a 2D network before the first 3D proposal generation stage further boosts performance (AP@0.25, AP@0.75, mAP). Finally, recursively applying our pipeline once more demonstrates a small but consistent additional boost in all metrics.

Methods	Input	Bathtub	Bed	Bookshelf	Chair	Desk	Dresser	Nightstand	Sofa	Table	Toilet	AP@0.25
F-PointNet [10]	point+RGB	43.3	81.1	33.3	64.2	24.7	32.0	58.1	61.1	51.1	90.9	54.0
VoteNet [10]	point	74.4	73.0	28.8	75.3	22.0	29.8	62.2	64.0	47.3	90.1	57.7
VoteNet [10]*	point	74.1	85.8	34.3	75.6	26.0	28.3	60.6	66.7	50.1	89.6	58.7
ImVoteNet [10]	point+RGB	75.9	87.6	41.3	76.7	28.7	41.4	69.9	70.7	51.1	90.5	63.4
H3DNet [10] [†]	point	73.8	85.6	31.0	76.7	29.6	33.4	65.5	66.5	50.8	88.2	60.1
BRNet [10]	point	76.2	86.9	29.7	77.4	29.6	35.9	65.9	66.4	51.8	91.3	61.1
Group-Free [10]	point	80.0	87.8	32.5	79.4	32.6	37.0	66.7	70.0	53.8	91.1	63.0
Ours (3D→3D)	point	76.7	83.9	25.8	77.5	25.5	31.1	67.0	69.2	54.9	90.9	60.2
Ours (3D→2D→3D)	point+RGB	79.7	85.9	43.9	78.3	28.3	45.2	69.2	68.4	55.3	92.2	64.6
Ours (2D→3D→2D→3D)	point+RGB	77.0	86.2	47.5	79.5	29.2	47.5	68.2	67.6	54.5	92.6	65.0
Ours (2D→3D) × 3	point+RGB	78.8	86.3	46.3	79.7	29.7	47.2	69.5	68.2	54.4	92.8	65.3

Table 5: 3D object detection results on SUN RGB-D. We present per-category average precision (AP) at the **0.25** IoU threshold. *We report 5-times evaluation results on the checkpoint from MMdetection3D[10] which has higher results than the official paper. [†] H3DNet uses 4 PointNet++ backbones.

Methods	Input	Bathtub	Bed	Bookshelf	Chair	Desk	Dresser	Nightstand	Sofa	Table	Toilet	AP@0.50
VoteNet [10]	point	49.9	47.3	4.6	54.1	5.2	13.6	35.0	41.4	19.7	58.6	32.9
VoteNet [10]*	point	43.0	54.2	7.3	54.7	6.0	13.1	39.4	49.9	21.6	62.1	35.1
H3DNet [10] [†]	point	47.6	52.9	8.6	60.1	8.4	20.6	45.6	50.4	27.1	69.1	39.0
BRNet [10]	point	55.5	63.8	9.3	61.6	10.0	27.3	53.2	56.7	28.6	70.9	43.7
SparsePoint [10]	point	60.9	63.2	13.8	61.2	14.2	23.7	49.1	57.7	33.2	65.4	44.2
Group-Free [10]	point	64.0	67.1	12.4	62.6	14.5	21.9	49.8	58.2	29.2	72.2	45.2
Ours (3D→3D)	point	63.5	64.8	8.9	64.3	10.8	22.7	56.9	58.6	32.0	79.7	46.0
Ours (3D→2D→3D)	point+RGB	64.9	67.0	20.0	65.8	11.4	33.9	57.6	58.1	34.2	76.9	49.0
Ours (2D→3D→2D→3D)	point+RGB	54.6	66.2	23.2	67.0	12.8	32.6	54.6	58.6	34.3	80.2	48.4
Ours (2D→3D) × 3	point+RGB	56.1	67.2	22.5	67.3	12.7	32.3	55.5	59.1	34.4	78.9	48.6

Table 6: 3D object detection results on SUN RGB-D. We present per-category average precision (AP) at the **0.50** IoU threshold. *We report 5-times evaluation results on the checkpoint from MMdetection3D[10] which has higher results than the official paper. [†] H3DNet uses 4 PointNet++ backbones.

Methods	Input	Bathtub	Bed	Bookshelf	Chair	Desk	Dresser	Nightstand	Sofa	Table	Toilet	AP@0.75
VoteNet [10]*	point	0.8	4.0	0.0	2.5	0.0	0.1	0.2	4.3	0.5	2.6	1.5
BRNet [10]	point	-	-	-	-	-	-	-	-	-	-	5.3
Ours (3D→3D)	point	5.1	18.2	0.1	8.1	0.4	1.4	2.0	15.3	2.3	10.8	6.4
Ours (3D→2D→3D)	point+RGB	4.1	25.1	0.4	8.8	0.5	3.5	3.7	16.8	2.5	12.0	7.7
Ours (2D→3D→2D→3D)	point+RGB	4.0	22.8	0.3	9.1	1.1	2.6	4.5	17.2	3.8	16.6	8.2
Ours (2D→3D) × 3	point+RGB	4.6	23.0	0.5	9.2	1.1	2.9	4.7	18.2	3.8	16.4	8.4

Table 7: 3D object detection results on SUN RGB-D. We present per-category average precision (AP) at the **0.75** IoU threshold. *We report 5-times evaluation results on the checkpoint from MMdetection3D[10] which has higher results than the official paper.

Methods	Input	Bathtub	Bed	Bookshelf	Chair	Desk	Dresser	Nightstand	Sofa	Table	Toilet	mAP (AP _{0.25:0.95})
VoteNet [10]*	point	27.0	36.7	9.4	34.2	6.7	10.0	24.9	32.0	17.1	40.3	23.8
Ours (3D→3D)	point	38.3	44.0	8.8	40.4	8.5	13.7	33.2	39.1	22.8	49.3	29.8
Ours (3D→2D→3D)	point+RGB	40.0	46.4	14.5	41.1	9.3	21.4	34.5	38.7	23.4	49.1	31.8
Ours (2D→3D→2D→3D)	point+RGB	36.5	45.7	16.9	41.8	10.1	21.1	33.7	38.8	23.8	51.1	32.0
Ours (2D→3D) × 3	point+RGB	37.6	46.1	16.6	42.0	10.1	20.9	34.7	39.3	23.9	50.8	32.2

Table 8: 3D object detection results on SUN RGB-D. We present per-category mean average precision (mAP), averaged over APs from IoUs from 0.25 to 0.95 at 0.05 intervals. *We report 5-times evaluation results on the checkpoint from MMdetection3D[10] which has higher results than the official paper.

6 More Qualitative Results

We show additional 3D detection results from our pipeline in Figure 3. We find that our method is able to produce highly accurate predictions, often capturing objects not labeled in ground truth.

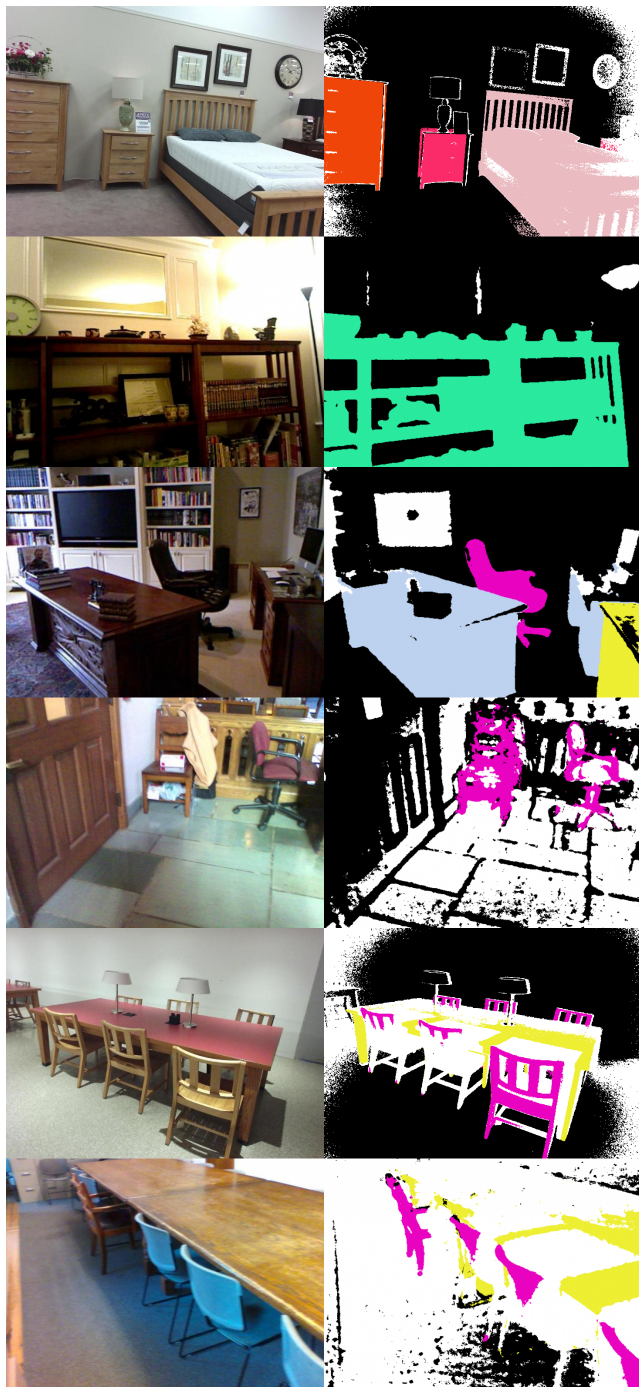


Figure 1: Visualization of generated 2D segmentation ground truth.

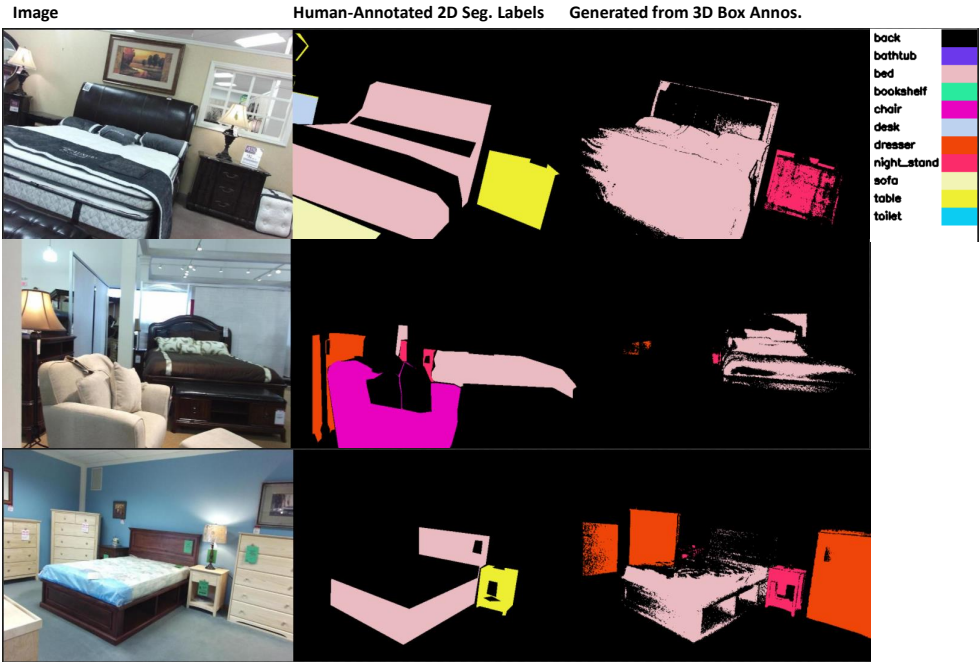


Figure 2: Visualization of 2D Segmentation Labels.

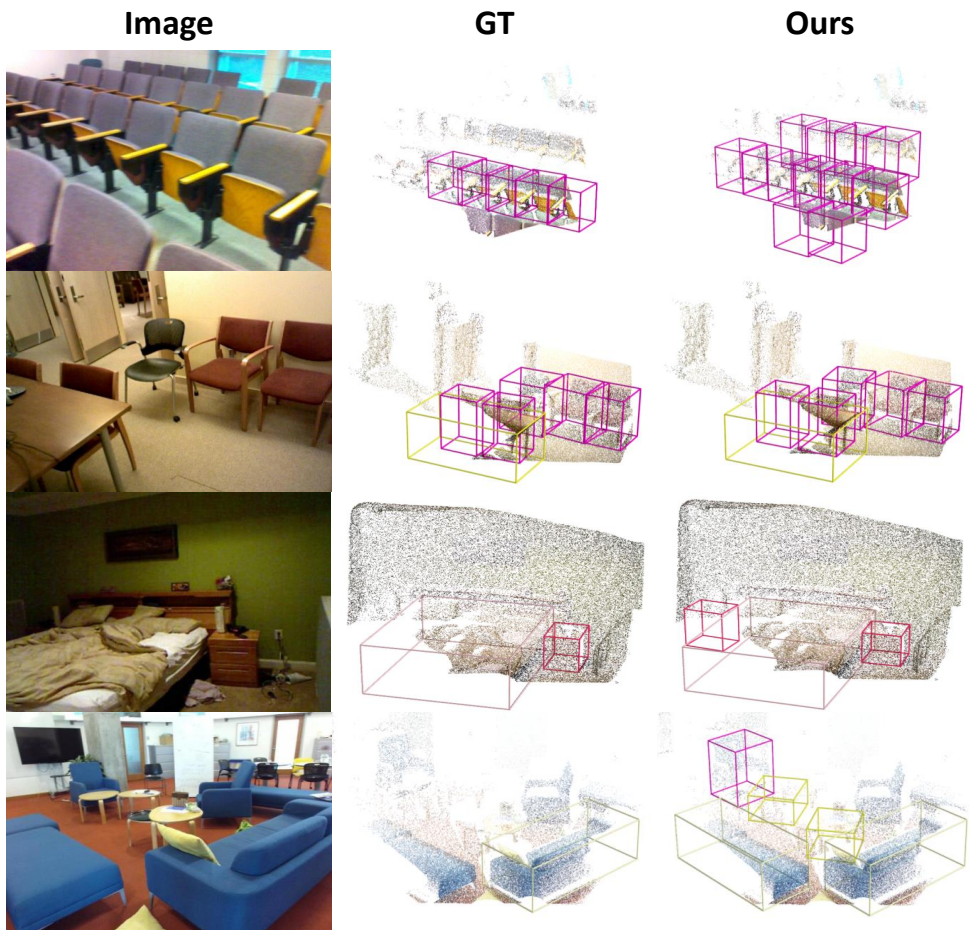


Figure 3: Additional qualitative results from our model.

References

- [1] Bowen Cheng, Lu Sheng, Shaoshuai Shi, Ming Yang, and Dong Xu. Back-tracing representative points for voting-based 3d object detection in point clouds. *ArXiv*, abs/2104.06114, 2021.
- [2] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. <https://github.com/open-mmlab/mmdetection3d>, 2020.
- [3] Zhang-Wei Hong, Yuming Chen, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, Hsuan-Kung Yang, Brian Hsi-Lin Ho, Chih-Chieh Tu, Yueh-Chuan Chang, Tsu-Ching Hsiao, Hsin-Wei Hsiao, S. Lai, and Chun-Yi Lee. Virtual-to-real: Learning to control in visual semantic segmentation. In *IJCAI*, 2018.
- [4] Zhichao Li, Feng Wang, and Naiyan Wang. Lidar r-cnn: An efficient and universal 3d object detector. *CVPR*, 2021.
- [5] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. *ArXiv*, abs/2104.00678, 2021.
- [6] Zili Liu, Guodong Xu, Honghui Yang, Haifeng Liu, and Deng Cai. Sparsepoint: Fully end-to-end sparse 3d object detector. *ArXiv*, abs/2103.10042, 2021.
- [7] Arsalan Mousavian, A. Toshev, Marek Fiser, J. Kosecka, and James Davidson. Visual representations for semantic target driven navigation. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8846–8852, 2019.
- [8] Matthias Müller, A. Dosovitskiy, Bernard Ghanem, and V. Koltun. Driving policy transfer via modularity and abstraction. In *CoRL*, 2018.
- [9] Suvam Patra, Pranjal Maheshwari, Shashank Yadav, C. Arora, and Subhashis Banerjee. A joint 3d-2d based method for free space detection on roads. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 643–652, 2018.
- [10] C. Qi, W. Liu, Chenxia Wu, Hao Su, and L. Guibas. Frustum pointnets for 3d object detection from rgb-d data. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.
- [11] C. Qi, O. Litany, Kaiming He, and L. Guibas. Deep hough voting for 3d object detection in point clouds. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9276–9285, 2019.
- [12] C. Qi, Xinlei Chen, O. Litany, and L. Guibas. Imvotenet: Boosting 3d object detection in point clouds with image votes. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4403–4412, 2020.
- [13] Nitish Srivastava, Geoffrey E. Hinton, A. Krizhevsky, Ilya Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

- [14] Jamie Watson, Michael Firman, Aron Monszpart, and G. Brostow. Footprints and free space from a single color image. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11–20, 2020.
- [15] Zaiwei Zhang, Bo Sun, Haitao Yang, and Qi-Xing Huang. H3dnet: 3d object detection using hybrid geometric primitives. In *ECCV*, 2020.