SparsePoint: Fully End-to-End Sparse 3D Object Detector

Zili Liu[1*] Guodong Xu[1*] Honghui Yang[1] Haifeng Liu[1]
Deng Cai[1,2†]
[1]State Key Lab of CAD&CG
[2]Alibaba-Zhejiang University Joint Institute of Frontier Technologies
zililiuzju@gmail.com {memoiry, yanghonghui, haifengliu, dcai}@zju.edu.cn

Abstract

Object detectors based on sparse object proposals have recently been proven to be successful in the 2D domain, which makes it possible to establish a fully end-to-end detector without time-consuming post-processing. This development is also attractive for 3D object detectors. However, considering the remarkably larger search space in the 3D domain, whether it is feasible to adopt the sparse method in the 3D object detection setting is still an open question. In this paper, we propose SparsePoint, the first sparse method for 3D object detection. Our SparsePoint adopts a number of learnable proposals to encode most likely potential positions of 3D objects and a foreground embedding to encode shared semantic features of all objects. Besides, with the attention module to provide object-level interaction for redundant proposal removal and Hungarian algorithm to supply one-one label assignment, our method can produce sparse and accurate predictions. SparsePoint sets a new state-of-the-art on four public datasets, including ScanNetV2, SUN RGB-D, S3DIS, and Matterport3D. Our code will be publicly available soon.

1. Introduction

3D object detection aims to localize and recognize 3D objects in a 3D scene. A large number of practical applications rely on 3D object detection, such as augmented reality, robot navigation, robot grabbing, etc. At present, 3D object detection has become a popular research topic and has received widespread attention with the popularity of depth sensors.

Although the performance of 3D object detectors continues to make breakthroughs, they heavily rely on dense object proposals to achieve decent performance. These dense proposals lead to complicated and time-consuming post-processing (e.g., 3D-NMS) in the inference stage. Although 2D object detectors faced the same problem in the past, adopting learnable sparse proposals instead of dense proposals has recently become popular in 2D object detection [2, 36, 23, 24]. They can predict all objects at once without any post-processing. However, considering remarkably larger search space in 3D object detection, it is still an open question whether it is feasible to adopt sparse proposals in the 3D object detection setting. In this paper, we introduce the SparsePoint, which is the first fully end-to-end 3D object detector without requiring dense proposals or any post-processing while achieving state-of-the-art performance. The SparsePoint is illustrated in Figure 1, which consists of two stages — *feature learning* stage and *sparse proposal learning* stage.

In the *feature learning* stage, we first use widely used PointNet++ [18] network to extract point cloud features. Since point clouds are usually distributed on the surface of objects, VoteNet [16] proposes the voting module to aggregate informative foreground points. It learns a set of offsets to push points closer to object centroids. The voting module has been successfully applied in many recent work [32, 4, 27, 7, 10, 8]. However, as shown in Figure 2, we notice that background points may produce uncontrollable offset predictions, which degrades the network optimization. Therefore, we propose a harmonized voting module by learning objectness scores to suppress these undefined offset predictions of background points.

The main difference between SparsePoint and previous 3D object detectors is in the *sparse proposal learning* stage. We adopt K learnable proposal boxes and a single foreground embedding. The K boxes encode K most likely potential positions of objects in 3D space, and the foreground embedding encodes shared semantic information of all 3D objects. We propose the RoI aggregation pooling module to extract K features of K 3D boxes efficiently. Then, we use the foreground embedding to focus on the effective information in K features, and the resulting K refined fea-

*Equal contribution
†Corresponding author

tures will go through a self-attention module to filter correlated features for the removal of the redundant proposals. Finally, We adopt the Hungarian matching algorithm to assign a unique object proposal for each ground truth, while unmatched proposals are treated as negative training samples.

To locate objects more accurately, we propose to refine the detection results multiple times. In the refinement, we replace the above learnable boxes and foreground embedding with predicted boxes and features, respectively. Therefore, we can obtain better initial guesses and object-specific embedding, which is expected to help extract finer object features. We show that multiple refinements make the model locate objects more accurately. Besides, sharing parameters in the refinement makes the model faster to converge.

In experiments, our model set a new state-of-the-art on the four challenging 3D object detection datasets: ScanNetV2 [6], SUN RGB-D [22], S3DIS [1], and Matterport3D [3]. Our study illustrates that 3D object detectors are able to detect objects well through a small number of learnable sparse proposals even in 3D space. Our contribution can be summarized as follows:

- We propose the harmonized voting module to make points closer to object centroids while suppressing uncontrollable offset predictions of background points.
- We propose the RoI aggregation pooling module to efficiently aggregate features and then pool RoI features. The module enlarges the receptive field and improves performance.
- We introduce learnable proposal boxes and a foreground embedding to learn sparse proposals and produce non-redundant predictions effectively.
- We show that dense object proposals are not necessary for 3D object detectors, and a sparse 3D detector is able to set a new state-of-the-art without any post-processing.

2. Related Works

3D Object Detection Based on Point Cloud. The performance of 3D object detectors continues to make breakthroughs with the development of depth sensors and deep learning. Due to the difficulty of multi-sensor fusion, mainstream 3D object detectors directly detect 3D objects on 3D point clouds and have made great progress. According to the point cloud feature extraction networks, these detectors can be divided into two streams: grid-based [29, 28, 11] and point-based [18, 21, 31]. Grid-based methods will first partition point cloud and use 2D CNN or 3D CNN to extract grid-level features. Then, they will take off-the-shelf 2D detection heads for proposal generation.

Point-based methods take raw point cloud feature extraction network such as PointNet++ [18] to generate point-wise features for the subsequent detection. PointRCNN [21] proposes to use a two-stage network similar to Faster-RCNN [19] to localize 3D objects. VoteNet [16] introduces deep hough voting to accurately gather point clouds inside objects for object-level detection. Recently, there is a popular stream attempting to combine features learned from different point cloud representations to obtain more robust features. MVF [35] uses the original point cloud as a bridge to integrate grid-level features in the orthogonal coordinate system and spherical coordinate system. PV-RCNN [20] combines grid-level and point-level features to obtain stronger point cloud feature representation. Without exception, these methods heavily rely on dense proposals to predict accurate 3D objects, and there is a one-to-many match strategy between predictions and ground-truth for the loss calculation during model training. The design of dense proposals makes it necessary to add NMS at the end of the model to remove redundant predictions. Our work explores the use of sparse proposals for one-to-one matching between predictions and ground-truth, so that the time-consuming NMS post-processing can be removed.

2D Object Detection. Object detectors that rely on dense proposals have been developed for several years with significant improvements. In spite of the outstanding achievements in performance [19, 12, 25], the reliance on dense proposals makes complicated post-processing necessary. Although some work [34, 13, 14] successfully removes the post-processing, they still rely on dense proposals in design. Recently, adopting learnable sparse proposal boxes to produce unique prediction [2, 36, 23, 24] becomes popular. They only assign one positive sample for each ground truth, which explicitly makes detectors produce unique predictions for objects. With proposal-level interaction, they can safely remove the post-processing and build a fully end-to-end detector.

3. SparsePoint

3.1. Feature Learning

Backbone Network. We use PointNet++ [18] as our backbone network to directly extract features on the point cloud. Following VoteNet [16], our backbone network consists of multiple set-abstraction layers and up-sampling layers. The outputs of the backbone network are denoted as seed points $S_p = \{s_{p1}, ..., s_{pm}\} \in R^{m\times 3}$ and seed features $S_f = \{s_{f1}, ..., s_{fm}\} \in R^{m\times c}$, where m refers the number of seeds and c denotes the feature dimension. In our

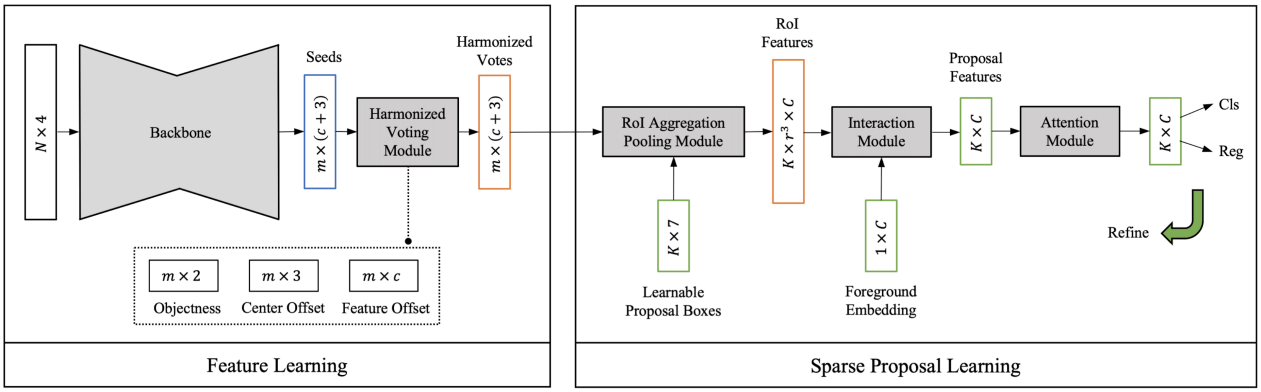


Figure 1. Architecture of SparsePoint for 3D object detection. Our model consists of two stages, i.e., the feature learning stage and the sparse proposal learning stage. In the first stage, our model extracts point features like the most recent dense methods. In the second stage, we use a set of learnable sparse proposal boxes to extract RoI features and produce non-redundant predictions at the end.

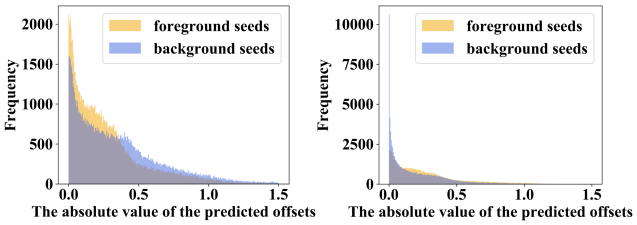
implementation, m is 1024 and c is 256.

Harmonized Voting Module. The point cloud is usually distributed on the object surface, which may be far from the object centroid. To make training easier, VoteNet [16] proposes centroid offset prediction. Specifically, it learns the euclidean offset $\hat{O}_p \in R^{m\times 3}$ from seed points $S_p \in R^{m\times 3}$ to their corresponding object centroids. It defines vote points as $V_p = S_p + \hat{O}_p$, which are expected to be clustered near object centroids.

However, those seed points that do not fall on any object surfaces, i.e., background seeds, are actually not supervised during training. It leads to their uncontrollable offset predictions in inference. The predicted background votes will be harmful for the following detection. As shown in Figure 2(a), absolute values of offsets predicted by background seeds are relatively large. Therefore, many background votes may be scattered outside the scene or inside the object, which is shown in Figure 5. To alleviate this problem, we propose to predict a binary objectness score $\hat{A} = [\hat{A}^-; \hat{A}^+] \in R^{m\times 2}$ to supervise all seed points. Then, we use this score to suppress offset predictions. Since background seeds are expected to have a low objectness score, their offset predictions are well suppressed. Therefore, our harmonized voting points are redefined as $V_p = S_p + \hat{O}_p \cdot sigmoid(\hat{A}^+)$.

Given seed points S_p, we define the binary objectness label $A \in R^{m\times 1}$. Those foreground seeds located inside ground truth boxes are labeled as 1, while background seeds are labeled as 0. Our objectness scores $\hat{A}$ are optimized using the cross-entropy loss L_{obj}.

Suppose we have a subset $T = \{t_1, ..., t_n\} \subseteq S_p \in R^{m\times 3}$ that refers to n foreground seeds in m seeds, and we have n corresponding object centroids $C = \{c_1, ..., c_n\} \in R^{n\times 3}$. The centroid offset loss is defined as:



(a) Voting Module (b) Harmonized Voting Module

Figure 2. Histogram of absolute values of offsets predicted by foreground seeds and background seeds. We randomly select 100000 offset predictions in both (a) and (b), respectively. After adopting our harmonized voting module, absolute values of offsets predicted by background seeds are significantly reduced.

$$L_{off} = \frac{1}{n}\sum_{i=1}^{n} ||\hat{o}_i \cdot sigmoid(\hat{a}_i^+) - (t_i - c_i)|| \quad (1)$$

where $\hat{o}_i$ refers to the predicted offset of i-th foreground seed, and $\hat{a}_i^+$ refers to the predicted objectness score of i-th foreground seed.

In addition to learn the centroid offset, we also learn a feature offset $\hat{O}_f \in R^{m\times c}$. As in VoteNet [16], we define the vote feature as $V_f = S_f + \hat{O}_f$, where S_f is the seed feature.

The total loss in our vote module is defined as follows, weighted by λ:

$$L_{vote} = \lambda_{obj} \cdot L_{obj} + \lambda_{off} \cdot L_{off} \quad (2)$$

3.2. Sparse Proposal Learning

Previous works produce dense object proposals based on seeds or votes [30, 16, 32, 27, 4, 7, 10]. Instead, we use learnable sparse 3D proposal boxes $P_b = \{p_{b1}, ..., p_{bK}\} \in R^{K\times 7}$, where K refers to the number of the boxes. K is

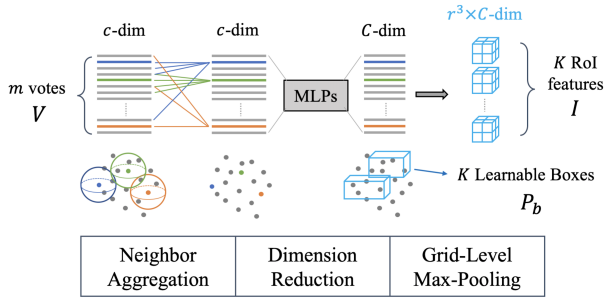


Figure 3. Illustration of the RoI aggregation pooling module. This module takes m vote features and K 3D boxes to produce K RoI features. The pipeline consists of three parts, i.e., neighbor aggregation, dimension reduction, and grid-level max-pooling. The first two parts guarantee larger respective fields and fewer computation budgets in follow-up processes, respectively. The last part efficiently produces RoI features.

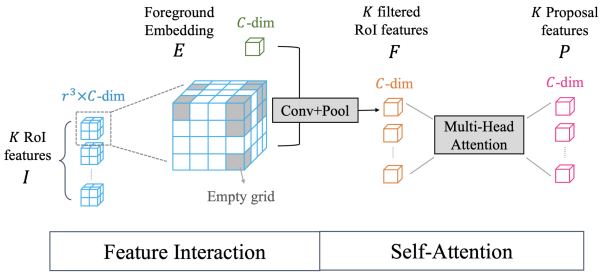


Figure 4. Illustration of the proposal interaction module, which consists of the feature interaction module for refining proposal features and the self-attention module for removing redundant proposals. The module takes K RoI features and a foreground embedding as input. The foreground embedding is learnable, and it encodes the basic feature of all objects. Since many grids in RoI features contain no information, i.e., empty grid, we adopt our foreground embedding as $1 \times 1 \times 1$ convolution parameters to filter effective information. Note that the foreground embedding is shared for all K RoI features. Then, the self-attention provides proposal-level interaction, thus makes it easier to identify redundant proposals and produce unique predictions.

set to be 128 in our implementation, which is greater than the maximum possible number of objects in the 3D scene. P_b encodes K positions where objects are most likely to appear in the training data set. Each box p_{bi} is represented as a 7-dimensional vector $(x, y, z, l, w, h, \theta)$, which denotes the center position, size, and oriented angle of the box.

Sample-Specific Mapping. Different point cloud samples have different point ranges. Therefore, we limit the center position (x, y, z) and size (l, w, h) of box p_{bi} between 0 and 1, and we dynamically map p_{bi} to the real 3D space based on point ranges of a specific point cloud sample. In implementation, we calculate the 5-th quantile (x_5, y_5, z_5) and the 95-th quantile (x_{95}, y_{95}, z_{95}) of input point cloud sample. Given a proposal box p_{bi}, we map its center coordinate $t \in \{x, y, z\}$ to $t \cdot (t_{95} - t_5) + t_5$ and map its size $u \in \{l, w, h\}$ to $u \cdot (t_{95} - t_5)$.

RoI Aggregation Pooling Module. We propose our RoI aggregation pooling module to extract RoI features of K proposal boxes. The module is shown in Figure 3. This module consists of neighbor aggregation, dimension reduction, and grid-level max-pooling. The neighbor aggregation helps enlarge the receptive field and obtain context information, while the dimension reduction reduces the dimension of features to reduce computation budgets. In the grid-level max-pooling, we divide each learnable 3D box into r^3 grids and use max-pooling for features inside each grid.

The resulting feature is named RoI feature $I \in R^{K \times r^3 \times C}$. It denotes K C-dimensional RoI features with a resolution of r^3. In our implementation, r is 5 and C is 128.

Proposal Interaction Module. To suppress redundant predictions, explicitly providing proposal-level interaction is necessary [9]. However, considering that point clouds are sparse and irregular in the 3D space, many grids in the RoI feature I contain no information. In order to extract the information in I more effectively, we propose a feature interaction module.

As shown in Figure 4, we use a learnable foreground embedding $E \in R^{1 \times C}$. The foreground embedding encodes basic features of objects, which is used to filter the effective information in the RoI feature I. Specifically, we treat the foreground embedding as a $1 \times 1 \times 1$ convolution parameter and then conduct convolution and 3D average pooling on the RoI feature I. The resulting features are denoted as $F \in R^{K \times C}$. F contains discriminative features of K potential objects.

After obtaining the discriminative feature F, we introduce a self-attention module to provide proposal-level interaction. It allows the model to identify redundant proposals and produce unique predictions. The resulting interacted feature is named proposal feature $P \in R^{K \times C}$. Our self-attention module is implemented by the multi-head attention [26].

Set Prediction. The model infers the classification and regression results of K proposals. For n ground truth, we find optimal matched n predictions through the Hungarian algorithm and assign n ground truth to n matched predictions [2]. Unmatched $K - n$ predictions are treated as negative training samples. Our binary matching loss is defined as:

$$L_{match} = w_{cls}L_{cls} + w_{L1}L_{L1} + w_{iou}L_{iou} + w_{cor}L_{cor} \quad (3)$$

L_{cls} is the focal loss [12] between predicted categories and target categories. L_{L1} is the L1 loss between normalized

regression results and the target results. L_{iou} is the IoU loss [33] between decoded 3D boxes and the ground truth boxes. L_{cor} is the corner loss [17] between decoded 3D boxes and the ground truth boxes. w represents the weight of each item. After obtaining the optimal match, we use the same loss function but only for the matched pair.

3.3. Proposal Refinement

To better localize objects and make the model converge faster, we propose to refine the predicted boxes multiple times. We replace the initial proposal boxes and foreground embedding with the predicted boxes and interacted proposal feature P. The refinement shares the same model parameters, and the only difference in each refinement is the input proposal box and foreground embedding. During each refinement, the shared foreground embedding gradually transforms from general foreground features ($1 \times C$) to specific object features ($K \times C$), which is beneficial to better information extraction.

Each refinement produces K classification and regression results. We calculate the optimal matching pair and set loss for these prediction results. The final loss is the sum of all set losses.

4. Experiments

4.1. Experimental Settings

Implementation Details. We adopt PointNet++ [18] as our backbone network, and the network details follows VoteNet [16]. Our vote module is implemented by a multi-layer perceptron (MLP) with output channel 256, 256, 261. The last layer outputs 2-dimension objectness score, 3-dimension centroid offset, and 256-dimension feature offset. The loss weight λ_{obj}, λ_{off}, and λ_{se} is set to 1.0, 10.0, and 1.0.

Our RoI aggregation pooling module is implemented by set abstraction layers, MLPs, and a grid-level max pooling. Except pooling features on votes $V = [V_p, V_f]$, as shown in Figure 3, we use another pooling module to pool features on seeds $S = [S_p, S_f]$. The RoI feature I is obtained by using element-wise average on pooled seeds and votes. We find that neighbor aggregation is beneficial when pooling on votes but useless when pooling on seeds. Therefore, we remove the neighbor aggregation when pooling on seeds.

In our self-attention module, we adopt a multi-head attention layer. We set the embedding dimension in this layer to 1024 and the head number to 8. We also enable the dropout in this layer and set the ratio to 0.1.

Finally, the loss weight is set as $w_{cls} = 1.5$, $w_{L1} = 0.45$, $w_{iou} = 2$, $w_{cor} = 0.25$.

Training Details. Our SparsePoint is optimized by AdamW optimizer [15]. We train the model for 180 epochs and the learning rate is reduced by a factor of 10 at epoch 120 and 160. Besides, the weight decay is set to 0.01. Our experiments are based on open source detection toolbox MMDetection3D [5].

4.2. Comparison with State-of-the-Arts

Dataset. ScanNetV2 [6] is a richly annotated dataset of indoor scenes. It contains about 1200 training samples and 18 object categories. ScanNetV2 does not provide orientation information of bounding boxes. Thus we predict axis-aligned boxes. In this dataset, the initial learning rate is 0.0018, and the batch size is 16.

SUN RGB-D [22] is a single-view RGB-D dataset that contains about 5000 images annotated with oriented 3D bounding boxes. We convert RGB-D images to point clouds and follow a standard evaluation protocol, and we report results on the ten most common categories as in VoteNet [16]. In this dataset, the initial learning rate is 0.0018, and the batch size is 16. In this dataset, the initial learning rate is 0.0016, and the batch size is 16.

S3DIS [1] and Matterport3D [3] are both indoor datasets similar to ScanNetV2, and they do not provide rotation information. So we follow the same training and evaluation strategy as ScanNetV2. S3DIS dataset contains 266 samples (199 training samples, 67 validation samples), while Matterport3D dataset is larger and contains 1632 samples (1.4k training samples, 232 validation samples).

Results. We summarize the results in Table 1 and Table 2. Our SparsePoint achieves 65.1 mAP@0.25 and 49.9 mAP@0.5 on ScanNetV2, 61.4 mAP@0.25 and 43.9 mAP@0.5 on SUN RGB-D, 49.7 mAP@0.25 and 27.1 mAP@0.5 on S3DIS, and 43.6 mAP@0.25 and 32.9 mAP@0.5 on Matterport3D. Our results are the state-of-the-art, especially for the mAP@50. It suggests that our model can locate objects more accurately and refined than other detectors. We also list the per-category comparison with VoteNet [16] in Table 3 and Table 4. The improvement in most categories is significant.

Although H3DNet [32] achieves comparable results on ScanNetV2 (i.e., 67.2 mAP@0.25 and 48.1 mAP@0.5), it requires a four times larger backbone to extract point features. Besides, it requires time-consuming post-processing steps like most detectors. Considering that most 3D object detectors need to be deployed to mobile devices with limited computation resources, its inference speed is relatively slow, i.e., 4.4 FPS on a single NVIDIA Tesla V100, as shown in Table 5. On the contrary, the inference speed of our model is about 3 times faster, i.e., 12.9 FPS on the same device.

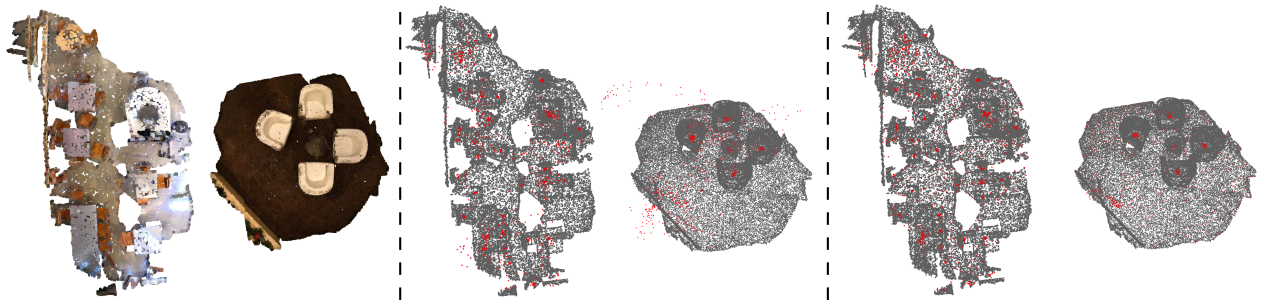


Figure 5. Illustration of vote points (red dots). In the voting module, many vote points are scattered outside the scene. After adopting the harmonized voting module, those uncontrollable vote points are well suppressed.

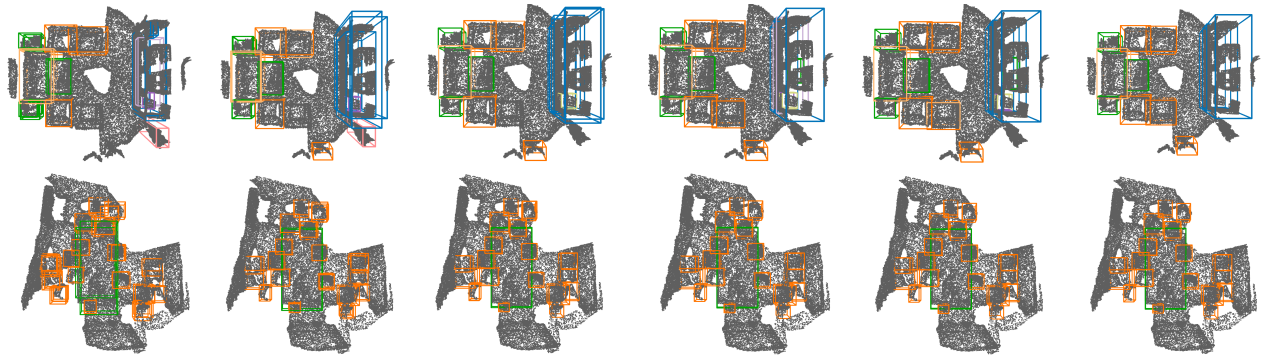


Figure 6. Illustration of predicted boxes during refinement. As the refinement progresses, the model can produce finer 3D boxes while suppressing redundancy.

Method	ScanNetV2 [6]		SUN RGB-D [22]	
	mAP@0.25	mAP@0.5	mAP@0.25	mAP@0.5
VoteNet [16]	58.6	33.5	57.7	32.9
HGNet [4]	61.3	34.4	61.6	-
H3DNet [32]	67.2	48.1	60.1	39.0
MLCVNet [27]	64.5	41.4	59.8	-
SparsePoint	65.1	49.9	61.5	44.2

Table 1. Performance comparison on ScanNetV2 and SUN RGB-D.

Method	S3DIS [1]		Matterport3D [4]	
	mAP@0.25	mAP@0.5	mAP@0.25	mAP@0.5
VoteNet [16]	31.2	-	38.8	-
SparsePoint	49.7	27.1	43.6	32.9

Table 2. Performance comparison on S3DIS and Matterport3D.

4.3. Ablation Study

In this section, we conduct our ablation study on ScanNetV2 [6], and the results are reported on the validation dataset using a single NVIDIA Tesla V100.

Harmonized Voting Module. In our harmonized voting module, we propose to predict objectness scores to suppress those uncontrollable offset predictions of background seeds. As shown in Figure 2, adopting this method significantly reduces the absolute values of offsets predicted by background seeds. Besides, as shown in Figure 5, we can see that the background points tend to stay in place instead of being randomly shifted. This avoids that the background points may be pushed onto the object, causing noise to be introduced when the object information is aggregated, thereby degrading the performance of the detector. Since predicting objectness scores itself also helps introduce more supervise information to improve the performance, we separately study the effects of supervising objectness scores and suppressing offset predictions. As shown in Table 6, both supervising objectness scores and suppressing offset predictions promote the performance. Specifically, they bring 1.2 and 0.7 mAP@0.25 improvements and 0.8 and 1.9 mAP@0.5 improvements, respectively. Besides, the harmonized voting module can also be adapted to other detectors. We replace the voting module in VoteNet [16] with our harmonized voting module. The performance improves 1.9 mAP@0.5.

RoI Aggregation Pooling Module. We propose the RoI aggregation pooling module to extract features of proposal

Method	cab	bed	chair	sofa	table	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	mAP
VoteNet [16]	8.1	76.1	67.2	68.8	42.4	15.3	6.4	28.0	1.3	9.5	37.5	11.6	27.8	10.0	**86.5**	16.8	78.9	11.7	33.5
VoteNet*	17.5	**79.3**	74.5	74.7	47.5	22.5	16.2	37.5	3.3	25.2	36.1	27.2	32.0	13.6	83.7	23.3	**82.1**	26.0	40.1
SparsePoint	**26.3**	77.8	**79.7**	**86.7**	**56.7**	**40.1**	**22.1**	**47.5**	**4.2**	**45.1**	**55.3**	**40.0**	**43.7**	**43.5**	82.1	**40.2**	77.5	**29.0**	**49.9**

Table 3. 3D object detection performance comparison on ScanNetV2 validation dataset. The evaluation metric is Average Precision with 0.5 IoU threshold. * denotes that the model is re-implemented by us.

Method	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	mAP
VoteNet [16]	49.9	47.3	4.6	54.1	5.2	13.6	35.0	41.4	19.7	58.6	32.9
VoteNet*	45.4	53.4	6.8	56.5	5.9	12.0	38.6	49.1	21.3	**68.5**	35.8
SparsePoint	**60.9**	**63.2**	**13.8**	**61.2**	**14.2**	**23.7**	**49.1**	**57.7**	**33.2**	65.4	**44.2**

Table 4. 3D object detection performance comparison on SUN RGB-D validation dataset. The evaluation metric is Average Precision with 0.5 IoU threshold. * denotes that the model is re-implemented by us.

Method	Model Time	NMS Time	Total Time
VoteNet [16]	50.0	43.5	93.5
H3DNet [32]	204.1	23.1	227.2
SparsePoint	77.5	0	77.5

Table 5. Comparison of inference time (ms) across different methods on ScanNetV2. The inference time is measured on a single NVIDIA Tesla V100.

Method	w/ Obj	w/ Sup	mAP@0.25	mAP@0.5
VoteNet [16]			58.6	33.5
VoteNet*			63.2	40.1
	✓		63.3	40.9
	✓	✓	63.2	42.0
SparsePoint			63.1	47.3
	✓		64.3	48.0
	✓	✓	65.1	49.9

Table 6. Effects of supervising objectness scores (Obj) and suppressing offset predictions (Sup) in the harmonized voting module. * denotes that the model is re-implemented by us.

w/ Agg	w/ Red	mAP@0.25	mAP@0.5
		53.8	40.3
✓		55.2	42.9
	✓	64.2	48.3
✓	✓	65.1	49.9

Table 7. Effect of neighbor aggregation (Agg) and feature reduction (Red) in the RoI aggregation pooling module.

boxes. This module consists of three parts, i.e., neighbor aggregation, dimension reduction, and grid-level max-pooling. The first part can help obtain context information, and the second part is used to reduce computation budgets and makes the following attention module easier to optimize. As shown in Table 7, removing any of them is harmful to the speed-accuracy trade-off. We notice that removing dimension reduction leads to more obvious accuracy loss. It shows that the attention module needs a suitable feature dimension. Otherwise, the model will be difficult to train. Besides, the introduction of neighbor aggregation enables the ROI feature to obtain more contextual information, thereby further improving the performance of the detector.

w/ Int	w/ Att	w/ NMS	mAP@0.25	mAP@0.5
✓		✓	60.4	44.6
✓			54.0	41.2
	✓	✓	64.1	45.8
	✓		64.0	46.6
✓	✓	✓	64.7	48.9
✓	✓		65.1	49.9

Table 8. Effect of feature interaction module (Int), self-attention module (Att), and NMS on SparsePoint.

Proposal Interaction Module Proposal Interaction Module is an important component in SparsePoint as it allows communication between proposals for removing redundant proposal and refining proposal features. It consists of a feature interaction module for efficient feature extraction and a self-attention module for proposal-level interaction, and we study the effect of each of them. Results are shown in Table 8. After removing the self-attention module, we notice a significant drop in performance, i.e., 11.1 mAP@0.25 and 8.7 mAP@0.5. However, the performance will improve a lot if we use NMS to suppress redundant predictions. It means that self-attention plays an important role in ensuring producing unique predictions. This phenomenon can also be well observed in Figure 6. After using multiple proposal interaction modules, the redundant detection frames are gradually eliminated. We also notice that the feature interaction module helps to refine proposal features and improve the performance of the model. Besides, after using the self-attention, further adopting the NMS will cause performance degradation.

Proposal Refinement. We refine predicted boxes multiple times in inference. To figure out its effect, we adjust refinement times in our ablation study. As shown in Table 9, refining 6 times achieves the best speed-accuracy trade-off. Besides, not sharing parameters leads to lower performances. We believe that the small size of the dataset is an important factor. We also show the predicted boxes during refinement in Figure 6 for a better illustration. We can see that most of the objects are well located in the first one or two stages, but there is still a certain degree of detection

#Refine	w/ Share	mAP@0.25	mAP@0.5	FPS
1	✓	41.4	23.3	16.7
2	✓	47.7	35.9	15.9
3	✓	53.6	42.1	14.9
4	✓	62.7	48.7	14.2
5	✓	65.1	49.4	13.5
6	✓	65.1	49.9	12.9
7	✓	64.1	47.7	12.2
6		56.6	42.5	12.7

Table 9. Speed-Accuracy trade-off when adjusting the refinement number and removing the parameter sharing.

redundancy and false positive. Through multiple stages of proposal interaction, the model can produce more accurate 3D boxes while suppressing redundancy.

5. Qualitative Results and Discussion

We compare the qualitative results of VoteNet and SparsePoint on the ScanNetV2 validation set in Figure 7. It is observed that SparsePoint can significantly reduce the occurrence of false positives and generate more accurate true positives. In particular, we can see in Figure 7 (b) and (c) that VoteNet designed with the dense proposal will tend to produce detection results with different categories for the same instance, and it is difficult to filter out redundant detection results by using commonly adopted class-aware NMS (NMS is only performed within objects of the same class.). Besides, the class-agnostic NMS also cannot handle this situation well. For instance, Figure 7 (a) shows that the table and the chair overlap, and the class-agnostic NMS may filter out the chair or the table and cause false negatives. In contrast, benefiting from the sparse proposal design, SparsePoint can generate a certain type of detection result for a single instance without resorting to NMS, and can handle occlusion and non-occlusion conditions well. This advantage makes SparsePoint a more practical detector in the real-world scenario.

6. Conclusion

We introduce SparsePoint, the first fully end-to-end 3D object detector without requiring dense object proposals or any post-processing. We show that the widely used voting module produces uncontrollable offset predictions on background points. Therefore, we propose harmonized voting module to effectively suppress these offset predictions. We develop the RoI aggregation pooling module to capture rich context information and then extract RoI features, which is proven to be useful for promoting performance. With multiple refinements, our sparse method can locate 3D objects accurately without any post-processing (e.g., 3D NMS). The SparsePoint achieves state-of-the-art performances on ScanNetV2, SUN RGB-D, S3DIS, and Matterport3D, which demonstrates that our sparse method is effective and works well in the 3D domain.

References

[1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016. 2, 5, 6

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 1, 2, 4

[3] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from RGB-D data in indoor environments. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*, pages 667–676. IEEE Computer Society, 2017. 2, 5

[4] Jintai Chen, Biwen Lei, Qingyu Song, Haochao Ying, Danny Z Chen, and Jian Wu. A hierarchical graph network for 3d object detection on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 392–401, 2020. 1, 3, 6

[5] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. *CoRR*, abs/1906.07155, 2019. 5

[6] 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, 5, 6

[7] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9031–9040, 2020. 1, 3

[8] 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. 1

[9] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018. 4

[10] 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. 1, 3

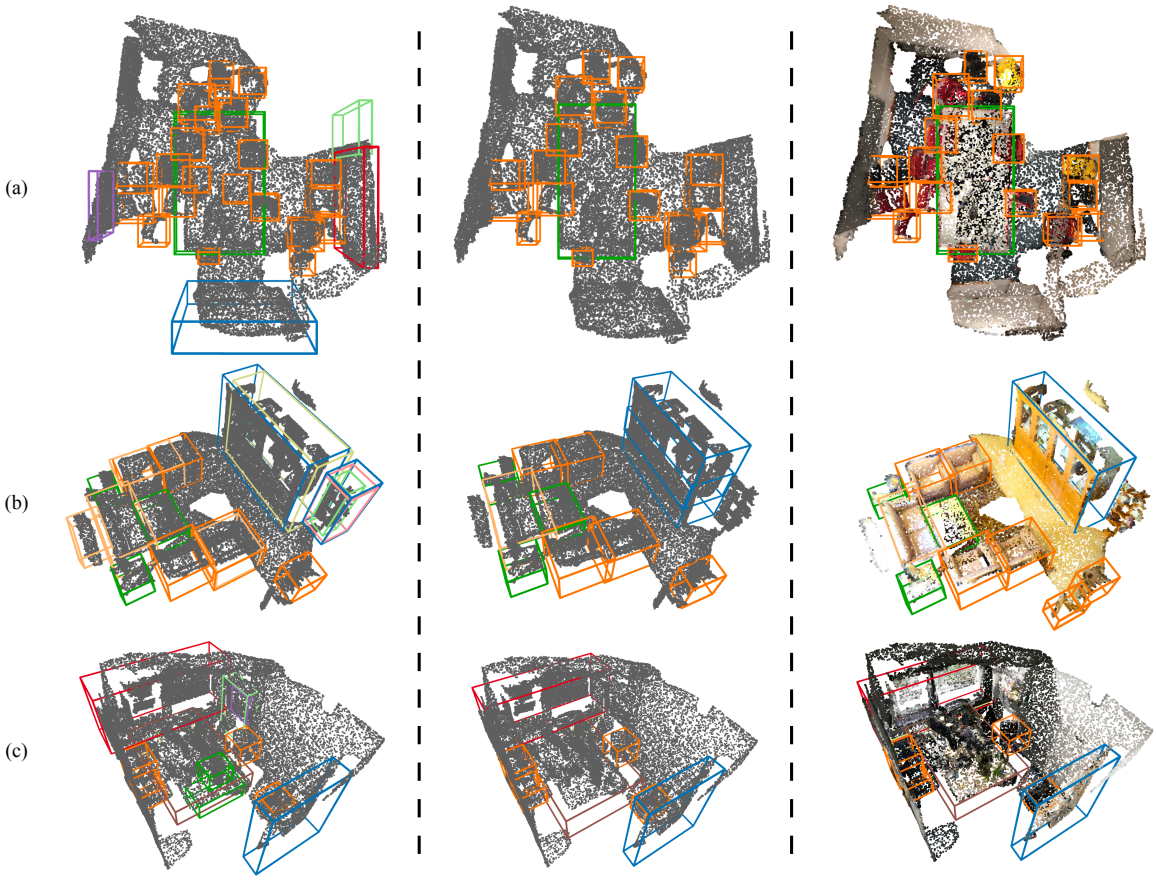


Figure 7. Qualitative results of 3D object detection on the ScanNetV2.

[11] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12697–12705. Computer Vision Foundation / IEEE, 2019. 2

[12] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2, 4

[13] Zili Liu, Tu Zheng, Guodong Xu, Zheng Yang, Haifeng Liu, and Deng Cai. Training-time-friendly network for real-time object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11685–11692, 2020. 2

[14] Zili Liu, Tu Zheng, Guodong Xu, Zheng Yang, Haifeng Liu, and Deng Cai. Ttfnext for real-time object detection. *Neurocomputing*, 433:59–70, 2021. 2

[15] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018. 5

[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/CVF International Conference on Computer Vision*, pages 9277–9286, 2019. 1, 2, 3, 5, 6, 7

[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. 5

[18] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5099–5108, 2017. 1, 2, 5

[19] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 2

[20] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

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

[22] 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, 5, 6

[23] Peize Sun, Yi Jiang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Onenet: Towards end-to-end one-stage object detection. *CoRR*, abs/2012.05780, 2020. 1, 2

[24] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse R-CNN: end-to-end object detection with learnable proposals. *CoRR*, abs/2011.12450, 2020. 1, 2

[25] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9627–9636, 2019. 2

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. 4

[27] Qian Xie, Yu-Kun Lai, Jing Wu, Zhoutao Wang, Yiming Zhang, Kai Xu, and Jun Wang. Mlcvnet: Multi-level context votenet for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10447–10456, 2020. 1, 3, 6

[28] Yan Yan, Yuxing Mao, and Bo Li. SECOND: sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2

[29] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[30] 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 Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 6737–6746, 2019. 3

[31] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11037–11045. IEEE, 2020. 2

[32] 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. 1, 3, 5, 6, 7

[33] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. Iou loss for 2d/3d object detection. In *2019 International Conference on 3D Vision (3DV)*, pages 85–94. IEEE, 2019. 5

[34] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *CoRR*, abs/1904.07850, 2019. 2

[35] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 923–932. PMLR, 2019. 2

[36] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. *CoRR*, abs/2010.04159, 2020. 1, 2