Canonical Voting: Towards Robust Oriented Bounding Box Detection in 3D Scenes

Yang You, Zelin Ye, Yujing Lou, Chengkun Li, Yong-Lu Li, Lizhuang Ma, Weiming Wang*, Cewu Lu*
Shanghai Jiao Tong University, China
{qq456cvb, h_e_r_o, louyujing, sjtulck, yonglu_li, ma-lz, wangweiming, lucewu}@sjtu.edu.cn

Abstract

3D object detection has attracted much attention thanks to the advances in sensors and deep learning methods for point clouds. Current state-of-the-art methods like VoteNet regress direct offset towards object centers and box orientations with an additional Multi-Layer-Perceptron network. Both their offset and orientation predictions are not accurate due to the fundamental difficulty in rotation classification. In the work, we disentangle the direct offset into Local Canonical Coordinates (LCC), box scales and box orientations. Only LCC and box scales are regressed, while box orientations are generated by a canonical voting scheme. Finally, an LCC-aware back-projection checking algorithm iteratively cuts out bounding boxes from the generated vote maps, with the elimination of false positives. Our model achieves state-of-the-art performance on three standard real-world benchmarks: ScanNet, SceneNN and SUN RGB-D. Our code is available on https://github.com/qq456cvb/CanonicalVoting.

1. Introduction

With the use of depth cameras and Lidar sensors, 3D object detection is becoming more and more important for real-world scene understanding. Recently, with advances in deep networks for point clouds, several methods [16, 20, 22, 26] have shown state-of-the-art 3D detection results. Among them, the recently proposed VoteNet [16] showed remarkable improvement over previous methods on 3D oriented bounding box detection (rotation around gravity axis).

VoteNet passes the input point cloud through a backbone network and then samples a set of seed points, which generate center votes. These votes are offset, targeted to reach object centers. After that, vote clusters are aggregated through a learned module to generate box orientations and scales. VoteNet resembles with traditional Hough voting in that bounding box centers are proposed in those peaks of votes. However, we find that neither box orientations nor offsets towards object centers are accurately predicted for most points, even with modern 3D sparse convolution techniques [3]. The absolute error of point-wise predicted offsets and box orientations is not even better than random guess in many cases, as shown in Table 1. Though VoteNet proposes a two-stage pipeline and refines the center and orientation in the later stage, the accumulated error cannot be eliminated.

To handle this problem, we disentangle direct offset towards object centers into the following three parts: Local Canonical Coordinates (LCC), box scales and box orientations. First, we estimate Local Canonical Coordinates (LCC) and box scales instead of regressing orientations.
In LCC, all objects are consistently aligned and centered. In comparison to conventional orientation regression like voteNet, regression of LCC is generally easier because points belonging to the same part of an object are mapped to similar LCCs, no matter how the object rotates. Based on this, our model only needs to solve a task that is similar to part segmentation, for which we have seen great success in recent years [11, 18, 25]. Experiments show that our LCC predictions are far more accurate than direct offset. Therefore, these canonical votes could be directly used without any post-processing step like clustering. Similar ideas on regressing local coordinates have also been explored by previous methods like NOCS [24]. However, it requires an additional Mask-RCNN to do instance segmentation, and then finds a closed-form solution of translation, rotation and scale. This closed-form solution only exists for a single object. If there are multiple instances, no global closed-form solution exists as far as we know.

To solve this problem, we design a canonical voting algorithm to find possible object orientations and centers in Euclidean space. Object bounding boxes are proposed by looking at those locations with high votes. However, there will be some votes that accidentally accumulate as false positives. In order to eliminate them, it is crucial to project proposed object coordinates back into canonical space and compare them with LCC predictions. We call this step as LCC checking with back projection.

We evaluate our approach on two challenging large-scale 3D scan datasets: ScanNet [4] and SceneNN [9], and one smaller indoor RGB-D dataset: SUN RGB-D [21]. Our approach achieves state-of-the-art performance on ScanNet and SceneNN, with an absolute advance of 9.6 and 5.1 mAP, respectively. It also shows superior performance on SUN RGB-D benchmark. In addition, our experiments show that LCC regression and canonical voting are more robust over direct offset and orientation regression on detecting occluded objects.

To summarize, our contributions are:

- Bypassing orientation regression difficulties through Local Canonical Coordinates and Canonical Voting.
- Devising a back projection validation module to eliminate false positives, achieving high average precision.
- State-of-the-art performance on three 3D bounding box detection benchmarks.

2. Related Work

2.1. 3D Object Detection

There are many previous methods that are able to predict 3D bounding boxes of objects. COG [19] proposes to detect object by capturing contextual relationships among categories and layout. PointPillars [13] utilizes PointNets [18] to learn a representation of point clouds organized in vertical columns (pillars). PointRCNN [20] leverages two stages to do bottom-up 3D proposal generation and further refine these proposals. Frustum PointNet [17] first detects 2D bounding boxes in images and then back projects them into 3D space in order to get 3D bounding boxes. VoteNet [16] and ImVoteNet [15] both utilizes a clustering and voting scheme, though their direct offset predictions are not accurate and need to be refined with another MLP network. PointFusion [26] predicts multiple 3D box hypotheses and their confidences, using the input 3D points as spatial anchors. GSPN [28] generate proposals by reconstructing shapes from noisy observations in a scene. There are also a bunch of instance segmentation methods [6, 12, 27] that predict instances without oriented bounding boxes.

2.2. Voting Methods

Recently, there are several methods that combined deep learning and Hough Voting procedure, for various tasks. VoteNet [16] and ImVoteNet [15] use a PointNet++ [18] backbone and generate votes for each seed point. These votes are then clustered in Euclidean space to form proposals, which are fed into another MLP network to give final detections. PVNet [14] regresses pixel-wise unit vectors pointing to the predefined keypoints and solves a Perspective-n-Point (PnP) problem for pose estimation in RGB images. Zhao et al. [30] parameterizes lines with slopes and biases, and perform Hough transforms to translate deep representations into the parametric domain, in order to detect semantic lines in 2D images. In addition, there are several methods based on point-pair features [5, 7] to do a general Hough transform on object 6D poses. They need to sample a quadratic number of point pairs, making them extremely slow in large scenes.

3. Method

3.1. Overview

Figure 2 illustrates our detection pipeline. It can be split into three stages: first, we regress Local Canonical Coordinates, scales and objectness for each scene point, and we explain why it works in comparison to conventional direct offset regression [16]; then a canonical voting algorithm is proposed to generate a vote map on 3D grids; finally, an
We follow previous works [16] and only consider the heading where \( p \) current 6D pose estimation benchmarks, with more details be applied to full 3D rotations, showing an advancement on angle around the gravity axis. Nonetheless, our model can

\[
\begin{align*}
\Psi & = \Psi_{s, \alpha, t}(\tilde{p}) \\
& = \text{diag}(s) \cdot R_y(\alpha) \cdot \tilde{p} + t \\
& = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \cdot \tilde{p} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix},
\end{align*}
\]

where \( p \in \mathbb{R}^3 \) is world coordinates and \( \tilde{p} \in \mathbb{R}^3 \) is Local Canonical Coordinates (LCC). \( s = [s_x, s_y, s_z]^T \) are bounding box scales, \( \alpha \) is the heading angle of the bounding box around gravity axis, which is \( y \)-axis in our coordinate system. We follow previous works [16] and only consider the heading angle around the gravity axis. Nonetheless, our model can be applied to full 3D rotations, showing an advancement on current 6D pose estimation benchmarks, with more details

LCC back projection checking module is leveraged to progressively eliminate false positives and generate bounding boxes. To make it easier to understand, we will first describe the procedure for single-class object prediction and then discuss how it can be extended to multiple classes.

### 3.2. Regressing Local Canonical Coordinates

Inspired by NOCS [24], we propose to regress Local Canonical Coordinates. Local Canonical Coordinates (LCC), similar to NOCS, is defined as a 3D space contained within a unit cube i.e., \( \{x, y, z\} \in [-1, 1] \). In LCCs, all the models are consistently aligned and centered. Figure 3 shows an example of LCCs for two chairs.

For each object bounding box, within which LCCs are linked to world coordinates by the bounding box parameters:

\[
L_{reg} = \sum_{j=1}^{\{B_j\}} \sum_{i=1}^{N} \left( \|s^*_j - s_i\| + \|\Psi^{-1}_{s, \alpha_j^* , t_j^*}(\tilde{p}_i) - \tilde{p}_i\| \right) \cdot \mathbb{I}(\tilde{p}_i \text{ on object } j \text{'s surface}),
\]

where \( s^*_j, \alpha_j^*, t_j^* \) are ground truth scale, heading angle and translation parameters of bounding box \( B_j \), respectively. \( \mathbb{I}(\tilde{p}_i \text{ on object } j \text{'s surface}) \) indicates whether the point \( p_i \) is on the surface of the object \( j \) with bounding box \( B_j \).

It is worth mentioning that our LCC representation is invariant under rotations, while direct offsets are not. We do not predict rotations \( \alpha_i \) and object centers \( t_i \), as these two parameters will be generated from the canonical voting stage described in the next section.

**Why Local Canonical Coordinates?** At a first glance, it seems that LCCs are indirect and hard to predict. However, this is not the case. Take a 2D image with rotations as an example as shown in Figure 4, and suppose we would like to output direct offsets/LCCs for each pixel with 2D convolutional networks. In the left of Figure 4, we see how direct

```
Point Cloud with RGB colors: N x 3

U-Net with Sparse Convolution

Local Canonical Coordinates \( \{\tilde{p}\} \in \mathbb{R}^{N \times 3} \)

Canonical Voting with Objectness

Bounding Box Scales \( \{s\} \in \mathbb{R}^{N \times 3} \)

Objectness \( \{o\} \in \mathbb{R}^{N \times 1} \)

3D Vote Mip \n
G_{\alpha, t} \in \mathbb{R}^{W \times D \times W}

LCC Checking with Back Projection

Bounding Box Predictions
```

Figure 2. **Our model pipeline.** We first regress Local Canonical Coordinates (LCC), bounding box scales and objectness for each point; then an objectness weighted canonical voting algorithm is conducted to generate votes on 3D grids; finally, an LCC checking module with back projection validation is leveraged to progressively generate bounding boxes from the 3D vote map, which is shown in bird’s-eye view.
offset regression works: each local pattern is mapped into its corresponding direction. When the image rotates, different parts of the duck are mapped to the same output offset (indicated by the same color), since direct offset does not “go with” rotation. This makes it tough to identify different offset angles and box scales, respectively. Details are given in Algorithm 1 and this step is visualized in Figure 5.

To accumulate votes, we discretize continuous Euclidean object symmetry Our LCC representation would result in large errors for symmetric objects (e.g. tables, trash bins) as it is defined for a single orientation. To resolve this, we follow the common practice \[8, 23, 24\] by using a variant of the number of points, and it generally runs within 30ms.

3.3. Canonical Voting with Objectness

Next, we propose a canonical voting algorithm that produces a vote map indicating the likelihood of the existence of any object. In order to filter out those votes from points that do not belong to any objects, an additional objectness score \( o_i \in [0, 1] \) is predicted for each point, optimized by Cross Entropy loss with ground-truth objectness \( o_i^* \in \{0, 1\} \).

Object Symmetry Our LCC representation would result in large errors for symmetric objects (e.g. tables, trash bins) as it is defined for a single orientation. To resolve this, we follow the common practice \[8, 23, 24\] by using a variant of loss function. We calculate the loss \( L_{\text{reg}} \) for all symmetric reflections of an object, and then take the minimum of them.

Once we have \( s_i, \bar{p}_i, o_i \), every point votes to its corresponding bounding box center, for every possible orientation. To accumulate votes, we discretize continuous Euclidean space into grids \( G_{\text{obj}} \in \mathbb{R}^{H \times D \times W} \), where \( H, D, W \) is determined by a predefined grid interval \( \tau \) and the extent of input point clouds. Two additional grids \( G_{\text{rot}} \in \mathbb{R}^{H \times D \times W} \), \( G_{\text{scale}} \in \mathbb{R}^{H \times D \times W \times 3} \) are leveraged to capture voted heading angles and box scales, respectively. Details are given in Algorithm 1 and this step is visualized in Figure 5.

In line 4 of Algorithm 1, we generate possible point offset to object center without considering rotations. Since every object is considered to be rotated along gravity direction, we rotate the offset for every possible rotation in line 6 up to a predefined resolution \( K \). In line 7, possible object centers are found by inverting Equation 1. Then a vote will be accumulated in a predefined grid through trilinear interpolation. Finally, the voted heading angle map and scale map are weighted by objectness in line 14. 120 possible rotations are tested for each point, and the algorithm is highly paralleled and implemented on GPU. This step is linear in the number of points, and it generally runs within 30ms.

Every Point is a First-class Citizen Every single point participates in the canonical voting process in our model. As a consequence, our model is more robust than previous methods on detecting occluded objects, as each point would generate a vote. In contrast, previous two-stage methods like VoteNet generate proposals by clustering or sub-sampling, where occluded objects are more likely to be missed.

3.4. Iterative Bounding Box Generation

LCC Checking with Back Projection Because of the exhaustive orientation search in canonical voting process, there will be some false peaks in the resulting vote map \( G_{\text{obj}} \). As shown in Figure 6. In order to eliminate them, we leverage an LCC checking process with back projection. Specifically, we first generate a bounding box candidate according to the peak of \( G_{\text{obj}} \), \( G_{\text{rot}} \) and \( G_{\text{scale}} \). Then, we back project original points into Local Canonical Coordinates \( \bar{p} \) according to the current candidate bounding box. Finally, we check whether the projected LCCs \( \bar{p} \) are consistent with dense LCC predictions \( \bar{p} \) of the network. If the candidate box is placed wrongly, then its orientation will be inaccurate and the projected LCCs will be inconsistent with those of network predictions. This is described in Algorithm 2.

Algorithm 1 Canonical voting process.

1. Input: For each point \( i = 1, \ldots, N \), scale \( s_i \), LCC \( \bar{p}_i \), objectness \( o_i \), world coordinate \( p_i \).
2. Output: Accumulated votes for objectness \( G_{\text{obj}} \), rotation \( G_{\text{rot}} \) and scale \( G_{\text{scale}} \).
3. for \( i = 1, \ldots, N \) do
4. \( v_i = s_i * \bar{p}_i \), where * is element-wise product.
5. for \( j = 1, \ldots, K \) do
6. Find possible orientation \( r_j = \frac{j}{K} * 2 \pi \).
7. Find possible center \( \bar{p}_i = p_i - R_{\mu}(r_j) * v_i \).
8. Find \( \bar{p}_i \)'s \( 2^4 = 8 \) discrete grid neighborhoods \( N \).
9. Add \( o_i \) to \( N \) on \( G_{\text{obj}} \) with trilinear interpolation.
10. Add \( o_i \cdot r_j \) to \( N \) on \( G_{\text{rot}} \) with trilinear interpolation.
11. Add \( o_i \cdot s_i \) to \( N \) on \( G_{\text{scale}} \) with trilinear interpolation.
12. end for
13. end for
14. Normalize rotation and scale by element-wise division:
   \[ G_{\text{rot}} = \frac{G_{\text{rot}}}{G_{\text{obj}}}, G_{\text{scale}} = \frac{G_{\text{scale}}}{G_{\text{obj}}}. \]
Afterwards, two additional validation steps are leveraged to filter out those false positives. The first is to check if there are enough positive points (with probability $\beta$) within the bounding box, and the other is to check whether the back projected LCCs $\tilde{p}'$ are consistent with network LCC predictions.

In Algorithm 2, from line 4 to line 6, a bounding box center is proposed greedily from the peaks in $G_{obj}$, which is repeated until the vote count is below some threshold $\delta$. Then we read the scale and heading angle at the corresponding location of $G_{scale}$ and $G_{rot}$, and the world coordinate of bounding box center is retrieved by converting discrete grid index to continuous coordinate.

Next, we back project all the points back into LCCs $\tilde{p}'$ according to the current candidate bounding box in line 13. If $\tilde{p}'$ is within the proposed bounding box, we set its vote count to 0 in $G_{obj}$, so that it will never be detected again. Afterwards, two additional validation steps are leveraged to filter out those false positives. The first is to check if there are enough positive points (with probability $\beta$) within the bounding box, and the other is to check whether the back projected LCCs $\tilde{p}'$ are consistent with network LCC predictions.

**Algorithm 2** Bounding Box Generation with LCC Back Projection Checking.

1. **Input**: Accumulated votes for objectness $G_{obj}$, rotation $G_{rot}$ and scale $G_{scale}$; for $i = 1, \ldots, N$, scale $s_i$, LCC $p_i$, objectness $o_i$, point $p_i$.
2. **Output**: Predicted bounding box set $B$.
3. Initialize $B = \{\}$.
4. while True do
5. \[ [h, d, w]^T = \text{argmax}(G_{obj}). \]
6. if $G_{obj}[h, d, w] < \delta$ then
7. break
8. end if
9. Convert grid index $[h, d, w]^T$ to world coordinate $t$.
10. $s := G_{scale}[h, d, w]$, $\alpha := G_{rot}[h, d, w]$.
11. $err := 0$, $pos := 0$, $cnt := 0$, $o_{sum} := 0$.
12. for $i = 1, \ldots, N$ do
13. Let $\tilde{p}'_i = \Psi^{-1}_{s,\alpha,t}(p_i)$.
14. if $-1 < \tilde{p}'_i < 1$ then
15. $cnt := cnt + 1$.
16. Convert world coordinate $p_i$ to grid index $[h_i, d_i, w_i]^T$.
17. $G_{obj}[h_i, d_i, w_i] := 0$.
18. if $o_i > 0.3$ then
20. $o_{sum} := o_{sum} + o_i$.
21. $err := err + o_i \cdot ||\tilde{p}'_i - \tilde{p}'||_2$.
22. end if
23. end if
24. end for
25. if $pos > \beta \cdot cnt$ and $\frac{o_{sum}}{err} < \gamma$ then
26. Add bounding box $\{s, \alpha, t\}$ to $B$.
27. end if
28. end while

$\tilde{p}$ in line 21. This process is visualized in Figure 6.

### 3.5. Extension to Multiple Classes

Extension to multiple classes is quite straightforward. In addition to scale $s_i$, LCC $p_i$, and objectness $o_i$, a per-point class score $c_i$ is predicted, trained with cross entropy loss.
Figure 6. LCC Checking with Back Projection Module. LCC coordinates are visualized as RGB colors. At iteration 3, three true positive (in green) boxes for (partial) chairs are generated. When their coordinates are back projected to LCCs, they are consistent with network predictions. At iteration 4, due to the exhaustive orientation search in the canonical voting process, there is a false positive box (in red). It is eliminated by comparing the LCCs back-projected by the candidate box and those from network predictions.

The bounding box generation algorithm is almost identical to Algorithm 2, except that we determine the class with the majority votes of points within the bounding box.

4. Experiments

4.1. Experimental Setup

Dataset ScanNet is a richly annotated dataset of 3D reconstructed meshes of indoor scenes. Since original ScanNet does not provide amodal or oriented bounding box annotation, we use the oriented bounding box labels provided by Scan2CAD [1], where they annotate 14K+ oriented object bounding boxes for all 1506 scenes in ScanNet. Scan2CAD contains 9 common categories with rotation-aligned models.

SceneNN [9] is an RGB-D scene dataset consisting of 100 scenes. We follow [10] to evaluate on 76 scenes with orientated bounding box annotations. We directly transfer the model trained on ScanNet in order to see its generalization ability. Categories that appear in both SceneNN and ScanNet are reported.

SUN RGB-D [21] is a single-view RGB-D dataset for scene understanding. It consists of 5K RGB-D training images annotated with oriented 3D bounding boxes for 37 object categories. We follow a standard evaluation protocol and report performance on the 10 most common categories.

Compared Methods We compare with a number of prior methods on 3D bounding box estimation: PointFusion [26], GSPN [28], F-PointNet [17], VoteNet [16], H3DNet [29] and BRNet [2]. Note that both PointFusion and F-PointNet require additional 2D corresponding images, which are not annotated by [10] on SceneNN. Therefore, only GSPN, VoteNet and H3DNet results are reported on SceneNN. DSS [22] and COG [19] are also reported on SUN RGB-D.

Evaluation Metrics Evaluation metric is average precision with 3D bounding box IoU under various thresholds. We also find that our network performs better when trained separately for each category, due to the similarity of LCCs within each category. Therefore, we report results under two settings: separate training for each category and joint training for all categories.

Implementation Details Our detection pipeline takes an RGB point cloud as input. To augment data, we follow the same strategy as [1] so that point clouds are rotated around gravity direction four times (90° increments with 20° random jitter). We follow the common practice to discretize point clouds into voxels with grid size 0.03 and adopt 3D Minkowski convolution networks with a similar structure as ResNet34. The output size is 7NC + 1, consisting 3NC scales, 3NC LCC coordinates, NC + 1 class scores including backgrounds. We train the network from scratch with Adam optimizer, batch size 3 with an initial learning rate of 0.001. We set β = 0.2, γ = 0.3, δ = 60 in Algorithm 2, and use heading angle directions \((\cos(\alpha), \sin(\alpha))\) rather than raw heading angle \(\alpha\). Non-Maximum-Suppression with threshold 0.3 is leveraged on generated bounding boxes.

SUN RGB-D dataset does not provide segmentation labels but bounding boxes for each individual object. In order to train the objectness score, we consider all the points in a slightly enlarged (i.e., 1.2×) bounding box belong to this object. Furthermore, because of the limited data (i.e., single RGB-D frames compared to multi-view reconstructed scenes in ScanNet), the generated vote map is not accurate enough. To solve this problem, instead of a deterministic bounding box generation procedure, we sample multiple box candidates with probability proportional to the vote map, and then leverage a refinement module to further refine these bounding boxes. For more details of this architecture, we refer the reader to the supplementary.

4.2. Evaluation on ScanNet and SceneNN

We first report performance of our model on ScanNet validation set under two training settings: separately trained for each category and trained jointly for all categories. The results are listed in Table 2. Our model outperforms previous methods under both training settings on Scan2CAD benchmark. It has an improvement of 3.3 mAP when trained.
jointly and 9.8 mAP when trained separately for each category. Our model outperforms baseline methods on Trashbin by 28.1 AP, Sofa by 15.8 AP, which is a large margin. Our model achieves even better performance when trained separately because it could benefit from Local Canonical Coordinate definitions that are similar and consistent in each category (Figure 3). In contrast, previous methods do not gain much or even perform worse when training separately.

Then we report our model’s performance by directly evaluating on SceneNN dataset with network trained on ScanNet. The results are listed in Table 3. Our method achieves highest mAP with good generalization ability. Some qualitative results in cluttered scenes are demonstrated in Figure 8.

### 4.3. Evaluation on SUN RGB-D

Next, we evaluate on SUN RGB-D dataset, which contains single RGB-D frames instead of complete 3D scans. Quantitative results are listed in Table 4, where our method also achieves state-of-the-art performance. All the compared methods are trained and evaluated jointly. With the help of our proposed canonical voting module, we are able to get higher mAP. This demonstrates that the voting center sampled from our vote map is more accurate than that of a simple Farthest Point Sampling (FPS) module.

### 4.4. Detailed Analysis

**Robustness on Occluded/Partial Objects** As we explained in Section 3.3, each individual point is a first-class citizen, which has an advantage over previous methods on detecting occluded or partial objects, as these objects usually contain fewer points and any clustering or sub-sampling step would miss them by chance. To quantitatively illustrate this,
<table>
<thead>
<tr>
<th>Method</th>
<th>mAP25</th>
<th>mAP50</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSS [22]</td>
<td>42.1</td>
<td>-</td>
</tr>
<tr>
<td>COG [19]</td>
<td>47.6</td>
<td>-</td>
</tr>
<tr>
<td>F-PointNet [17]</td>
<td>54.0</td>
<td></td>
</tr>
<tr>
<td>VoteNet [16]</td>
<td>57.7</td>
<td>32.9</td>
</tr>
<tr>
<td>H3DNet [29]</td>
<td>60.1</td>
<td>39.0</td>
</tr>
<tr>
<td>BRNet [2]</td>
<td>61.1</td>
<td>43.7</td>
</tr>
<tr>
<td>Ours</td>
<td>61.3</td>
<td>44.3</td>
</tr>
</tbody>
</table>

Table 4. Quantitative comparison on SUN RGB-D.

Table 5. Ablation study results, evaluated by mAP50, with separate models. Results reported on ScanNet.

<table>
<thead>
<tr>
<th>Method</th>
<th>ScanNet</th>
<th>SceneNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (direct voting)</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Ours (no back projection check)</td>
<td>8.1</td>
<td>10.0</td>
</tr>
<tr>
<td>Ours (no objectness)</td>
<td>12.2</td>
<td>21.4</td>
</tr>
<tr>
<td>Ours (final)</td>
<td>21.7</td>
<td>26.8</td>
</tr>
</tbody>
</table>

Table 5. Ablation study results, evaluated by mAP50, with separate models. Results reported on ScanNet.

Decoupled Orientation/Translation Error and Running Time The running time and decoupled orientation/translation error of all compared methods are listed in Table 6. Our method outperforms previous methods by a large margin in terms of orientation error.

<table>
<thead>
<tr>
<th>Method</th>
<th>Ori. Error(°)</th>
<th>Trans. Error(m)</th>
<th>Proc. Time(s)</th>
<th>mAP50</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet [26]</td>
<td>77.3</td>
<td>0.07</td>
<td>0.23</td>
<td>6.6</td>
</tr>
<tr>
<td>F-PointNet [17]</td>
<td>62.5</td>
<td>0.09</td>
<td>0.46</td>
<td>10.6</td>
</tr>
<tr>
<td>VoteNet [16]</td>
<td>76.2</td>
<td>0.05</td>
<td>0.43</td>
<td>10.3</td>
</tr>
<tr>
<td>H3DNet [29]</td>
<td>65.4</td>
<td>0.03</td>
<td>0.78</td>
<td>12.1</td>
</tr>
<tr>
<td>BRNet [2]</td>
<td>79.1</td>
<td>0.05</td>
<td>0.47</td>
<td>8.3</td>
</tr>
<tr>
<td>Ours</td>
<td>10.8</td>
<td>0.05</td>
<td>0.32</td>
<td>15.4</td>
</tr>
</tbody>
</table>

Table 6. Decoupled rotation/translation error (for mAP50) and running time analysis on ScanNet with joint models. Running time is reported on i9-7900X CPU with 1080Ti GPU.

5. Conclusion
In this work, we propose a new method for robust oriented bounding box detection in large scale 3D scenes. We regress Local Canonical Coordinates (LCC) instead of direct offsets. Bounding box orientations are generated by Canonical Voting. LCC checking with back projection is leveraged to eliminate false positives. Results show that our model achieves state-of-the-art performance.

6. Acknowledgements
This work was supported by the National Key Research and Development Project of China (No. 2021ZD0110700), the National Natural Science Foundation of China under Grant 51975350, Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), Shanghai Qi Zhi Institute, and SHEITC (2018-RGZN-02046). This work was also supported by the Shanghai AI development project (2020-RGZN-02006) and “cross research fund for translational medicine” of Shanghai Jiao Tong University (zh2018qnb17, zh2018qna37, YG2022ZD018).
References


