Supplementary to RBGNet

In the supplementary material, we first elaborate on fine sampling (§A). Then, more implementation details and ray-based representation discussion are provided in §B and §C. Finally, we present per-category evaluation, visualization of positive anchor points and quantitative results from §D.1 to §D.3. We also discuss the limitation of RBGNet in §E.

A. Details on Fine Sampling

In this section, we will provide more technical details on fine sampling. After obtaining coarse point masks, $\mathcal{M}^{(c)} = \{m_{n,k}^{(c)}\}_{k=1,n=1}^{K_c,N}$, we generate fine anchor points biased towards the dense part of its corresponding object. To achieve this goal, we apply inverse transform sampling to uniformly generate K_f anchor points set $Q^{(f)} = \{q_k^{(f)}\}_{k=1}^{K_f}$ on positive regions of the n^{th} ray (we remove the subscript n of $Q_n^{(f)}$ for simplicity) based on the predicted coarse point masks $\{m_k^{(c)}\}_{k=1}^{K_c}$.

To be specific, we first normalize the coarse point masks as $\hat{m}_k^{(c)} = m_k^{(c)} / \sum_{j=1}^{K_c} m_j^{(c)}$ to produce a piecewise-constant probability density function (PDF). Then we translate it into the cumulative distribution function (CDF). Finally, sampling with the CDF at uniform steps concentrates samples around regions with positive coarse point masks.

To further illustrate it, we provide a demo case and visualize it in Fig.1. The number of coarse and fine anchor points on each ray is 8 and 10 respectively.

- The predicted coarse point masks of n^{th} ray are:

$$\{m_k^{(c)}\}_{k=1}^{K_c} = \{0, 1, 0, 1, 0, 0, 1, 0\}. \quad (1)$$

- We compute the piece-wise PDF by normalizing $m_k^{(c)}$:

$$\{\hat{m}_k^{(c)}\}_{k=1}^{K_c} = \{0., 1/3, 0., 1/3, 0., 0., 1/3, 0.\}. \quad (2)$$

- Then we convert PDF to CDF, $\{C_k^{(c)}\}_{k=1}^{K_c}$

$$\{C_k^{(c)}\}_{k=1}^{K_c} = \{0., 1/3, 1/3, 2/3, 2/3, 2/3, 1., 0.\}. \quad (3)$$

- Finally, sample 10 points based on the CDF at uniform steps and inverse them to original distribution. As demonstration in Fig.1, the relative distance of fine points from object center are as follows:

$$\{\hat{q}_k^{(f)}\}_{k=1}^{K_f} = \{0.1625, 0.2000, 0.2375, 0.4000, 0.4375, 0.4750, 0.7625, 0.8000, 0.8375, 0.875\} \times RayScale. \quad (4)$$

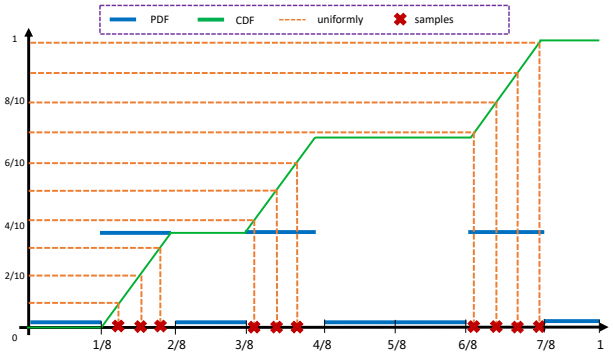

Figure 1: One case for visualization of fine sampling.

layer name	input layer	κ	α	β	MLP Channels
sa2	sa1	1024	896	128	[256, 256]
sa3	sa2	512	448	64	[256, 256]
sa4	sa3	256	224	32	[256, 256]

Table 1: Backbone network architecture: FBS parameters.

B. Implementation Details.

B.1. RBGNet architecture details.

As mentioned in the main paper, the RBGNet architecture consists of a backbone with foreground biased sampling, a voting layer, a ray-based feature grouping module and a proposal module.

The backbone network, based on the PointNet++ architecture, has four set abstraction layers and two feature upsampling layers. We follow the same layer parameters (*e.g.* ball-region radius, number of sample points and MLP channels) as VoteNet [5]. To sample points biased towards object surface, we append a segmentation head for estimating the foreground confidence of each point. The detailed layer parameters are shown in Table 1. The voting module is the same as VoteNet. Note that, in training stage, we generate M proposals from the votes by vote FPS (samples M clusters based on votes' XYZ), in test stage, we apply vote FPS on ScanNet V2 and seed FPS on SUN RGB-D (sample on seed XYZ and then find the votes corresponding to the sampled seeds).

The ray-based feature grouping module consists of two parts, *Ray Point Generation* and *Feature Enhancement by*

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	frig	showr	toil	sink	bath	ofurn	mAP
VoteNet [5]	47.87	90.79	90.07	**90.78**	60.22	53.83	43.71	55.56	12.38	66.85	66.02	52.37	52.05	63.94	97.40	52.32	92.57	43.37	62.90
MLCVNet [8]	42.45	88.48	89.98	87.40	63.50	56.93	46.98	**56.94**	11.94	63.94	76.05	56.72	**60.86**	65.91	98.33	59.18	87.22	47.89	64.48
BRNet [2]	49.90	88.30	91.90	86.90	69.30	59.20	45.90	52.10	15.30	72.00	76.80	57.10	60.40	73.60	93.80	58.80	92.20	47.10	66.10
H3DNet* [10]	49.40	88.60	91.80	90.20	64.90	**61.00**	51.90	54.90	18.60	62.00	75.90	57.30	57.20	75.30	97.90	67.40	92.50	53.60	67.20
Group-free [3]	**55.40**	86.60	91.80	86.60	73.30	54.50	49.40	47.70	13.10	63.30	82.40	63.30	53.20	74.00	99.20	67.70	91.70	55.80	67.20
Ours	52.62	**91.34**	**93.07**	89.71	**73.57**	60.10	**51.96**	53.53	**20.01**	**72.65**	**82.57**	**63.58**	59.79	**76.03**	**99.28**	**74.79**	**92.67**	**55.88**	**70.20**

Table 2: 3D object detection scores per category on the ScanNetV2 dataset, evaluated with mAP@0.25 IoU. * means that H3DNet [10] only provide the checkpoint with 4 PointNet++ backbones.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	frig	showr	toil	sink	bath	ofurn	mAP
VoteNet [5]	14.62	77.85	73.11	80.49	46.54	25.09	15.98	41.85	2.50	22.34	33.35	25.02	31.04	17.58	87.75	23.05	81.60	18.66	39.91
H3DNet* [10]	20.50	79.70	80.10	79.60	56.20	29.00	21.30	45.50	4.20	33.50	50.60	37.30	41.40	37.00	89.10	35.10	90.20	35.40	48.10
Group-free [3]	23.80	77.20	81.60	65.10	62.80	35.00	21.30	39.40	7.00	33.10	**66.30**	39.30	43.90	**47.00**	91.20	38.50	85.10	37.40	49.70
BRNet [2]	28.70	80.60	81.90	80.60	60.80	35.50	22.20	48.00	7.50	43.70	54.80	39.10	**51.80**	35.90	88.90	38.70	84.40	33.00	50.90
Ours	**30.69**	**80.95**	**86.48**	**84.82**	**66.45**	**40.37**	**29.59**	**48.60**	**7.96**	**44.76**	59.14	**40.83**	44.80	39.78	**92.92**	**45.30**	**90.90**	**41.49**	**54.21**

Table 3: 3D object detection scores per category on the ScanNetV2 dataset, evaluated with mAP@0.50 IoU. * means that H3DNet [10] only provide the checkpoint with 4 PointNet++ backbones.

layer name	Input channel	MLP Channels
$\mathcal{F}^{(c)}_{point}$	5×32	[32,]
$\mathcal{F}^{(f)}_{point}$	3×32	[32,]
$\mathcal{F}^{(c)}_{ray}$	66×32	[256, 128]
$\mathcal{F}^{(f)}_{ray}$	66×32	[256, 128]
$\mathcal{F}_{fuse}$	256	[256, 128]

Table 4: Detailed layer parameters of Feature Enhance module. In our case, the number of rays (N), coarse points (K_c) and fine points (K_f) are 66, 5, and 3.

Determined Rays. After generating a set of vote cluster centers $\{c_i\}_{i=1}^{M}$ based on the vote sampling and grouping, where $c_i = [v_i, f_i]$ (the vote center position $v_i \in \mathbb{R}^3$ and its corresponding features $f_i \in \mathbb{R}^{128}$), $M = 256$ (the number of vote clusters). In our case, for each vote cluster, 66 rays are emitted uniformly from the cluster center with the determined angles and lengths generated in §3.2.1. We use a *MLP [128, 128]* to regress the object scale of each cluster. As for the coarse-to-fine anchor point generation step, the number of coarse points (K_c) is 5 and fine points (K_f) is 3. To extract local feature of each anchor point, we apply two SA layers (coarse and fine), to aggregate the features of these seed points within a fixed radius (r = 0.2m) surrounding the query points. The two SA layers both have a receptive field specified by r=0.2m, a *MLP[128, 64, 32]* for feature transform. But coarse layer samples 8 points by ball query operation and fine layer is 4 points. In term of the point mask prediction, we use a *MLP[32+128, 32, 2]* to estimate the positive mask based on corresponding cluster features and local features. In *Feature Enhancement module*, all the functions $\mathcal{F}^{**}$ are MLP networks. The detailed layer parameters are shown in Table 4.

The proposal module is a two-layer *MLP[128, 128]*. We follow [5] on how to estimate the 3D bounding boxes, except for size prediction that we adopts class-agnostic head to regress bounding box size directly. The layer's output has *5+2NH+3+NC* where the first five channels are for objectness classification and center regression (relative to the vote cluster center), *2NH* channels are for heading bins classification and offsets regression, 3 is the scale regression for height, width and length, *NC* is the number of semantic classes. In SUN RGB-D: *NH* = 12, *NC* = 10, and in ScanNet: *NH* = 1, *NC* = 18, due to the axis aligned bounding box.

B.2. RBGNet loss function details.

As mentioned in the main paper, our model is trained end-to-end with a multi-task loss including foreground biased sampling $\mathcal{L}_{\text{fbs}}$, voting regression $\mathcal{L}_{\text{vote-reg}}$, ray-based feature grouping $\mathcal{L}_{\text{rbfg}}$, objectness $\mathcal{L}_{\text{obj-cls}}$, bounding box estimation $\mathcal{L}_{\text{box}}$, and semantic classification $\mathcal{L}_{\text{sem-cls}}$ losses.

$$L = \lambda_{\text{vote-reg}}\mathcal{L}_{\text{vote-reg}} + \lambda_{\text{fbs}}\mathcal{L}_{\text{fbs}} + \lambda_{\text{rbfg}}\mathcal{L}_{\text{rbfg}} + \lambda_{\text{obj-cls}}\mathcal{L}_{\text{obj-cls}} + \lambda_{\text{box}}\mathcal{L}_{\text{box}} + \lambda_{\text{sem-cls}}\mathcal{L}_{\text{sem-cls}}. \quad (5)$$

Following the setting in VoteNet [5], we use the same loss terms $\mathcal{L}_{\text{vote-reg}}$, $\mathcal{L}_{\text{obj-cls}}$, $\mathcal{L}_{\text{box}}$ and $\mathcal{L}_{\text{sem-cls}}$, but $\mathcal{L}_{\text{box}}$ is class-agnostic and contains an additional corner loss defined in [6] for accurate bounding box estimation,

$$\mathcal{L}_{\text{box}} = \lambda_{\text{size-reg}}\mathcal{L}_{\text{size-reg}} + \lambda_{\text{corner}}\mathcal{L}_{\text{corner}} + \lambda_{\text{angle-cls}}\mathcal{L}_{\text{angle-cls}} + \lambda_{\text{angle-reg}}\mathcal{L}_{\text{angle-reg}}. \quad (6)$$

As discussed in §3.4, $\mathcal{L}_{\text{fbs}}$ is a cross entropy loss used to supervise foreground sampling (see §3.2). $\mathcal{L}_{\text{rbfg}}$ is the sum loss of ray-based feature grouping module defined as follows:

$$\mathcal{L}_{\text{rbfg}} = \lambda_{\text{scale-reg}}\mathcal{L}_{\text{scale-reg}} + \lambda_{\text{c-cls}}\mathcal{L}_{\text{c-cls}} + \lambda_{\text{f-cls}}\mathcal{L}_{\text{f-cls}}. \quad (7)$$

The balancing factors are set default as $\lambda_{\text{vote-reg}}$=10.0, λ_{fbs}=3.0, λ_{rbfg}=10.0, $\lambda_{\text{obj-cls}}$=5.0, λ_{box}=10.0, $\lambda_{\text{sem-cls}}$=1.0,

	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	mAP
VoteNet [5]	75.50	85.60	31.90	77.40	24.80	27.90	58.60	67.40	51.10	90.50	59.10
MLCVNet [8]	79.20	85.80	31.90	75.80	26.50	31.30	61.50	66.30	50.40	89.10	59.80
H3DNet* [10]	73.80	85.60	31.00	76.70	29.60	33.40	65.50	66.50	50.80	88.20	60.10
BRNet [2]	76.20	86.90	29.70	77.40	29.60	35.90	65.90	66.40	51.80	91.30	61.10
HGNet [1]	78.00	84.50	**35.70**	75.20	**34.30**	37.60	61.70	65.70	51.60	91.10	61.60
Group-free [3]	80.00	87.80	32.50	79.40	32.60	36.00	66.70	70.00	53.80	91.10	63.00
Ours	**80.68**	**88.41**	34.56	**82.79**	32.09	**38.76**	**66.77**	**71.06**	**54.55**	**91.37**	**64.10**

Table 5: 3D object detection scores per category on the SUN RGB-D dataset, evaluated with mAP@0.25 IoU. * means that H3DNet [10] only provide the checkpoint with 4 PointNet++ backbones.

	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	mAP
VoteNet [5]	45.40	53.40	6.80	56.50	5.90	12.00	38.60	49.10	21.30	68.50	35.80
H3DNet* [10]	47.60	52.90	8.60	60.10	8.40	20.60	45.60	50.40	27.10	69.10	39.00
BRNet [2]	55.50	63.80	9.30	61.60	10.00	**27.30**	53.20	56.70	28.60	70.90	43.70
Group-free [3]	64.00	67.10	12.40	62.60	**14.50**	21.90	49.80	58.20	29.20	72.20	45.20
Ours	**65.74**	**68.02**	**12.99**	**65.46**	12.81	25.84	**54.89**	**59.55**	**32.38**	**74.50**	**47.22**

Table 6: 3D object detection scores per category on the SUN RGB-D dataset, evaluated with mAP@0.50 IoU. * means that H3DNet [10] only provide the checkpoint with 4 PointNet++ backbones.

$\lambda_{\text{size-reg}}$=0.11, λ_{corner}=0.33, $\lambda_{\text{angle-cls}}$=0.1, $\lambda_{\text{angle-reg}}$=0.11, $\lambda_{\text{scale-reg}}$=0.11, $\lambda_{\text{c-cls}}$=0.2 and $\lambda_{\text{f-cls}}$=0.2. $\mathcal{L}_{\text{scale-reg}}$, $\lambda_{\text{size-reg}}$ and $\lambda_{\text{angle-reg}}$ are all the smooth ℓ_1 loss and their betas are 0.0625, 0.0625, 0.0400 separately.

B.3. Other Grouping Mechanisms

To further ablate the effectiveness of our ray-based feature grouping module, we refer several grouping strategies in 3D object detection as baselines and compare with them in §4.4. For a fair comparison, we only switch the feature aggregation mechanism while all other settings remain unchanged (*e.g.* backbone with FBS, vote-FPS, proposal module). Here we give some detailed descriptions.

Voting. The voting mechanism is first introduced by VoteNet [5]. In our implementation, it is actually the VoteNet equipped with FBS, corner loss, vote-FPS in test stage and optimized hyperparameters.

RoI-Pooling. For a given object proposal, the points within the predicted box are aggregated together. We adopt the similar implementation with [2], predict the bounding boxes based on the voting cluster features and aggregate the points within the corresponding boxes by max-pooling. Finally, the aggregated features and cluster features are concatenated for 3D object detection.

Back-Tracing. It is first formulated in [2]. In our implementation, it is similar to RoI-Pooling described above, except that the prediction of bounding boxes is replaced by the maximum offsets of 6 directions. And then the points are aggregated by the balls uniformly sampled along the rays to enhance 3D bounding box estimation.

Group-free. We replace the ray-based feature grouping module with a transformer network adopted in Group-free [3]. For a fair comparison, we use vote-FPS for initial object candidate sampling instead of KPS. Then, the same as group-free [3], we adopt the transformer as the decoder to leverage all the seed points to compute the object feature of each candidate.

method	mAP@0.25	mAP@0.5
Reg-Ray(R66)	68.0	50.2
Our(R66)	**69.6**	**53.6**

Table 7: The average performance of different ray representations on ScanNet V2.

C. Ray-based Representation Discussion

There are many choices for anchor point generation, such as classification [4] and regression [7, 9].

- **Regression:** Given the center and surface points of an object, they want to represent the shape by polar coordinates. The length of n rays can be computed easily. Then, the model regresses the length of each ray and captures the points when the ray terminates somewhere.
- **Classification:** Given an instance, model predicts far bounds of all rays and samples a fixed number of potential query points on each ray, and then extracts local features and classifies those points whether belong to corresponding object to generate reasonable anchor points. Our model adopts this way and we will discuss why do we choose it.

In 2D perception community, some methods also represent object shape by rays [7, 9] in regression way, which applies the angle and distance as the coordinate to locate points. However, due to the particular property of point clouds, this

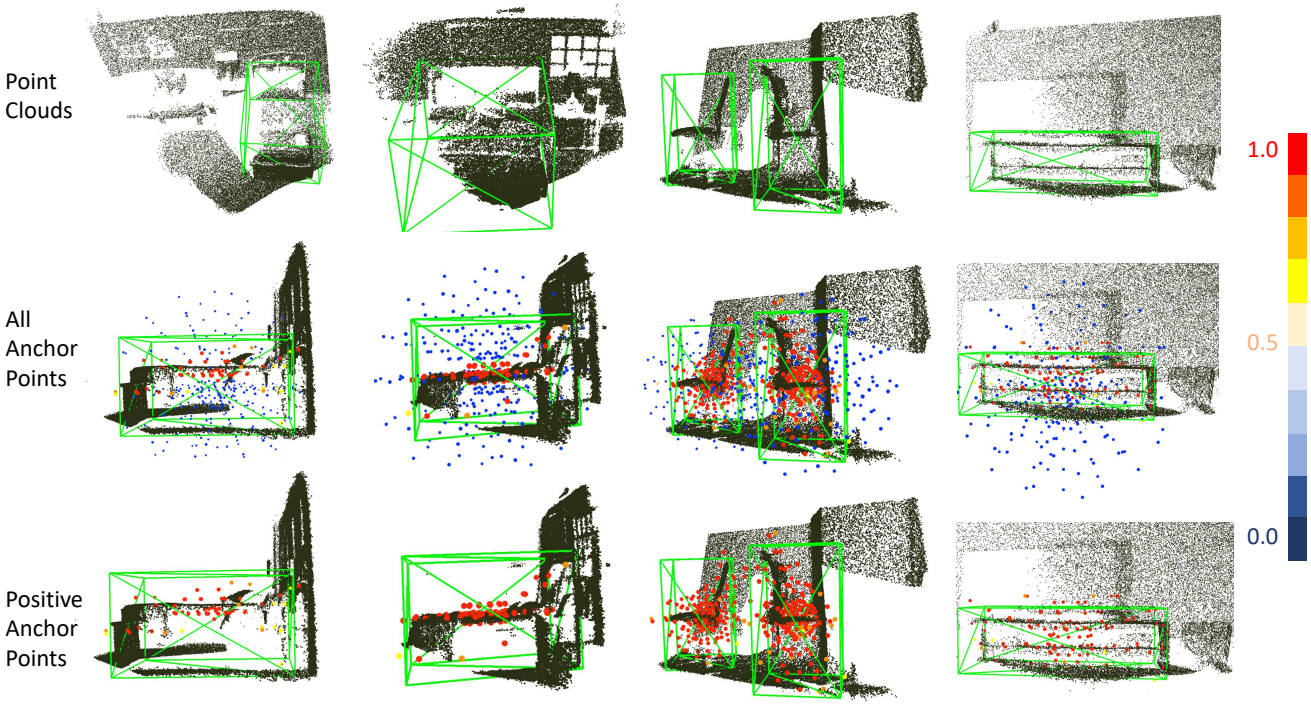

Figure 2: Qualitative results of shape distribution our model learned.

regression pipeline has many problems in 3D scenario, *i.e.*, i) center is outside of the object, no intersection with the object surface at some angles, ii) limited expressive ability on concave shape, one ray may has multiple intersections. Compare to regression, classification pipeline is more reasonable to represent point clouds and doesn't have the above problems, so we choose it to generate anchor points. In Table 7, we show the results of the two representations, classification approach performs better than regression.

D. More Results

D.1. Per-class Evaluation

We evaluate per-category on ScanNet V2 and SUN RGB-D under different IoU thresholds. Table 2 and Table 3 report the results on 18 classes of ScanNetV2 with 0.25 and 0.5 box IoU thresholds respectively. Table 5 and Table 6 show the results on 10 classes of SUN RGB-D with 0.25 and 0.5 box IoU thresholds. Our approach outperforms the baseline VoteNet [5] and prior state-of-the-art methods Group-free [3] significantly in almost every category. These improvements are achieved by using ray-based feature grouping and foreground biased sampling to better encode object surface geometry.

D.2. Visualization of Positive Anchor Points

Fig.2 shows the scores of coarse anchor points predicted from our RBGNet in a typical SUN RGB-D scene. We clearly see that the high responses are almost on the object surface (bed, chair *etc*.) while low responses are on the empty space or background surface. This verifies that our method can really learn the shape distribution and boost point-based 3D detectors.

D.3. Quantitative Results

We provide more qualitative comparisons between our method and the top-performing reference methods, such as Group-free [3] and VoteNet [5], on the ScanNet V2 and SUN RGB-D datasets. Please see Fig. 3 for more qualitative results.

E. Limitations

Although our method achieves promising performance on multiple datasets, there are still some limitations. Compared with the previous approaches, the performance of RBGNet is significantly better in the case of a large number of rays. However, there is a trade-off between computational cost and performance improvement, as shown in main paper. In the future, we hope to discover approaches that can encode surface geometry more efficiently.

References

[1] Jintai Chen, Biwen Lei, Qingyu Song, Haochao Ying, Danny Z Chen, and Jian Wu. A hierarchical graph network for 3d object detection on point clouds. In *CVPR*, 2020. 3

[2] Bowen Cheng, Lu Sheng, Shaoshuai Shi, Ming Yang, and Dong Xu. Back-tracing representative points for voting-based 3d object detection in point clouds. In *CVPR*, 2021. 2, 3

[3] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. 2021. 2, 3, 4, 6

[4] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 3

[5] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019. 1, 2, 3, 4, 6

[6] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 2018. 2

[7] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *CVPR*, 2020. 3

[8] Qian Xie, Yu-Kun Lai, Jing Wu, Zhoutao Wang, Yiming Zhang, Kai Xu, and Jun Wang. Mlcvnet: Multi-level context votenet for 3d object detection. In *CVPR*, 2020. 2, 3

[9] Wenqiang Xu, Haiyang Wang, Fubo Qi, and Cewu Lu. Explicit shape encoding for real-time instance segmentation. In *ICCV*, 2019. 3

[10] Zaiwei Zhang, Bo Sun, Haitao Yang, and Qixing Huang. H3dnet: 3d object detection using hybrid geometric primitives. In *ECCV*, 2020. 2, 3

Figure 3: Qualitative results on ScanNet V2(top) and SUN RGB-D(down). The baseline methods are Group-free [3] and VoteNet [5].