ImVoteNet: Boosting 3D Object Detection in Point Clouds with Image Votes Supplmentary Material

Charles R. Qi*† Xinlei Chen*[1] Or Litany[1,2] Leonidas J. Guibas[1,2]
[1]Facebook AI [2]Stanford University

A. Overview

In this supplementary, we provide more details on the IMVOTENET architecture in Sec. B, including point cloud network architecture, 2D detector, 2D votes and image votes lifting. We also show visualizations of the sparse point clouds in Sec. C.

B. Details on IMVOTENET Architecture

In this section, we explain the details in the IMVOTENET architecture. Sec. B.1 provides details in the point cloud deep net as well as the training procedure. Further details on the 2D detector and 2D votes are described in Sec. B.2 while details on lifting 2D votes with general camera parameters are described in Sec. B.3.

B.1. Point Cloud Network

Input and data augmentation. The point cloud backbone network takes a randomly sampled point cloud of a SUN RGB-D [8] depth image with $20k$ points. Each point has its XYZ coordinate as well as its height (distance to floor). The floor height is estimated as the 1% percentile of heights of the all points. Similar to [4], we augment the input point cloud by randomly sub-sampling the points from the depth image points on-the-fly. Points are also randomly flipped in both horizontal directions and randomly rotated along the up-axis by Uniform[-30,30] degrees. Points are also randomly scaled by Uniform[-.85, 1.15]. Note that the point height and the camera extrinsic are updated accordingly with the augmentation.

Network architecture. We adopt the same PointNet++ [5] backbone network as that in [4] with four set abstraction (SA) layers and two feature propagation/upsamplng (FP) layers. With input of $N\times 4$ where $N{=}20k$, the output of the backbone network is a set of seed points of $K\times(3+C)$ where $K{=}1024$ and $C{=}256$.

As for voting, different from VOTENET that directly predicts votes from the seed points, here we fuse lifted image votes and the seed points before voting. As each seed point can fall into multiple 2D detection boxes, we duplicate a seed point q times if it falls in q overlapping boxes. Each duplicated seed point has its feature augmented with a concatenation of the following image vote features: 5-dim lifted geometric cues (2 for the vote and 3 for the ray angle), 10-dim (per-class) semantic cues and 3-dim texture cues. In the end the fused seed point has 3-dim XYZ coordinate and a 274-dim feature vector.

The voting layer takes the seed point and maps its features to votes through a multi-layer perceptron (MLP) with FC output sizes of 256, 256 and 259, where the last FC layer outputs XYZ offset and feature residuals (with regard to the 256-dim seed feature) for the votes. As in [4], the proposal module is another set abstraction layer that takes in the generated votes and generate proposals of shape $K'\times(5{+}2NH{+}4NS{+}NC)$ where K' is the number of total duplicated seed points and the output dimension consists of 2 objectness scores, 3 center regression values, $2NH$ numbers for heading regression (NH heading bins) and $4NS$ numbers for box size regression (NS box anchors) and NC numbers for semantic classification.

Training procedure. We pre-train the 2D detector as described more in Sec. B.2 and use the extracted image votes as extra input to the point cloud network. We train the point cloud deep net with the Adam optimizer with batch size 8 and an initial learning rate of 0.001. The learning rate is decayed by $10\times$ after 80 epochs and then decayed by another $10\times$ after 120 epochs. Finally, the training stops at 140 epochs as we find further training does not improve performance.

B.2. 2D Detector and 2D Cues

2D detector training. While IMVOTENET can work with any 2D detector, in this paper we choose Faster R-CNN [6], which is the current dominant framework for bounding box detection in RGB. The detector we used has a basic ResNet-50 [1] backbone with Feature Pyramid Networks (FPN) [2] constructed as $\{P_2, P_3, \ldots, P_6\}$. It is pre-trained on the

*: equal contributions.
†: work done while at Facebook.

COCO *train2017* dataset [3] achieving a *val2017* AP of 41.0. To adapt the COCO detector to the specific dataset for 2D detection, we further fine-tune the model using ground truth 2D boxes from the training set of SUN-RGBD before applying the model only using the color channels. The fine-tuning lasts for 4K iterations, with the learning rate reduced by 10× at 3K-th iteration starting from 0.01. The batch size, weight decay, and momentum are set as 8, 1e-4, and 0.9, respectively. Two data augmentation techniques are used: 1) standard left-right flipping; and 2) scale augmentation by randomly sample the shorter side of the input image from [480,600]. The resulting detector achieves a mAP (at overlap 0.5) of 58.5 on val set.

Note that we specifically choose *not* to use the most advanced 2D detectors (*e.g.* based on ResNet-152 [1]) just for the sake of performance improvement. As our experimental results shown in the main paper, even with this simple baseline Faster R-CNN, we can already see significant boost thanks to the design of IMVOTENET.

2D boxes. To infer 2D boxes using the detector, we first resize the input image to a shorter side of 600 before feeding into the model. Then top 100 detection boxes across all classes for an image is aggregated. We further reduce the number of 2D boxes per image by filtering out any detection with a confidence score below 0.1. Two things to note about the 2D boxes used while training IMVOTENET: 1) we could also train with ground truth 2D boxes, however we empirically found that including them for training hurts performance, likely due to the different detection statistics at test time; 2) as the pre-training for the 2D detector is also performed on the same training set, it generally gives better detection results on SUN RGB-D train set images, to reduce the effect of over-fitting, we randomly dropped 2D boxes with a probability of 0.5.

Alternative semantic cues. Other than the default semantic cue to represent each 2D box region as the one-hot classification score vector (the detected class has the value of the confidence score from the detector, all other locations have zeros), we further experimented with dense `RoI` features extracted from that region. Two variants are reported in the paper, with the 1024-dim one being the output from the last FC layer *before* region classification and regression. For the 64-dim one, we insert an additional FC layer before the final output layers so that region information is compressed into the 64-dim vector. The added layer is pre-trained with the 2D detector (resulting in a val mAP of 57.9) and fixed when training IMVOTENET.

Alternative texture cues. The default texture cue is the raw `RGB` values (normalized to [-1,1]). Besides this simple texture cue, we also experimented more advanced per-pixel features. One handy feature that preserves such spatial information is the feature maps P_k from FPN that fuse top-down and lateral connections [2]. Here k is the index to the layers in the feature pyramid, which also designates the feature strides and size. For example, P_2 has a stride of 2^2=4 for both height and width; and a spatial size of roughly $1/16$ of the input image[1]. For P_3 the strides are 2^3=8. All feature maps have a channel size of 256, which becomes the input dimension when used as texture cues for IMVOTENET.

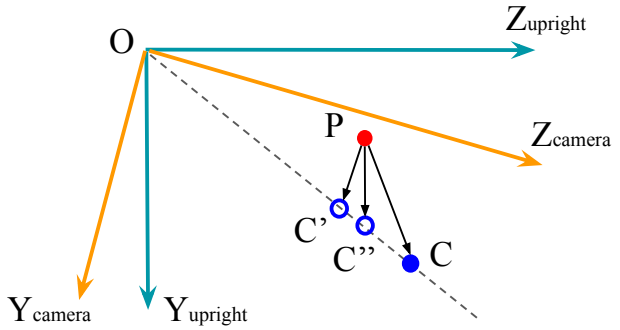


Figure 1. **Image vote lifting with camera extrinsic.** Here we show surface point P and object center C in two coordinate systems: camera coordinate and upright coordinate (OY is along gravitational direction). $\overrightarrow{PC}$ is the true 3D vote. $\overrightarrow{PC'}$ is the pseudo vote as calculated in the main paper and $\overrightarrow{PC''}$ is the transformed pseudo vote finally used in feature fusion.

B.3. Image Votes Lifting

In the main paper we derived the lifting process to transform a 2D image vote to a 3D pseudo vote *without* considering the camera extrinsic. As the point cloud sampled from depth image points is transformed to the *upright coordinate* before feeding to the point cloud network (through camera extrinsic R as a rotational matrix), the 3D pseudo vote also needs to be transformed to the same coordinate.

Fig. 1 shows the surface point P, object center C and the end point of the pseudo vote C'. Since the point cloud is in the upright coordinate, the point cloud deep net can only estimate the depth displacement of P and C along the Z_{upright} direction (it cannot estimate the depth displacement along the Z_{camera} direction as the rotational angles from camera to upright coordinate are unknown to the network). Therefore, we need to calculate a new pseudo vote $\overrightarrow{PC''}$ where C'' is on the ray OC and PC'' is perpendicular to the OZ_{upright}.

To calculate the C'' we need to firstly transform P and C' to the upright coordinate. Then assume P=(x_p, y_p, z_p) and C'=$(x_{c'}, y_{c'}, z_{c'})$ in the upright coordinate, we can compute:

$$C'' = (z_p \frac{x_{c'}}{z_{c'}}, z_p \frac{y_{c'}}{y_{c'}}, z_p). \quad (1)$$

[1]Note that different from 2D box detection, we feed the images directly to the model *without* resizing to shorter-side 600 to compute FPN features.

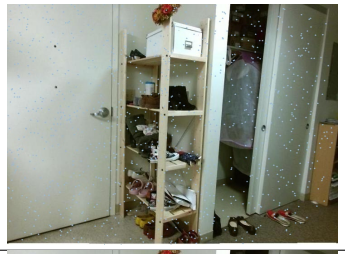
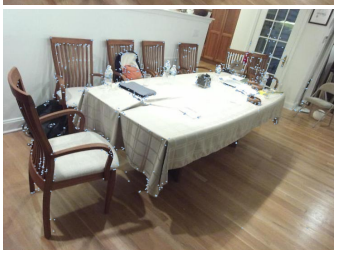
point cloud settings		images from SUN RGB-D [8] train set		
sampling method	# points	1	2	3
random uniform	20k			
	5k			
	1k			
ORB [7]	5k			
	1k			

Table 1. **Sparse point cloud visualization.** We show projected point clouds on three SUN RGB-D images and compare point density and distribution among random sampling (to $20k$, $5k$ and $1k$ points) and ORB key points based sampling (to $5k$ and $1k$ points). For ORB key point sampling, we firstly detect ORB key points in the RGB images with an ORB key point detector and then keep 3D points that are projected near those key points. Best viewed in color with zoom in.

C. Visualization of Sparse Points

In the Sec. 4.4 of the main paper we showed how image information and ImVoteNet model can be specially helpful in detections with sparse point clouds. Here in Table. 1 we visualize the sampled sparse point clouds on three example SUN RGB-D images. We project the sampled points to the RGB images to show their distribution and density. We see that the $20k$ points in the first row have a dense and uniform coverage of the entire scene. After randomly subsampling the points to $5k$ and $1k$ points in the second and third rows respectively, we see the coverage is much more sparse but still uniform. In contrast the ORB key point based sampling (the last two rows) results in very uneven distribu-

tion of points where they are clustered around corners and edges. The non-uniformity and low coverage of ORB key points makes it especially difficult to recognize objects in point cloud only. That's also where our IMVOTENET model showed the most significant improvement upon VOTENET.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2

[2] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1, 2

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2

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

[5] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 1

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1

[7] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, 2011. 3

[8] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015. 1, 3