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, a 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 challenging large-scale datasets of real point cloud scans: ScanNet and SceneNN, with an absolute advance of **8.8** and **5.1** mAP, respectively. Code is available on github.*

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 regressed in 3D scenes, even with modern 3D sparse convolution techniques [2]. The absolute error of point-wise predicted offsets and box orientations is even *not better than random guess* in many cases, as shown in Table 1. Direct offset and orientation regression do not take rotational invariant patterns into consideration and fail to capture the complex relationship between rotated objects and their corresponding orientations.

To handle this problem, we disentangle direct offset towards object centers into the following three parts: Local Canonical Coordinates (LCC), box scales and box orien-

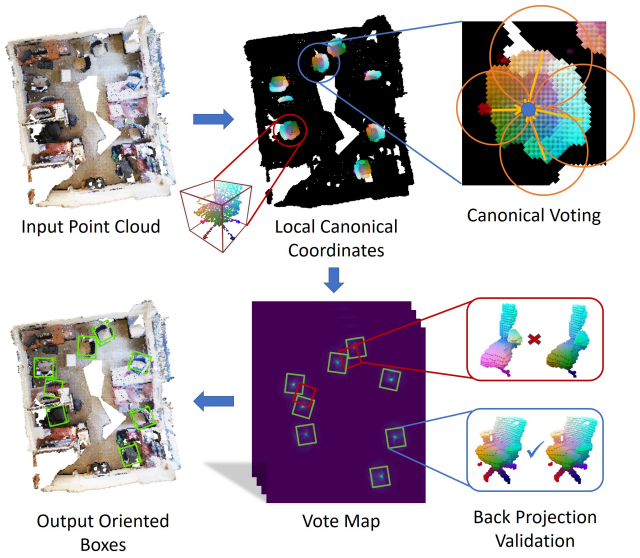

Figure 1: **We present a method that regresses Local Canonical Coordinates disentangled from orientations.** We leverage a canonical voting module to find possible orientations and object centers. Back projection validation is utilized to further eliminate false positives.

*Cewu Lu is the corresponding author.

	Offset Towards Center	Orientation Vector
Direct Regression	0.197	0.806
Random Guess	0.228	0.801

Table 1: **Mean absolute error of pointwise predicted offsets and orientations, evaluated on ScanNet.** We see that direct regression is only slightly better than random guess on offset predictions and worse on orientation predictions.

tations. 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 [25, 18, 11]. 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 [3] 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 **8.8** and **5.1** mAP, respectively. It also shows competitive 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 two challenging 3D point cloud detection benchmarks and competitve performance on one RGBD frame detection benchmark.

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 [12, 5, 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. Han et al. [6] parameterizes lines with slopes and biases, and perform Hough transform 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 [7, 4] to do a general Hough transform on object 6D poses. They need to sample 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

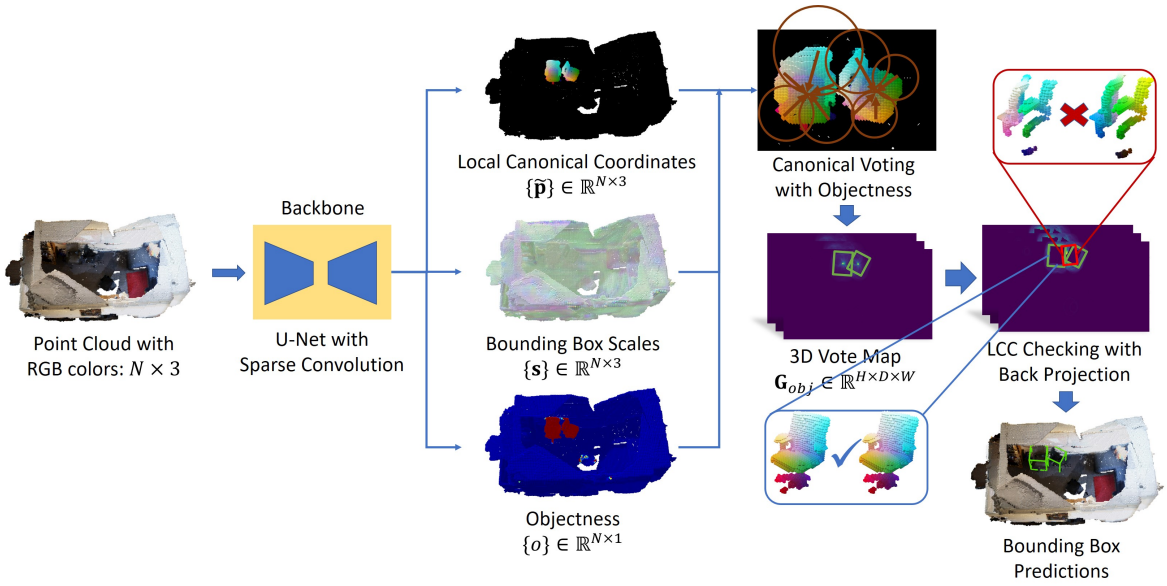

Figure 2: **Our model pipeline.** We first regress Local Canonical Coordinates (LCC), bounding box scales and objectness for each point; and then a objectness weighted canonical voting algorithm is conducted to generate votes on dense 3D grids; finally, a LCC checking module with back projection validation is leveraged to progressively generate bounding boxes from the 3D vote map. The votemap is illustrated in 2D bird's-eye view.

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, a LCC back projection checking module is leveraged to progressively eliminate false positives and generate bounding boxes from the vote map. 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:

$$\begin{aligned}\mathbf{p} &= \Psi_{\mathbf{s},\alpha,\mathbf{t}}(\tilde{\mathbf{p}}) \\ &= \text{diag}(\mathbf{s}) \cdot \text{R}_y(\alpha) \cdot \tilde{\mathbf{p}} + \mathbf{t} \\ &= \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \cdot \begin{bmatrix} cos(\alpha) & 0 & -sin(\alpha) \\ 0 & 1 & 0 \\ sin(\alpha) & 0 & cos(\alpha) \end{bmatrix} \cdot \tilde{\mathbf{p}} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix},\end{aligned} \tag{1}$$

where $\mathbf{p} \in \mathbb{R}^3$ is world coordinates and $\tilde{\mathbf{p}} \in \mathbb{R}^3$ is Local Canonical Coordinates (LCC). $\mathbf{s} = [s_x, s_y, s_z]^T$ are bounding box scales, α 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. $\mathbf{t} = [t_x, t_y, t_z]^T$ are bounding box centers. $\text{R}_y(\cdot)$ is an operator that generate a rotation matrix around y-axis given the heading angle.

Every point of an object in world coordinates can be transformed uniquely to LCC by Equation 1. In the following sections, we will denote Equation 1 as $\mathbf{p} = \Psi_{\mathbf{s},\alpha,\mathbf{t}}(\tilde{\mathbf{p}})$.

Then, we regress LCCs and bounding box scales for each point on objects. Specifically, given a point cloud $\{\mathbf{p}_i\}_{i=1}^N$, we point-wise predict $\mathbf{s}_i, \tilde{\mathbf{p}}_i$, with the following loss:

$$\begin{aligned}L_{reg} = \sum_{j=1}^{|\{B_j\}|} \sum_{i=1}^{N} (\|\mathbf{s}_j^* - \mathbf{s}_i\| + \|\Psi^{-1}_{\mathbf{s}_j^*,\alpha_j^*,\mathbf{t}_j^*}(\mathbf{p}_i) - \tilde{\mathbf{p}}_i\|) \\ \cdot \mathbb{1}(\mathbf{p}_i \text{ on object } j\text{'s surface}),\end{aligned} \tag{2}$$

where $\mathbf{s}_j^*, \alpha_j^*, \mathbf{t}_j^*$ are ground truth scale, heading angle and translation parameters of bounding box B_j, respectively. $\mathbf{p}_i$ on object j's surface indicates whether point $\mathbf{p}_i$ is on the surface of 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 α_i and object centers $\mathbf{t}_i$, 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 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 offsets based on such divergent input patterns. In contrast, we can see how LCC regression works from the right of Figure 4: no matter how the image rotates, patterns that belong to the same part (e.g. beak) are always mapped to the same LCC in canonical pose. This makes it easier to learn the relationship between inputs and outputs. The network only needs to classify different parts (with a few of rotated versions) of the object in order to output corresponding LCCs.

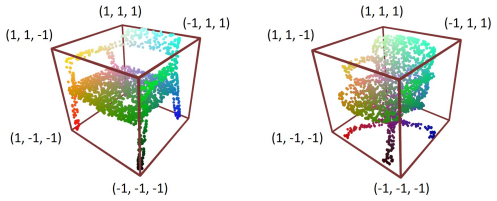

Figure 3: **The Local Canonical Coordinate (LCC) space.** RGB colors indicate the (x, y, z) positions in the LCC.

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

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 $o_i^* \in \{0, 1\}$. $o_i^* = 1$ if points are on any instance; $o_i^* = 0$ otherwise.

Once we have $\mathbf{s}_i, \tilde{\mathbf{p}}_i, o_i$, every point votes to its corresponding bounding box center, for every possible orientation. To accumulate votes, we discretize continuous Euclidean space into grids $\mathbf{G}_{obj} \in \mathbb{R}^{H \times D \times W}$, where H, D, W is determined by a predefined grid interval τ and the extent of input point clouds. Two additional grids $\mathbf{G}_{rot} \in \mathbb{R}^{H \times D \times W}$, $\mathbf{G}_{scale} \in \mathbb{R}^{H \times D \times W \times 3}$ 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.

Algorithm 1 Canonical voting process.

1: **Input**: For each point $i = 1, \ldots, N$, scale $\mathbf{s}_i$, LCC $\tilde{\mathbf{p}}_i$, objectness o_i, world coordinate $\mathbf{p}_i$.
2: **Output**: Accumulated votes for objectness $\mathbf{G}_{obj}$, rotation $\mathbf{G}_{rot}$ and scale $\mathbf{G}_{scale}$.
3: **for** $i = 1, \ldots, N$ **do**
4: $\quad \mathbf{v}_i = \mathbf{s}_i * \tilde{\mathbf{p}}_i$, where $*$ is element-wise product.
5: $\quad$ **for** $j = 1, \ldots, K$ **do**
6: $\quad\quad$ Find possible orientation $r_j = \frac{j}{K} * 2\pi$.
7: $\quad\quad$ Find possible center $\bar{\mathbf{p}}_i = \mathbf{p}_i - \mathbf{R}_y(r_j) \cdot \mathbf{v}_i$.
8: $\quad\quad$ Find $\bar{\mathbf{p}}_i$'s $2^3 = 8$ discrete grid neighborhoods $\mathcal{N}$.
9: $\quad\quad$ Add o_i to $\mathcal{N}$ on $\mathbf{G}_{obj}$ with trilinear interpolation.
10: $\quad\quad$ Add $o_i \cdot r_j$ to $\mathcal{N}$ on $\mathbf{G}_{rot}$ with trilinear interpolation.
11: $\quad\quad$ Add $o_i \cdot \mathbf{s_i}$ to $\mathcal{N}$ on $\mathbf{G}_{scale}$ with trilinear interpolation.
12: $\quad$ **end for**
13: **end for**
14: Normalize rotation and scale by element-wise division: $\mathbf{G}_{rot} = \frac{\mathbf{G}_{rot}}{\mathbf{G}_{obj}}, \mathbf{G}_{scale} = \frac{\mathbf{G}_{scale}}{\mathbf{G}_{obj}}$.

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 We can see that every single point would participate in the canonical voting process. As a consequence, our model is more robust than previous methods on detecting partial or occluded objects, as each point would generate a candidate vote. In contrast, previous methods like VoteNet generate proposals by clustering or sub-sampling, where occluded objects are more likely to be missed. This will be investigated in detail in our experiments.

3.4. LCC Checking with Back Projection for Bounding Box Generation

Next, we identify the peaks in vote map $\mathbf{G}_{obj}$ and generate bounding boxes. Importantly, potential false positives are eliminated by operating LCC checking with back projection.

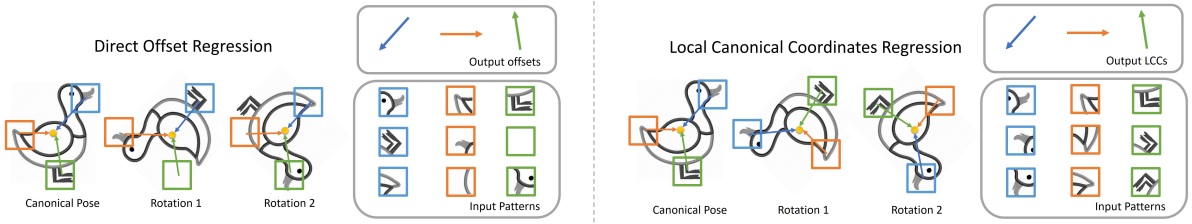

Figure 4: **An illustration on why LCC works better than direct offset regression. Left**: direct offset regression maps different parts of the object to the same output offsets. **Right**: LCC regression maps the same part to the same LCC in canonical pose, regardless of the rotation. Therefore, it is easier for a network to regress LCCs by doing a part segmentation task. Image from Clipart Library.

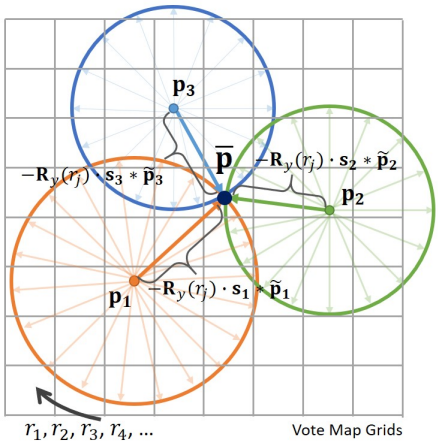

Figure 5: **Canonical Voting Process in Bird's-eye view.** For each point, we estimate its LCC $\tilde{\mathbf{p}}$ and bounding box scales $\mathbf{s}$. Then for each possible bounding box orientations $r_1, r_2, r_3, r_4, \ldots$, we generate a vote towards the box center $\mathbf{p}$ by subtracting rotated $\mathbf{s} * \tilde{\mathbf{p}}$ from its world coordinate $\mathbf{p}$. As last, these votes are accumulated in a predefined 3D grid by trilinear interpolation..

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 $\mathbf{G}_{obj}$, as shown in Figure 6. In order to eliminate them, we leverage a LCC checking process with back projection. Specifically, we first genereate a bounding box candidate according to the peak of $\mathbf{G}_{obj}$, $\mathbf{G}_{rot}$ and $\mathbf{G}_{scale}$. Then, we back project original points into Local Canonical Coordinates $\tilde{\mathbf{p}}'$ according to the current candidate bounding box. Points outside the candidate bounding box are discarded. Finally, we check whether the projected LCCs $\tilde{\mathbf{p}}'$ are consistent with dense LCC predictions $\tilde{\mathbf{p}}$ of the network. This step is effective and crucial and most false positives are eliminated in this step, reaching a high precision. 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. The full algorithm is demonstrated in Algorithm 2.

Algorithm 2 Bounding Box Generation with LCC Back Projection Checking.

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

In Algorithm 2, from line 4 to line 6, a bounding box center is proposed greedily from the peaks in $\mathbf{G}_{obj}$, which is repeated until the vote count is below some threshold δ. Then

we read the scale and heading angle at the corresponding location of $\mathbf{G}_{scale}$ and $\mathbf{G}_{rot}$, 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 $\tilde{\mathbf{p}}'$ according to the current candidate bounding box in line 13. If $\tilde{\mathbf{p}}'$ is within the proposed bounding box, we set its vote count to 0 in $\mathbf{G}_{obj}$, 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 β) within the bounding box, and the other is to check whether the back projected LCCs $\tilde{\mathbf{p}}'$ are consistent with network LCC predictions $\tilde{\mathbf{p}}$ in line 21. This process is visualized in Figure 6.

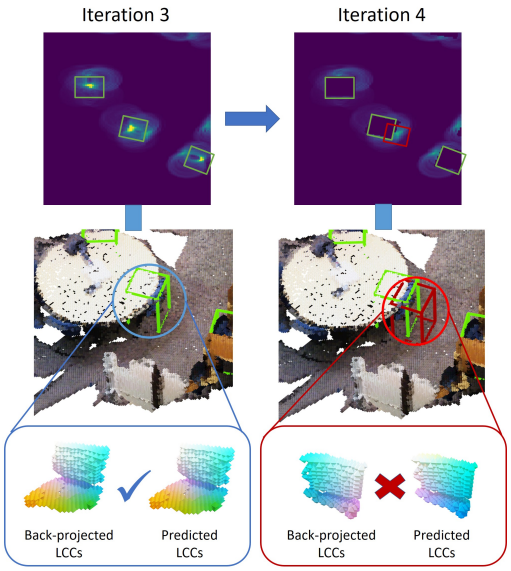

Figure 6: **LCC Checking with Back Projection Module.** Once the 3D vote map is proposed, we greedily find the peaks and generate candidate bounding boxes. LCC coordinates are visualized as RGB colors. At iteration 3, three true positive (in green) boxes for (partial) chairs are generated. When their 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 with the candidate bounding box and the LCCs from network predictions.

3.5. Extension to Multiple Classes

Extension to multiple classes is quite straight-forward. In addition to scale $\mathbf{s}_i$, LCC $\tilde{\mathbf{p}}_i$ and objectness o_i, a per-point class score c_i is predicted, trained with cross entropy loss. 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

In this section, we first evaluate on three object detection benchmarks: ScanNet [3], SceneNN [9] and SUN RGB-D [21] dataset. Then, we conduct ablation studies to verify our algorithm designs.

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. All scenes are reconstructed into triangle meshes and have per-vertex annotation. We follow [10] to evaluate on 76 scenes with orientated bounding box annotations. Since this is a small dataset, we directly transfer the model trained on ScanNet in order to see its generalization ability. All 76 scenes are used for evaluation. Categories that are common to 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 amodal oriented 3D bounding boxes for 37 object categories. To feed the data to our network, we firstly convert the depth images to point clouds using the provided camera parameters as VoteNet does. We follow a standard evaluation protocol and report performance on the 10 most common categories.

Compared Methods We compare with number of prior methods on 3D bounding box estimation: PointFusion [26], GSPN [28], F-PointNet [17], VoteNet [16] and H3DNet [29]. 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 dataset.

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 a 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. The hyper-parameters in Algorithm 2 are chosen with 5-fold cross-validation on training set: $\beta = 0.2, \gamma = 0.3, \delta = 60$. We use heading angle directions $(cos(\alpha), sin(\alpha))$ rather than raw heading angle α during canonical voting and bounding box generation. Non-Maximum-Suppression with threshold 0.3 is leveraged on generated bounding boxes.

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 **9.0** 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 for both datasets are demonstrated in Figure 8. We see that on both ScanNet and SceneNN, our method is able to detect accurate oriented bounding boxes in cluttered scenes.

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 gives competitive results, though not the state-of-the-art. There are several reasons why our method in inferior in this dataset. Firstly, there are no annotation on object symmetry, making our model confused about the LCC coordinates for those symmetric objects. Secondly, SUN RGB-D dataset does not provide instance segmentation labels for each individual point. Therefore, we regress LCCs for all the points within each object's bounding box, which may confuse our model on both LCC and objectness prediction because of background noise. It also makes multi-class joint training intractable for our method since the class label for each point is ambiguous when two bounding boxes intersect.

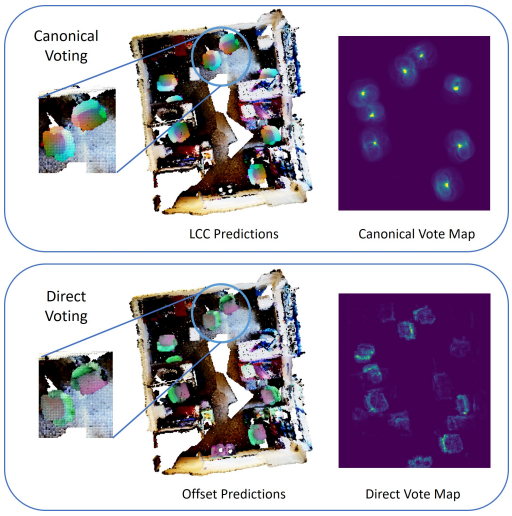

Figure 7: **LCC regression accumulates votes much better than direct offset regression.** LCC/offset predictions are visualized as RGB color tuples. In the top, we see that with LCCs and proposed voting process, object centers are aggregated with good locality. On the contrary, direct offsets are not distinguishable on xz-plane (perpendicular to gravity axis) and object centers fail to accumulate with voting.

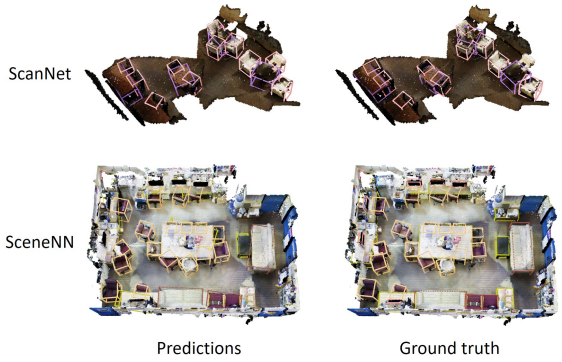

Figure 8: **Qualitative results of 3D oriented bounding box detection in ScanNet and SceneNN.**

4.4. Ablation Study

In this section, we present component analysis to validate our designs. Results are reported on both ScanNet and SceneNN with the model trained separately on each category.

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

	Trashbin	Bathtub	Bookshelf	Cabinet	Chair	Display	Sofa	Table & Desk	Others	mAP_{50}
PointFusion	**4.5**/-	6.2/-	4.6/-	5.3/-	25.1/-	0.6/-	3.8/-	3.1/-	6.6/-	6.6/-
GSPN	0.9/-	5.4/-	0.0/-	0.5/-	16.8/-	0.2/-	14.2/-	5.1/-	0.1/-	4.8/-
F-PointNet	0.0/2.3	7.9/9.2	3.8/6.9	12.8/14.7	39.3/34.1	0.0/**1.2**	16.9/15.6	9.9/11.7	4.4/**6.6**	10.6/11.4
VoteNet	2.3/5.2	7.4/**9.8**	0.0/0.1	1.9/4.4	52.1/71.0	**1.4**/0.1	2.5/0.5	17.0/10.2	**8.5**/6.4	10.3/11.9
H3DNet	2.4/1.5	**10.5**/3.2	1.1/0.5	9.8/4.4	23.7/28.7	0.2/0.0	23.7/3.2	**31.5**/5.8	6.0/3.6	12.1/5.7
Ours	0.5/**32.6**	0.0/0.1	**14.2/8.9**	**17.0/15.2**	**57.6/75.5**	0.1/0.0	**39.5/31.3**	3.2/**23.9**	6.4/0.3	**15.4/20.9**

Table 2: **3D oriented instance bounding box detection results on ScanNet val set, with Scan2CAD labels.** Results under both joint and separate training settings are reported, separated by slashes.

	Table & Desk	Chair	Cabinet	Sofa	Display	mAP_{50}
GSPN	8.4/-	34.6/-	0.2/-	9.8/-	0.0/-	10.6/-
VoteNet	**24.8**/27.3	65.2/78.0	1.5/1.0	15.4/2.4	**0.7**/0.0	**21.5**/21.7
H3DNet	12.1/7.4	29.4/38.2	0.8/0.7	10.5/**14.8**	0.0/0.0	10.6/12.2
Ours	11.9/**43.4**	**68.1/79.6**	**2.5/2.2**	**22.8**/8.8	0.0/0.0	21.1/**26.8**

Table 3: **3D oriented instance bounding box detection results on SceneNN.** Our model achieves the highest mAP_{50}.

	mAP_{25}	mAP_{50}
DSS	42.1/-	-/-
COG	47.6/-	-/-
F-PointNet	54.0/-	-/-
VoteNet	57.7/60.8	32.9/31.3
H3DNet	**60.1/61.4**	**39.0/32.4**
Ours	-/52.3	-/30.5

Table 4: **Quantitative comparison of various baselines on SUN RGB-D dataset.** Results for both joint and separate training settings are reported, separated by slashes.

contain fewer points and any clustering or sub-sampling step would miss them by chance. To quantitatively illustrate this, we define a *partial index* to be the number of points on an object, normalized by its bounding box volume. Intuitively, when an object has a small partial index, it is more likely to be occluded as fewer points are visible. We quantize these partial indexes into integers from 1 to 10, and plot the average recall for each discrete index, shown in Figure 9. Our method gives higher AR_{50} on severely occluded objects while VoteNet is better at detecting complete objects.

LCC + Canonical Voting vs. Direct Offset + Voting We compare with our method with a baseline that implements the direct offset prediction and naive voting algorithm. Qualitative results are demonstrated in Figure 7. Our model predicts LCCs accurately and these LCCs can be directly utilized to vote in Euclidean space. On the contrary, direct offset predictions are inaccurate and fail to accumulate in Euclidean space. Quantitative results are listed in Table 5.

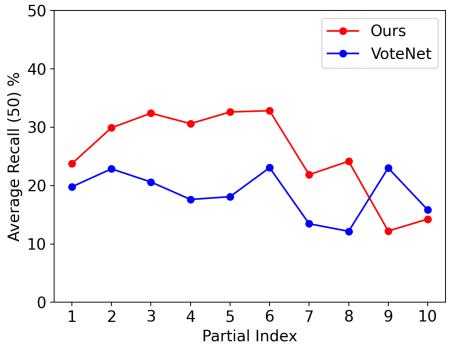

Figure 9: **Average Recall$_{50}$ comparison for partially occluded objects.** Our method achieves better performance on those partial/occluded objects.

How Does Back Projection Validation Help In Table 5, we show the effectiveness of our LCC checking module with back projection. In can be concluded that without back projection, mAP experiences a large drop on both ScanNet and SceneNN datasets. This back projection verification is critical for a high precision.

Effect of Objectness Objectness filters out many unlikely votes at the first stage. Here, we compare our objectness-weighted canonical voting algorithm and a baseline algorithm that treats every point a valid vote with weight 1. In can be seen that no objectness lowers mAP by 10.8 and 5.4 for ScanNet and SceneNN, respectively.

	ScanNet	SceneNN
Ours (final)	**20.9**	**26.8**
Ours (direct voting)	0.0	0.0
Ours (no back projection check)	7.9	10.0
Ours (no objectness)	12.0	21.4

Table 5: **Ablation study results, evaluated by mAP$_{50}$.**

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 on ScanNet and SceneNN with 8.8 and 5.1 mAP improvement respectively.

References

[1] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Niessner. Scan2cad: Learning cad model alignment in rgb-d scans. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 6

[2] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 1

[3] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 2, 6

[4] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 998–1005. Ieee, 2010. 2

[5] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2940–2949, 2020. 2

[6] Qi Han, Kai Zhao, Jun Xu, and Mingg-Ming Cheng. Deep hough transform for semantic line detection. *arXiv preprint arXiv:2003.04676*, 2020. 2

[7] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. Going further with point pair features. In *European conference on computer vision*, pages 834–848. Springer, 2016. 2

[8] Tomas Hodan, Daniel Barath, and Jiri Matas. Epos: estimating 6d pose of objects with symmetries. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11703–11712, 2020. 4

[9] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 92–101. IEEE, 2016. 2, 6

[10] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018. 6

[11] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2626–2635, 2018. 2

[12] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4867–4876, 2020. 2

[13] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 2

[14] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019. 2

[15] Charles R Qi, Xinlei Chen, Or Litany, and Leonidas J Guibas. Imvotenet: Boosting 3d object detection in point clouds with image votes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4404–4413, 2020. 2

[16] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9277–9286, 2019. 1, 2, 3, 6

[17] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018. 2, 6

[18] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 2

[19] Zhile Ren and Erik B Sudderth. Three-dimensional object detection and layout prediction using clouds of oriented gradients. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1525–1533, 2016. 2, 6

[20] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 1, 2

[21] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015. 2, 6

[22] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016. 1, 6

[23] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019. 4

[24] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019. 2, 3, 4

[25] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. 2

[26] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018. 1, 2, 6

[27] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. In *Advances in Neural Information Processing Systems*, pages 6740–6749, 2019. 2

[28] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3947–3956, 2019. 2, 6

[29] Zaiwei Zhang, Bo Sun, Haitao Yang, and Qixing Huang. H3dnet: 3d object detection using hybrid geometric primitives. In *European Conference on Computer Vision*, pages 311–329. Springer, 2020. 6