

# 2.5D-VoteNet: Depth Map based 3D Object Detection for Real-Time Applications

## Supplementary Materials

BMVC 2021 Submission # 501

### 1 Introduction

This document provides additional technical details, analysis experiments and qualitative results to our main paper.

### 2 Implementation of RDConv

In practice, the 2D convolution operation can be separated into three sub-operations. First, a 2D map is unfolded, in order to acquire data in each sliding windows (as known as local patches). This sub-operation is usually a built-in function of frameworks, *e.g.* *unfold* in Pytorch or *im2col* in MATLAB. Then, convolution kernels are applied to each window. The calculation is implemented as matrix multiplication. At last, the result of multiplication is reshaped into a 2D map.

RDConv normalizes the depth value in each sliding window. With the given unfolded map, the local reference depth is easy to be calculated. Thus, the only difference between RDConv and Conv is that RDConv normalizes the depth value, after the depth map is unfolded. All sub-operations are provided in main-stream frameworks.

Our source code is provided as supplementary materials as well.

### 3 Additional Experiments

#### 3.1 Different Backbones

We test the geometry-only 2.5D-VoteNet with different input resolutions and model depths. As shown in Tab. 1, the mAP drops by 1.1% when the resolution is reduced to  $320 \times 416$ . It implies that higher input resolution is beneficial for 3D detection. To research the impact of model depth, we replace the encoder in the 2D backbone with ResNet-18 and ResNet-50 [9]. The mAP of ResNet-34 based backbone is 0.5% better than the ResNet-18 based backbone. However, we get worse results with ResNet-50. It's probably due to over-fitting since only  $\sim 5K$  depth maps are used to train the model.

Resolution	Encoder	mAP (%)
416×544	ResNet-18	60.3
320×416	ResNet-34	59.7
416×544	ResNet-34	<b>60.8</b>
416×544	ResNet-50	59.9
416×544	VGG-16	58.7
416×544	MobileNetV2	60.1

Table 1: Detection results with different structures of 2D encoders in the backbone. All models use solely depth maps as input.

We also test the performance with VGG-16 [1] and MobileNetV2 [2]. Similar to the configuration in our main paper, we modify VGG-16 and MobileNetV2 by removing their classifiers and replacing their first layer with a strided  $7 \times 7$  Conv and a strided  $7 \times 7$  RDConv. The models reach similar performance with ResNet-34 and MobileNetV2. The performance with VGG-16 is 2.1% lower than ResNet-34 and 1.4% lower than MobileNetV2. We hypothesize that the residual connections in 2D backbones are beneficial for our models. The results also show that our method is not sensitive to the choice of 2D encoders.

### 3.2 Reference Depth

RDConv uses the masked mean average depth in each sliding window as the reference. Here, we further test the mean value and max value of each sliding window as reference. Moreover, depth maps in real-world applications may contain more noise than SUN RGB-D dataset [3]. To validate the robustness of our networks, we add extra bad pixels in this experiment. Fig. 1 shows that the three choices deliver similar results when no extra bad pixels are added. However, the mean value brings slightly better result than the max value and center value, when depth maps have more bad pixels. We hypothesize that our models are less sensitive to noise in data, if mean value is used as reference depth in RDConv.

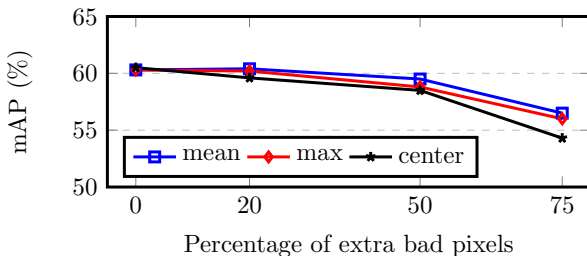


Figure 1: Performance of different choices of reference depth. Bad pixels are randomly added to both the training and validation data. Absolute depth is not used in this experiment.

### 3.3 Calibration-Awareness of the Backbone

With the known camera calibration, a depth map can be converted to a point cloud. As mentioned in the main paper, the 2D backbone of our models doesn't take camera calibration

as input and is therefore unaware of the calibration. On the contrary, point cloud based networks are calibration-aware, as point clouds already contain this information. Since we propose to use a 2D CNN to extract features directly from depth maps, the backbone of our detectors receive less information than point cloud based methods. More importantly, our models adopt the detection head with a voting module from VoteNet [4]. The voting module, consisting of a shared MLP, takes the feature carriers (also called points of interest) and predicts the 3D offset from each feature carrier to the closet 3D object centers (the offset is also called votes). As the voting module only uses high-level features from the backbone, which is unaware of the camera calibration, it's questionable if the voting module can generate accurate votes.

During the training, the ground truth votes can be calculated from bounding box labels. We use smoothed L1-norm as loss function to optimize the votes, following [4]. Thus, the voting loss on the validation set can be used a metric to evaluate the quality of voting. We train VoteNet and 2.5D-VoteNet to convergence and compare their voting losses. As the two solid lines in Fig. 2 show, although our 2D backbone is unaware of the calibration, our model has similar voting loss as the baseline VoteNet.

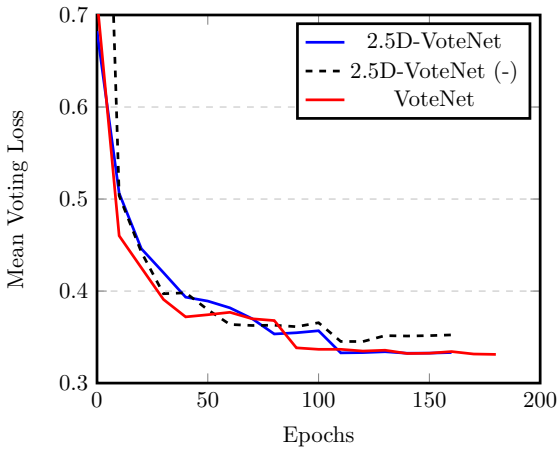


Figure 2: Mean validation voting loss of VoteNet [4] and 2.5D-VoteNet. Note the two methods use different learning rate schedules. The solid lines show the results with the standard configuration, while the dashed line shows the voting loss of 2.5D-VoteNet without rotation and scaling as data augmentation. The two solid lines converge at similar values (0.331 and 0.333, respectively), while the the dashed line converges at 0.355.

Another interesting question is, would the performance of our models be improved, if the calibration is explicitly or implicitly provided. The straightforward method is to concatenate extra features to feature carriers. We experiment with different choices of extra features, including 3D coordinates of feature carriers in upright coordinate system (intrinsic and extrinsic), the 3D coordinates of feature carriers in camera coordinate (intrinsic), the vectorized camera matrix (intrinsic), and the Euler angles of the camera rotation (extrinsic). We also apply CAM-Conv [4], which incorporates camera calibration into CNN.

We illustrate the converged mAP of these variants in Tab. 2. Interestingly, although extra features bring camera calibrations to the detection head, they either worsen the results or don't make significant differences.

Features	mAP (%)
Baseline (256-D features)	<b>60.8</b>
+ 3D Coord. (upright)	60.3
+ 3D Coord. (camera)	60.1
+ Camera Matrix	60.6
+ Euler Angles	60.7
+ CAM-Conv [2]	60.6

Table 2: Detection results when camera calibrations are concatenated as extra features to the output of the 2D backbone. The baseline in this experiment is geometry-only 2.5D-VoteNet. The extra features are normalized before the concatenation if necessary.

Data Augmentation	mAP (%)
w/ Scaling and Rotating	<b>60.8</b>
wo/ Scaling and Rotating	56.4

Table 3: Detection results with and without scaling and rotating the depth maps. We evaluate the geometry-only 2.5D-VoteNet in this experiment.

To explain these observations, we hypothesize that the 2D backbone of 2.5D-VoteNet already learns invariance to the camera calibration, thanks to the data augmentation we use. Specifically, we randomly scale and rotate depth maps and images around principle points during training. Meanwhile, we augment the camera calibrations accordingly, so that the augmented depth maps can be converted to the same 3D coordinates as the original depth maps. By doing this, the network has to learn calibration-invariance, as the calibration is changing during the training, but the 3D geometry remains the same. To validate our hypothesis, we train our network without scaling and rotation. As Tab. 3 shows, the mAP drops by 4.4%, when the camera calibration is not augmented. Also, the dashed line in Fig.2 shows that our model has higher voting loss with this configuration, which shows that the network doesn't learn accurate 3D spatial relationships. These results imply that our models learn calibration-invariance via data augmentation. Therefore, the extra features mentioned above are unnecessary and can even lead to over-fitting.

In this section, we discuss the calibration-awareness of our method in detail. The experiment results show that our models learn calibration-invariance implicitly. Consequently, our models work well, although their 2D backbones don't use the camera calibration as input. However, the explicit calibration-awareness might still be beneficial for the detection result. Instead of directly concatenating extra features to the output of the backbone, we intend to design calibration-aware 2D backbones in future works to further explore this topic.

### 3.4 Run-time of Pre-processing

In our main paper, we report the inference-time of our models but exclude the run-time of pre-processing, since it runs parallel on the CPU. However, pre-processing plays an important role, when the CPU has less computational power or the application is sensitive to the latency. In this section, we compare the pre-processing times of our depth map based detection pipeline and the baseline VoteNet.

For our depth map based pipeline, we first scale the depth map and RGB-image to a

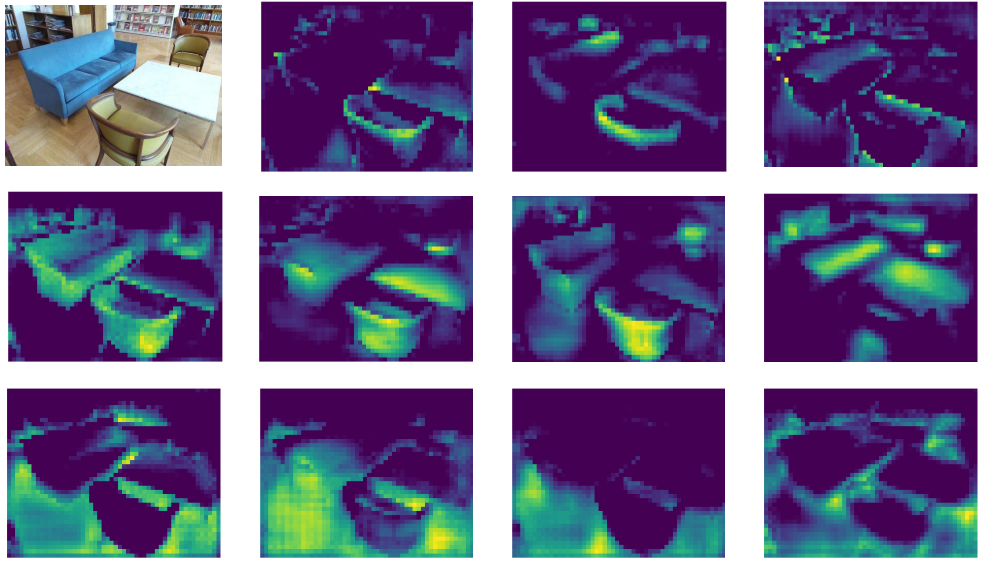


Figure 3: Examples of feature maps learned from depth maps. The brighter the color, the greater the activation. The scene is illustrated by the RGB image. However, the network doesn't use color information as input. The feature maps in the first row have high activation at low level features *e.g.* edges and corners. The second row shows high activation on objects of interest. The last row ignores the objects and gets activated at the background.

fixed size of  $416 \times 544$  and normalize their pixel values to  $[0, 1]$ . Also, we create an 8-times down-sampled copy of the depth map. To build feature carriers, we lift the down-sampled depth map to 3D space and further sample 1024 points using farthest point sampling.

The publication of VoteNet [1] uses off-line pre-processing to prepare point clouds. For fair comparison, in this experiment we directly lift the original depth map into 3D space and randomly sample 20000 points.

Our implementations use Numpy and OpenCV to process point clouds and images. Since these libraries can automatically activate multi-threading, we explicitly set the thread number to one in their back-ends, in order to control variables in this experiment. We use a C++ implementation of farthest point sampling via Pybind11. We measure the run-time on a desktop computer with an Intel Core i7-8700 CPU and an Nvidia RTX 2080Ti GPU. The reported times include loading data from the memory and sending data to the GPU.

As shown in Tab. 4, both our models require shorter pre-processing time than the baseline VoteNet, as converting depth maps to point clouds is expensive. The results show that our depth map based pipeline also accelerates the pre-processing, in comparison to point cloud based methods.

It's worth noting that the speed of pre-processing can significantly vary with different implementations and optimization levels (*e.g.* C++ vs. Python, multi-thread vs. single-thread). Our results should be considered as qualitative, because the reported run-times are implementation-specific.

Methods	Pre-processing Time (ms)
VoteNet	19.3
2.5D-VoteNet (geo-only)	<b>11.6</b>
2.5D-VoteNet (fused-only)	17.1

Table 4: Comparison of single-thread pre-processing time between our models and the base-line. Our geometry-only model requires less time than the fused model, since it doesn’t load and process RGB images. The single-thread pre-processing time of the fused model is longer than its inference time (14.5 ms). However, it wouldn’t limit the frame rate of our models, as multi-threading or multi-processing are usually employed in real-world applications

## 4 Qualitative Results

### 4.1 Learned Feature Maps

An interesting question about depth map based 3D detection pipeline is, what features can the 2D backbone actually learn. In this experiment, we visualize the output feature map of the backbone, which has the resolution of  $52\times68$  and 256 channels. Specifically, we normalize each channel of the feature map to the interval  $[0, 1]$  and plot them as color maps. Some representative feature maps are shown in Fig. 3. The quantitative results imply that the 2D backbone is able to learn rich features from depth maps.

### 4.2 Detection Results on SUN RGB-D

We show more detection results on SUN RGB-D [2] validation set in Fig. 4.

### 4.3 Detection Results on ScanNet

We randomly picked some frames from ScanNet [1] and illustrate the detection results in Fig. 5.

### 4.4 Detection with Live Videos

We also provide detection results on live videos as supplementary materials.

## References

[1] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[2] Jose M. Facil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. CAM-ConvS: Camera-Aware Multi-Scale Convolutions for Single-View Depth. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

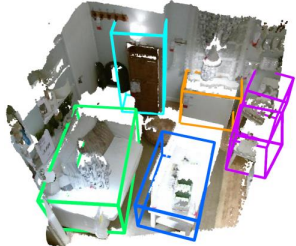
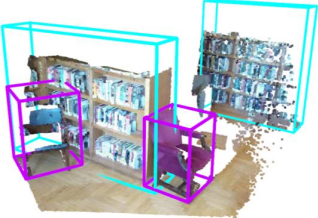
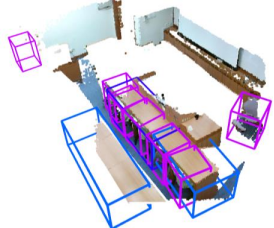
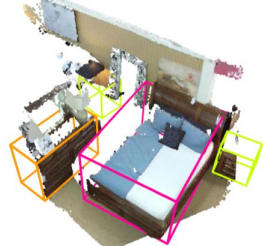
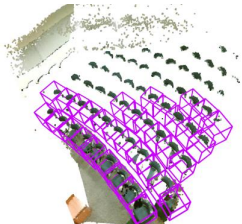
[4] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep Hough voting for 3d object detection in point clouds. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 9277–9286, October 2019.

[5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, June 2018.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[7] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, June 2015.





Bed	Table	Sofa	Chair	Toilet
Desk	Dresser	Night Stand	Bookshelf	Bathtub

Figure 4: Additional qualitative results on SUN RGB-D validation set.



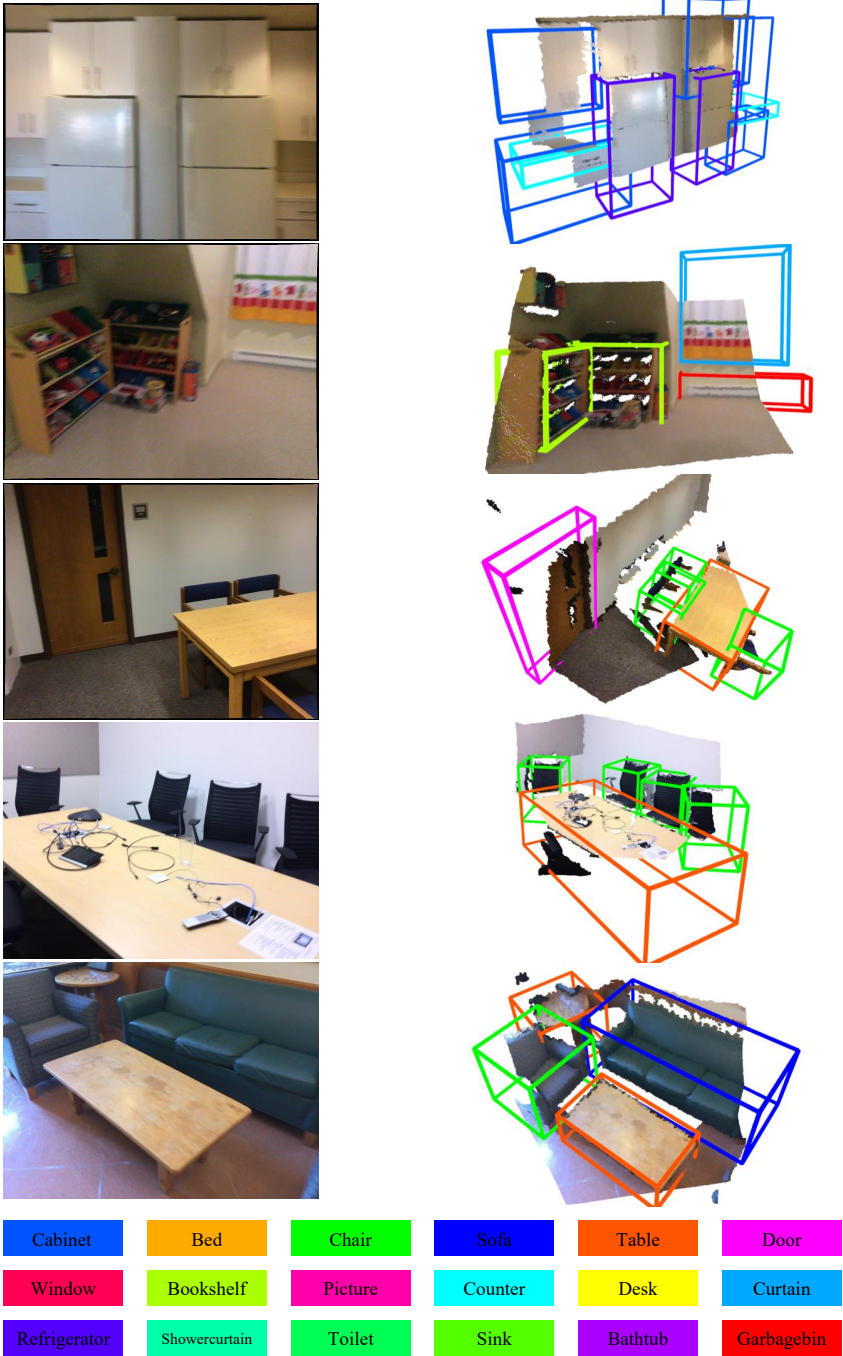


Figure 5: Qualitative results on ScanNet validation set (frame-level).