Density Based Clustering for 3D Object Detection in Point Clouds

Syeda Mariam Ahmed Chew Chee Meng
National University of Singapore
e0020829@u.nus.edu chewcm@nus.edu.sg

Abstract

Current 3D detection networks either rely on 2D object proposals or try to directly predict bounding box parameters from each point in a scene. While former methods are dependent on performance of 2D detectors, latter approaches are challenging due to the sparsity and occlusion in point clouds, making it difficult to regress accurate parameters. In this work, we introduce a novel approach for 3D object detection that is significant in two main aspects: a) cascaded modular approach that focuses the receptive field of each module on specific points in the point cloud, for improved feature learning and b) a class agnostic instance segmentation module that is initiated using unsupervised clustering. The objective of a cascaded approach is to sequentially minimize the number of points running through the network. While three different modules perform the tasks of background-foreground segmentation, class agnostic instance segmentation and object detection, through individually trained point based networks. We also evaluate bayesian uncertainty in modules, demonstrating the over all level of confidence in our prediction results. Performance of the network is evaluated on the SUN RGB-D benchmark dataset, that demonstrates an improvement as compared to state-of-the-art methods.

1. Introduction

Despite the recent breakthroughs in 3D object detection, the task of region proposal is heavily dependent on 2D object detectors. Following the conventions of image based detection, candidate 3D proposals are typically generated through sliding windows [35, 16, 31, 4] or by 3D region proposal networks such as [27, 28]. However, the computational complexity of 3D search grows cubically with respect to the resolution and becomes computationally expensive for large scenes or real-time applications. Alternatively, methods that project points to 2D images [3, 35] compromise on geometric and surface properties that may be important in heavily cluttered and occluded environments.

While few works have attempted to directly learn from point features, one of the most successful networks for processing 3D groups of points is PointNet [20, 22] that can be used to perform object detection and semantic segmentation for point clouds. Their initial work for a complete 3D object detection pipeline [19] integrates a 2D region proposal network that generates bounding box proposal from an RGB image and lifts them to a 3D frustum. The pointcloud within the proposed frustum is then segmented using PointNet and is used to regress the amodal bounding box of the object in 3D. While their approach shows improved accuracy, the performance of this method is restricted by the performance of the 2D object detector.

This approach is recently updated by a 3D network that employs Hough voting to predict probable instances of objects, followed by bounding box detection [18]. While the network directly operates on an input point cloud of 20-40k points, after feature learning, it only samples 1024 points as seeds for voting that generate object locations for unique instances. Since the point density for objects may vary in a point cloud, it is unlikely that smaller or heavily occluded objects are well represented in such a small number of seeds.

One of the biggest bottlenecks in processing unorganized pointclouds through state-of-the-art networks is the number of points that can be taken as input. This number of points directly affects the size and the computational complexity of the network, impacting the quality of feature learning. Consequently, this paper focuses on an incremental reduction in the size of the pointcloud, achieved through implementation of smaller modules for tasks with varying objectives, where each task outputs a reduced number of points for the subsequent module to process.

Towards this goal, we introduce a novel cascaded network that proposes individual modules for the following tasks: a) **background-foreground segmentation** and b) **class agnostic instance segmentation** and c) **3D object detection**, as shown in Fig. 1. The proposed network is trained in a modular fashion using PointNet++ [22] as a backbone for segmentation modules, while the segmented clusters are inferred through the recently proposed Edge-Aware PointNet [1] for the task of 3D object detection.

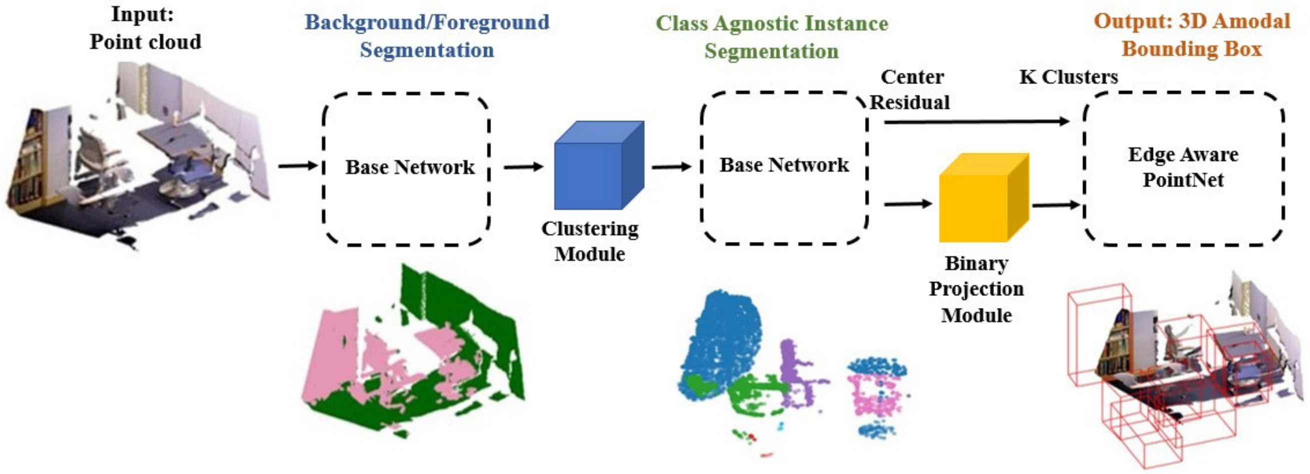

Figure 1. Overview of the proposed pipeline for object detection. Given a 3D scene, the point cloud is first segmented into background-foreground points. Next, only foreground points are clustered using DBSCAN, an unsupervised density based region growing algorithm. Each point in a cluster is assigned a centroid location that is fine-tuned using the second base network. Output from this module is K clusters that are inferred through Edge-Aware PointNet (EPN) to predict class and regress 3D amodal bounding boxes for each cluster.

Specifically we make the following contributions:

- Propose a novel cascaded framework for 3D object detection that can directly achieve class agnostic segmentation of point clouds, followed by amodal bounding box prediction. As opposed to training a network through multi-task learning, we propose to train simpler modules individually and leverage uncertainty predictions to generate confident proposals for the final task.

- The first module performs **background-foreground segmentation**, where the background class consists of walls and floors while foreground consists of all other objects. The objective is to remove all points from the point cloud that may generate false positives in the subsequent tasks. This step automatically reduces the size of the point cloud, without compromising object observability.

- The next module is a novel **clustering based instance segmentation** module where foreground points from the previous stage are clustered using an unsupervised algorithm, DBSCAN [5]. An encoder-decoder framework jointly predicts instances through regression of offset vector between the proposed cluster centroids and ground truth. Consequently, each point votes for an offset distance that indicates the true location of the instance centroid.

- Finally the last module, Edge-Aware PointNet (EPN) predicts bounding parameters that include size, location and orientation for individual object instances.

2. Related Work

2.1. Instance Segmentation

Despite concentrated efforts to improve 3D deep learning networks, literature for 3D instance segmentation lags behind its 2D counterpart. The first prominent attempt in this direction was introduction of Similarity Group Proposal Network (SGPN) [32], that generates group proposals of object instances by learning representation in the form of a similarity matrix. Consequently, points that belong to the same instance have similar features in the matrix as opposed to those in different object instances. Another recent approach was 3D Semantic Instance Segmentation (3D-SIS) [9] that employs multi-view RGB images to learn 2D features while backprojecting them on to associated 3D voxels. They leverage 2D feature learning to 3D for the tasks of object detection and instance segmentation.

Joint Semantic-Instance Segmentation (JSIS) [17] 3D is the only work that addresses panoptic (semantic and instance) segmentation as a joint task for learning from point clouds. Instead of directly processing the complete point cloud, they scan for overlapping 3D windows which are then passed to a point network to predict semantic class labels of the vertices within the window and embedd the vertices into high-dimensional vectors. This is followed by conditional random field for optimization of final results. They define instance segmentation using a *push-pull* embedding where L_{pull} attracts embedding towards the centroids, while L_{push} keeps these centroids away from each other. The regularisation loss L_{reg} acts as a small force that draws all centroids towards the origin. One of the main issues with this method is the requirement to run the network for all 3D windows generated from the original point cloud.

Thus this method can become computationally expensive for dense large scale point clouds.

Inspired by SGPN and JSIS, we address the task of 3D instance segmentation by directly learning from points. However, unlike both methods, we leverage geometric properties of objects in 3D to initiate this task using DBSCAN [5], an unsupervised clustering algorithm. As a result, we start the training of our network with the instance predictions of DBSCAN, while PointNet++ [22] fine-tunes the instances to achieve an improved performance for instance segmentation, while using a shallow network.

2.2. 3D Object Detection

The three most popular approaches for working with volumetric data are 3D CNNs, 2D multi-view CNNs and point networks. State-of-the-art 3D CNN methods [4, 16, 24, 27, 28, 31, 33] are predominantly volumetric quantization based approaches that are popular because they incorporate the complete point cloud and directly exploit 3D information. Song *et al.* [28] and Zhou *et al.* [35] proposed to encode each non-empty voxel with statistical quantities that are derived from points contained within each voxel such as the truncated signed distance or binary encoding [15].

Similarly, multi-view CNNs [3, 11, 21, 29, 30] use 2D rendered images of the 3D point cloud, significantly reducing the computational complexity of the network. PointPillars [14] is another method that enables end-to-end learning with only 2D convolutional layers by using a novel encoder that learns features on pillars (vertical columns) of the point cloud to predict 3D oriented boxes for objects. MV3D [3] introduces a multi-view representation for a LIDAR point cloud by computing a multi-channel feature map in the bird's eye view and the cylindrical coordinates in the frontal view.

While it is observed that multi-view CNN based recognition [34, 26, 12] is a better approach as the network is trained to recognize 3D objects under occlusion. However, both the bird's view projection and voxelization based methods suffer from information loss due to data quantization, and 3D CNN is both memory and computation inefficient.

Point based networks were first proposed by [20, 22] that have been used for object classification of full 3D CAD models, semantic segmentation and 3D object detection in scenes [19]. Based on this network, Shi et al [25] proposed PointRCNN, a novel bottom-up point based network which directly generates robust 3D proposals from raw point clouds, which is both efficient and quantization free. The underlying assumption for this network is that the desired objects are naturally well separated, which is true for KITTI dataset but it may not achieve high performance for indoor point clouds due to high level of clutter and occlusion where objects often overlap.

In this paper we propose to use a cascaded modular approach which allows the flexibility to modify the network architecture for each sub task. As a result, we use PointNet++ for background-foreground and instance segmentation, but Edge-Aware PointNet (EPN) [1] for object detection. EPN integrates PointNet++ with a parallel stream of 2D binary image based convolutional neural network (CNN). The PointNet++ layer of the proposed framework takes as input individual instances of 3D point clouds, while the complimentary CNN layer of the network is provided with 2D binary maps. This approach allows feature learning through the combination of point networks and traditional CNNs, resulting in improved performance.

3. Network Architecture

We introduce a cascaded modular network that performs background/foreground segmentation followed by class agnostic segmentation and 3D object detection, as shown in Figure 1. The primary reason for using multiple cascaded modules for segmentation is a principled reduction of points from the point cloud. This aspect of downsizing the point cloud is critical as the original raw point cloud may contain millions of points while the computational complexity of a point based network is directly affected by the number of points being processed.

Consequently, we observe that state-of-the-art point based methods [18, 19, 17] typically process 1024-5000 points. Assuming semantic segmentation is directly implemented on the complete point cloud, without foreground-background segmentation, the network can only process M points, as a result fewer points will contribute towards feature learning of each object. Due to the proposed cascaded modules, there is a significant reduction in the size of the point cloud, once background points are dropped from the scene. Thus, the subsequent modules will consist of points that are meaningful for the specific task, resulting in a more focused receptive field.

3.1. Background/Foreground Segmentation

The segmentation pipeline is initiated with background/foreground segmentation, where background consists of walls, floor and ceiling while foreground consists of all other objects. The input point cloud is of size $N \times 6$, where each point N is associated with a 3D coordinate $\{x, y, z\}$ and RGB $\{R, G, B\}$ values.

The backbone of segmentation modules is a point based encoder-decoder framework, as proposed by Qi et al [22]. The encoder consists of set abstraction layers (SA) that sample nearest neighboring points from the cloud and learn feature vectors using multi-layered perceptrons (MLPs). The SA layers sub-sample N_{l+i} points from the previous layer N_l, where $N_{l+i} \subset N_l$ and generate a feature vector for every point in the sub-sampled group. Similarly, the decoder

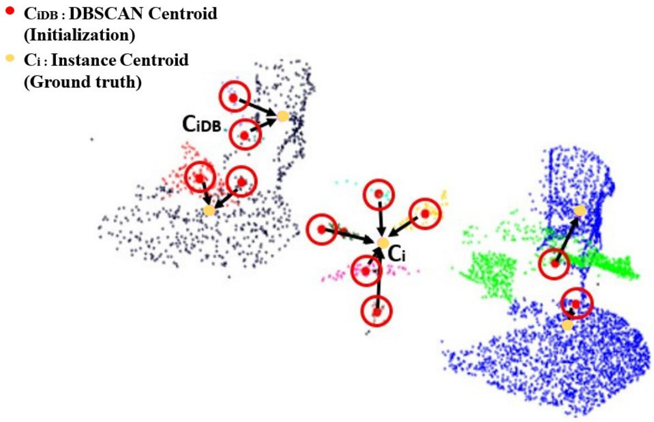
CiDB : DBSCAN Centroid (Initialization)
Ci : Instance Centroid (Ground truth)
CiDB
Ci

Figure 2. Details of the instance segmentation module. The module is initialized using initial clusters where each point is associated with a vector $\triangle C_i$ that represents the offset between $C_{i_{DB}}$ and C_i.

consists of feature propagation layers (FP) that interpolates feature values f of the sub-sampled points N_l to N_{l-i}. Interpolation of features from one layer to the next is carried out using the inverse distance weighted average that is defined as [22]:

$$f^{(j)}(x) = \frac{\sum_{i=1}^{k} w_i(x) f_i^{(j)}}{\sum_{i=1}^{k} w_i(x)} \tag{1}$$

where $w_i(x) = \frac{1}{d(x,x_i)^p}$ represents the inverse distance weighted average for k nearest neighbors.

Unlike a sliding window method that is traditionally used to process a complete point cloud [19], we randomly sample N points from the point cloud. While the network processes N points, the original cloud might be much larger, of a size $T \times N$. To propagate foreground/background semantic labels throughout the point cloud, we propose *'Nearest Neighbor - Upsampling (NN-Ups)'*, defined as follows.

$$S(\hat{y}_p) = arg\max\{S(y_1)...S(y_k)\} \tag{2}$$

where $S(y_1), ...S(y_k)$ are the predicted labels of k nearest neighbors from the points inferred through the network, while $S(\hat{y}_p)$ are points from the original point cloud. Using this approach, we can interpolate semantic labels to the original point cloud.

3.2. Class Agnostic Instance Segmentation

Once foreground points are obtained, a novel instance segmentation module is introduced, that predicts class agnostic instances. To generate individual object instances, direct regression of object centroids is often difficult to converge as there is large variation in object scale, size and density in 3D point clouds. Consequently, we propose to group points into clusters using an unsupervised clustering algorithm DBSCAN [5], which results in each point in the cluster being associated with a centroid $C_{i_{DB}}$ for the instance that it belongs to, as shown in Figure 2. Thus the objective of the network is to predict an offset $\triangle C_i = \{\triangle c_{x_i}, \triangle c_{y_i}, \triangle c_{z_i}\}$ between $C_{i_{DB}}$ and the actual centroid C_i obtained from the ground truth.

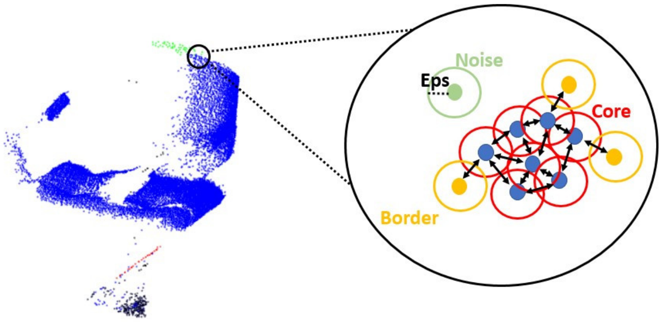

Figure 3. Explanation of the concept of density based clustering using DBSCAN. This algorithm is dependent on two parameters, ϵ that specifies the range of neighborhood for each point and μ that gives the min number of points that can form a cluster.

The DBSCAN algorithm is dependent on two parameters, ϵ that specifies the range of neighborhood for each point and μ that gives the min number of points that can form a cluster, as shown in Figure 3. The algorithm initially identifies core points in the data, followed by searching for connected components of the core points in the neighbor graph and forming clusters from among these points. Finally, the non-core points are assigned to the nearest cluster if it is an ϵ neighbor. This approach does not require prior information regarding the number of clusters and is robust to outliers.

The network architecture uses the same backbone as the foreground/background segmentation module. However the network performs the task of instance regression. The loss function of the network is L_{ins} which is given as:

$$L_{ins} = \frac{1}{N}\sum_{i}^{N} ||\triangle C_i - \triangle C_i{}^*|| \tag{3}$$

where $\triangle C_i$ is the true offset between $C_{i_{DB}}$ and C_i, while $\triangle C_i{}^*$ is the predicted offset vector for every point i in the point cloud.

The concept of initial clustering has two fundamental benefits over directly regressing instance centroids, as proposed by Deep Hough Voting [18]. First, the initial clustering of dense point clouds generates a geometry based estimate about objects. This improves sampling of points from the dense point cloud, which will now consider to sample points from all clusters, irrespective of their size and density. Unlike Farthest point sampling algorithm or random sampling, this approach directly ensures that points are being sampled from all important regions that may contain an object.

The second advantage exists in the faster convergence of the training network that is able to predict smaller centroid offsets $\triangle C_i$ with a higher accuracy as opposed to direct regression of centroid locations. This concept is demonstrated in Figure 2 where all points are initially associated with $C_{i_{DB}}$, shown with red dots, indicating the cluster instance they belong to. Since the ground truth centroid location is C_i, shown with yellow dots, the network is trained to learn the true offset between predicted centroid and true centroid. Once the true instances are predicted, each instance is sampled by the following module to predict amodal bounding boxes and class labels.

Similar to the previous module, predicted centroids for instances are propagated through all the foreground points using Eqn.(2).

3.3. EPN: Edge-Aware PointNet

Edge-Aware PointNet (EPN) is a novel deep learning network that we recently proposed [1] for object recognition from 3D point clouds. The architecture of EPN consists of two parallel channels for feature learning, which is an extension to the originally proposed PointNet++ [22]. The novelty of the proposed network is the integration of PointNet++ Feature Extractor and Edge-Aware CNN branch that explicitly uses 3D boundary points as geometric priors to enhance feature learning.

More formally, the network takes as input a matrix $X \in \mathbb{R}^{N\times 4}$ where N is the fixed number of points randomly sampled from the output of our second module, while the four channels represent Euclidean coordinates and a logical value indicating if the given point is edge or non-edge. The edge detection algorithm used is also based on our previous work [2].

The network consists of three main aspects: a) Binary Projection, b) PointNet Feature Extractor and c) ECNN Feature Extractor. The binary projection module use a mapping function S to convert the 3D matrix X to a series of 2D binary maps [1]. These maps fundamentally represent the geometric shape of the given object, while being low resolution, which prevents the network from being computationally intensive. We use this network as our final module that takes K clusters and predicts class labels and bounding box parameters for each object.

3.4. Loss Function for EPN

EPN aims to predict object class label and an amodal bounding box through six classification and regression tasks, defined as $\mathbf{y} = \{y_{lc}, y_{ac}, y_{oc}, y_{cen}, y_{\triangle ac}, y_{\triangle oc}\}$. The classification tasks are y_{lc} which represents label classification, y_{ac} which represents anchor box classification while y_{oc} represents orientation bin classification. Similarly, the regression tasks are $y_{cen}, y_{\triangle ac}, y_{\triangle oc}$ that represent regression of bounding box centroid and offsets from anchor box and orientation bin.

We parameterize the 3D bounding box as $BB_i = \{x_i, y_i, z_i, l_i, w_i, h_i, \theta_i\}$, where $\{x_i, y_i, z_i\}$ represents the centroid location, $\{l_i, w_i, h_i\}$ are length, width and height of the box, representing its size, and θ_i is the yaw rotation around Z-axis, for a given point cloud instance i. We formulate the bounding box prediction based on [19] where we first pre-define a fixed number of anchor boxes with specific dimensions and then regress the offset of each object from the closest anchor.

The classification module takes as input the joint feature vector $F = \{F_{PN}, F_{ECNN}\}$ from the PointNet and ECNN branches respectively and are followed through a series of fully connected (FC) and dropout layers. The softmax function is used to generate C class probabilities while the cross-entropy loss function is used for training all classification tasks.

All regression tasks $\{y_{cen}, y_{\triangle ac}, y_{\triangle oc}\}$ are trained using Huber loss [10], defined as follows.

$$L_r = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \leqslant \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{4}$$

where y is the true value, $f(x)$ is the predicted value from the network and δ is pre-defined threshold below which the loss becomes quadratic. The combined loss for the EPN network is:

$$L = L_c\{\text{lc}\} + \lambda_1 L_c\{\text{ac}\} + \lambda_2 L_c\{\text{oc}\} + \lambda_3 L_r\{\text{cen}\}$$

$$+\lambda_4 L_r\{\triangle ac\} + \lambda_5 L_r\{\triangle oc\}\} \tag{5}$$

where $\lambda_1, ..., \lambda_5$ are variable parameters that describe the weights associated with each task.

3.5. Bayesian Uncertainty Estimation

Bayesian neural networks replace the deterministic weights with probability distributions over the weights, in a neural network [6]. As a result, the objective is no longer to optimize the weights directly, but to optimize the parameters of the prior distributions. Recently, [7] proposed to compute uncertainty using a Dropout layer, where a prior Bernoulli distribution is placed over weights of the neural network, which when performed during inference, can be used to compute uncertainty in prediction.

In the proposed network, all PointNet++ modules consist of at least a single Dropout layer, which is used to compute variance for for the respective module. This variance in the cloud is used to filter out points associated with uncertain decisions and improve overall prediction accuracy.

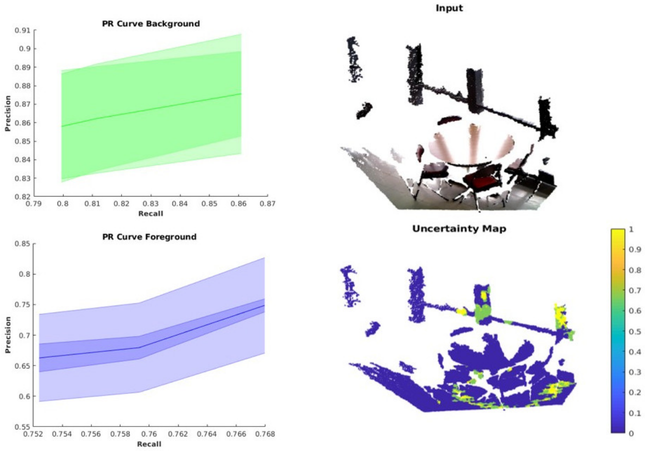

Figure 4. Precision recall curve for background-foreground segmentation under varying Bayesian Mean threshold. The graphs show that precision and recall values increase when a lower Bayesian Mean threshold is selected to filter out non-decisive point predictions. Consequently, the uncertainty map on the bottom-right image shows that points with higher uncertainty (light colored points) belong to the background class, indicating the erroneous labelling for this class.

4. Experimental Results

4.1. Implementation Details

Our network operates on dense indoor pointclouds from the SUN-RGBD dataset that contains above 100k points per scene. The first segmentation module samples 10K points from the point cloud, followed by instance segmentation that samples 5k while the number points sampled by EPN from each of the S object proposals are 1024. The points are randomly sampled on-the-fly while if a point cloud has fewer points, the same points are repeated to achieve the designated number. The backbone network PointNet++ consists of four set abstraction (SA) layers and four feature propagation (FP) layers, for both segmentation modules. The SA layers use a variable radius of $0.1 - 0.8$ for sampling the nearest neighbors. The four FP layers sample the points back to the original number, with 256 dimensions in the first two layers followed by 128 dimensional features. This is followed by two fully connected layers (FC) with a dropout layer of $p = 0.5$. The PointNet branch in EPN consists of three SA layers where the first two layers perform multi-scale grouping to sub-sample $N = \{512, 128\}$ points with $k = \{64, 64\}$ nearest neighbours, using a radius based search of radius $\{0.2, 0.4\}$. Each SA layer is followed by PointNet layer with MLP of sizes $\{64, 64, 128\}, \{128, 128, 256\}$. The last SA layer does not perform sub sampling but accumulates all feature maps, followed by an MLP of size $\{256, 512, 1024\}$.

The ECNN branch of the network trains in parallel to the PointNet branch with two CNN layers. The dimensions of the convolutional filters for the two layers are $f = \{64, 128\}$ while the kernel size for convolutions are $k = \{5, 5\}$. Each 2D CNN layer is followed by a max pooling layer with a kernel size of $k = \{2, 2\}$ and a stride of 2. After the second max pooling layer, the feature maps are converted to a vector following through two fully connected layers of size $\{2048, 1024\}$ where each FC layer is followed by a dropout layer with dropout probability of 0.5.

Class	Acc
Foreground	75.8
Background	75.9
mAcc	75.8
BM	**79.1**

Table 1. Mean accuracy for background/foreground segmentation on the SUN-RGBD dataset.

4.2. Evaluation and Comparison

We evaluate all three modules on the SUN-RGBD dataset. The first module segments background and foreground from the point clouds. The ground truth is generated using centroids of objects and finding all points within the bounding box, that are labelled as foreground. All the remaining points are labelled as background. The evaluation metric for this segmentation is mean class accuracy (mAcc) and Bayesian Mean (BM). BM is defined as mAcc over points that have low Bayesian variance, evaluated as:

$$Var(y) = \frac{1}{T}\sum_{t=1}^{T} f^{\tilde{w_i}}(x) f^{\tilde{w_i}}(x_t) - E(y)^T E(y) \qquad (6)$$

BM filters out points that have Var $> \lambda$, where λ is the threshold that decides the cutoff threshold for discarding points.

Figures 4, 5 show details of ground truth and prediction from the first segmentation module. It can be seen from the figures that labelling is slightly erroneous. However, using Bayesian uncertainty estimates, incorrect points can be identified with higher variance and can thus be ignored in the final prediction. As a result, instance segmentation only processes points with Var $< \lambda$ where $\lambda = 0.2$. The results for this segmentation are tabulated in Table 1 that shows that Bayesian Mean results in an improved accuracy in the overall segmentation task.

To evaluate results for instance segmentation, we use the standard AP^r [8], which computes the mean average precision (mAP) under different Intersection over Union (IoU) scores between the predicted and ground truth segmentation (in place of IoU between bounding boxes). This module generates an offset $\triangle C_i = \{\triangle c_{x_i}, \triangle c_{y_i}, \triangle c_{z_i}\}$ between $C_{i_{DB}}$, the cluster centroids predicted by DBSCAN clustering, and the actual centroid C_i obtained from the ground truth. Once the new centroids are predicted, they are clustered using a distance based method that combines centroids

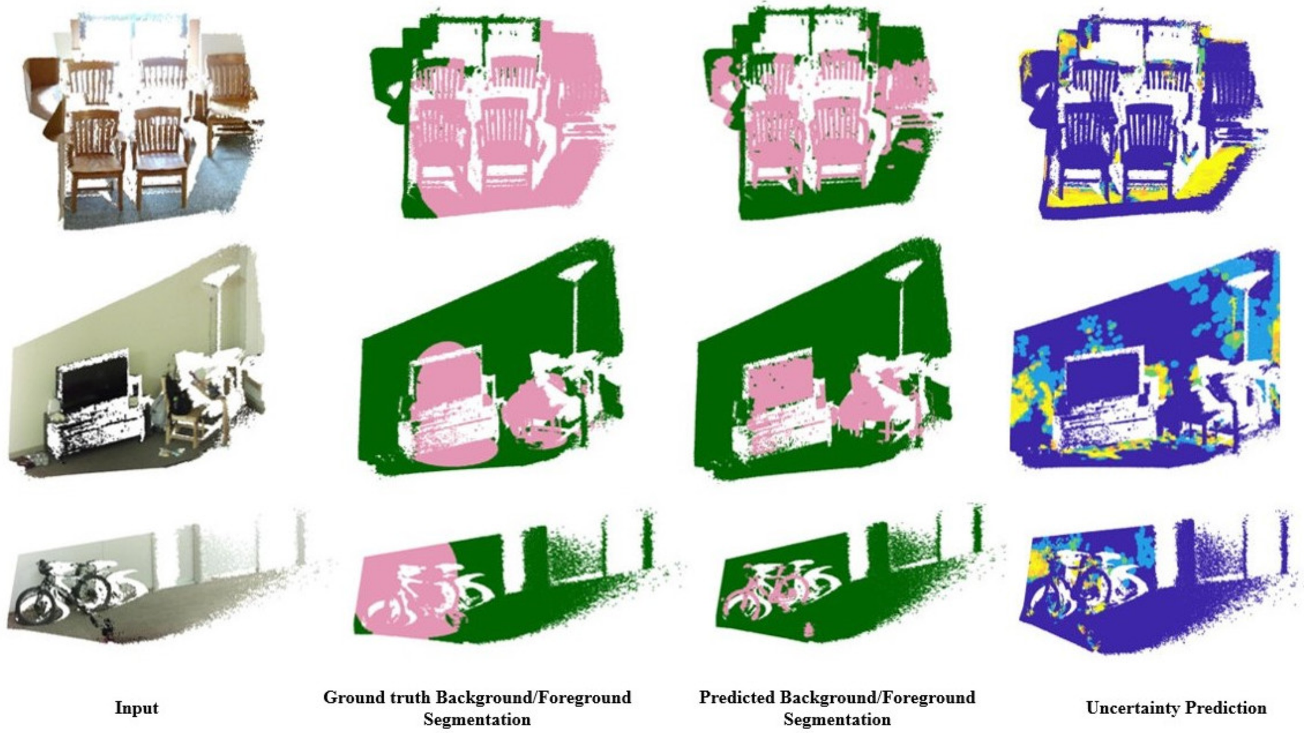
Input
Ground truth Background/Foreground Segmentation
Predicted Background/Foreground Segmentation
Uncertainty Prediction

Figure 5. From left to right: Original point cloud, ground truth segmentation, predicted segmentation and uncertainty map. The figures show examples of the module and uncertainty prediction that removes points with high variance, thus achieving improved segmentation results. For predicted uncertainty, a darker color represents minimal variance while lighter colors indicate high variance.

Metric	Bathtub	Bed	Bookshelf	Chair	Desk	Dresser	Nightstand	Sofa	Table	Toilet
mAP/0.25	68.71	53.55	59.34	24.58	31.04	64.17	38.91	52.46	44.13	93.48
mAP/0.35	69.32	54.28	60.12	24.59	31.39	64.49	**38.92**	52.99	44.32	93.67
mAP/0.45	69.38	54.83	60.42	24.67	31.76	64.71	38.17	53.73	44.77	93.86
mAP/0.55	68.90	56.20	61.23	**24.91**	32.56	65.15	38.60	54.58	45.89	**93.96**
mAP/0.95	**83.29**	**64.39**	**68.84**	22.57	**38.01**	**68.11**	36.23	**64.15**	**54.77**	91.94

Table 2. 3D instance segmentation on scans from SUN-RGBD. We evaluate mAP for varying ϵ, with IoU@0.25 threshold over 10 classes.

that are within a specified range, defined by ϵ. Table 2 shows instance segmentation results for varying ϵ. It can be seen that with higher ϵ, mAP for larger objects increases significantly as compared to lower ϵ.

While this module is able to achieve accurate results for larger objects, with higher ϵ, it is observed that smaller objects like chairs and nightstand have a significantly lower mAP. These results are also verified by Figure 6 where we can see that smaller clusters of chairs are often confused to be part of the table. It is also observed that the DBSCAN algorithm discards disconnected components of same object as noise which often results in an incomplete object at the output. We hypothesize that these issues can be resolved by considering a larger number of points (currently this module processes 5000 points), so that a higher density of objects will reduce objects being labelled as noise.

We evaluate bayesian uncertainty for the instance segmentation over 10 trials. However, this variance does not measure beyond 0.02 which is not significant enough to bring an impact in the overall results. Consequently, the main impact of uncertainty estimation comes from the first segmentation module. The instances predicted by the instance segmentation module serves as input for EPN, where we use $\epsilon = 0.95$. In addition, we concatenate the original points with predicted instances to determine K clusters, where $K = 15$. The results for this network are tabulated in Table 3 where EPN is able to achieve mAP comparable with the state-of-the-art methods. While there is significant improvement in some categories, the particular poor performance for the category chair is a reflection of the issues identified in the instance segmentation module. Another factor that impacts the chair category is the fixed number of proposals $K = 15$, while many scenes consist of up to $35 - 40$ instances of chairs. This restriction comes from the GTX 1070 GPU that we use, due to which many clusters are not processed by the network. Thus our results show

	bathtub	bed	bookshelf	chair	desk	dresser	night_stand	sofa	table	toilet	mAP
DSS [28]	44.2	78.8	11.9	61.2	20.5	6.4	15.4	53.5	50.3	78.9	42.1
COG [23]	58.3	63.7	31.8	62.2	**45.2**	15.5	27.4	51.0	51.3	70.1	47.6
2D-driven [13]	43.5	64.5	31.4	48.3	27.9	25.9	41.9	50.4	37.0	80.4	45.1
FPN [19]	43.3	81.1	33.3	64.2	24.7	32.0	58.1	61.1	51.1	90.9	54.0
VoteNet [18]	74.4	83.0	28.8	**75.3**	22.0	29.8	62.2	64.0	47.3	90.1	57.7
Ours (λ=10)	**79.4**	88.2	32.1	17.0	37.4	**53.7**	50.0	**65.3**	**53.3**	**95.8**	57.2
Multi-Scale EPN (standalone)	72.0	**90.5**	**62.2**	66.9	43.7	47.0	**62.6**	62.3	47.2	94.6	**64.9**

Table 3. Average precision for 3D amodal object detection, with IoU threshold 0.25, compared with state-of-the-art algorithms on the SUN-RGBD test dataset.

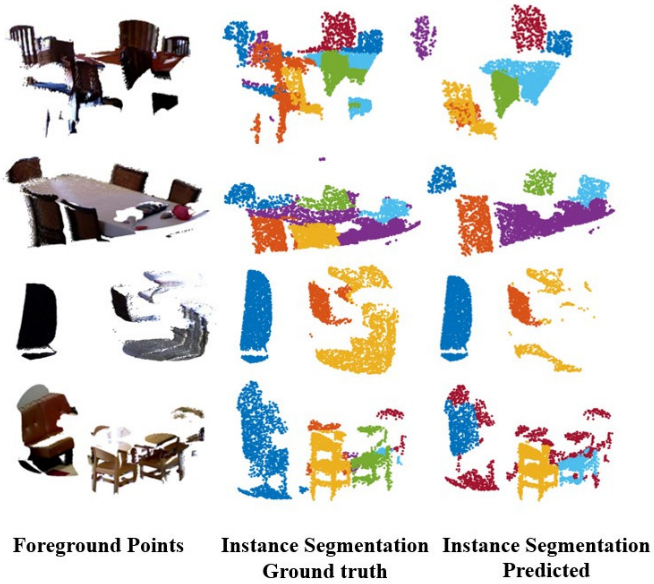
Foreground Points
Instance Segmentation Ground truth
Instance Segmentation Predicted

Figure 6. Qualitative results for class agnostic instance segmentation module. We observe that smaller objects, like chairs, are often incorrectly merged with larger objects, like table, that results in the significantly poor performance.

that accurate ROI generation and reducing the number of points are still critical factors that limit the performance of 3D object detectors.

4.3. Multi-Scale EPN

Since the parameter λ directly affects EPNs performance, we evaluate a Multi-scale EPN that combines the output from different scales and concatenates the binary maps at the input. However, for this experiment, we use ideal clusters that are directly generated using ground truth. Thus the objective is to evaluate multi-scale EPN as a standalone module and determine the effect it has on prediction accuracy. Consequently, the input tensor for the ECNN branch of the network is now defined as $X \rightarrow \mathbb{R}^{M\text{x}M\text{x}S_{\lambda}\text{x}3}$, where M is the resolution of a 2D binary map and S_{λ} defines the number of different scales used. Thus, the mapping generated by different λ is concatenated, while the rest of the architecture remains the same.

The results of this experiment are tabulated in Table 3, where $\lambda = \{5, 8, 10\}$. Our results show a performance improvement in the categories of bed, bookshelf, chair, desk and night stand of $\{2.3\%, 30.1\%, 49.9\%, 6.3\%, 12.6\%\}$, while achieving comparable AP for all other categories. The results of this experiment shows that with correctly clustered objects, EPN can achieve a much higher detection accuracy.

4.4. Execution Time

The training of every individual module took almost 48 hours on a single GTX 1070 GPU. Inference time for the background/foreground segmentation and instance segmentation module took an average of 0.58 and 0.45 seconds for a single point cloud of size 10000×3 and 5000×3 respectively. Inference for the EPN network takes 0.21 secs for 15 proposals (belonging to the same scene), with 1024 points and a $32x32x3$ binary image per object proposal. Overall the network takes 1.24 secs for inference on a single point cloud. While the inference time is much larger as compared to Frustum PointNet [19] and VoteNet [18], which take 0.09 and 0.1 secs respectively, a direct comparison is difficult as we use GTX 1070 while the authors in [18, 19] use GTX 1080.

5. Conclusion

In this paper we propose a cascaded framework of multiple deep networks that perform clustering based instance segmentation followed by 3D amodal object detection. Given that point based networks are limited by the number of points they can process, we propose a modular approach with a primary objective of filtering out irrelevant points so that the subsequent module focuses the receptive field on the desired objects. With this approach we propose a segmentation module that distinguishes foreground from background. This is followed by class agnostic instance segmentation that is initiated by an unsupervised clustering algorithm, learning to predict the offset of each point from the actual centroid to the proposed centroid. Finally all proposals are evaluated using the EPN module for predicting amodal object bounding boxes. We show how uncertainty estimates in the background/foreground segmentation results in improved accuracy for the task. Our results also show that the proposed approach achieves comparable results to the state-of-the-art on the SUN-RGBD dataset.

References

[1] S. M. Ahmed and C. M. Chew. Epn: Edge-aware pointnet for object recognition from multi-view 2.5d point clouds. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[2] S. M. Ahmed, Y. Z. Tan, C. M. Chew, A. Al Mamun, and F. S. Wong. Edge and corner detection for unorganized 3d point clouds with application to robotic welding. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7350–7355. IEEE, 2018.

[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE CVPR*, volume 1, page 3, 2017.

[4] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1355–1361. IEEE, 2017.

[5] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[6] Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[7] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[8] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[9] J. Hou, A. Dai, and M. Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4421–4430, 2019.

[10] P. J. Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992.

[11] A. Kanezaki, Y. Matsushita, and Y. Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[13] J. Lahoud and B. Ghanem. 2d-driven 3d object detection in rgb-d images. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 4632–4640. IEEE, 2017.

[14] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *arXiv preprint arXiv:1812.05784*, 2018.

[15] B. Li. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518. IEEE, 2017.

[16] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

[17] Q.-H. Pham, D. T. Nguyen, B.-S. Hua, G. Roig, and S.-K. Yeung. Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. *arXiv preprint arXiv:1904.00699*, 2019.

[18] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. *arXiv preprint arXiv:1904.09664*, 2019.

[19] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017.

[20] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[21] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016.

[22] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[23] Z. Ren and E. 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.

[24] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 3, 2017.

[25] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. *arXiv preprint arXiv:1812.04244*, 2018.

[26] M. Simon, S. Milz, K. Amende, and H.-M. Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *European Conference on Computer Vision*, pages 197–209. Springer, 2018.

[27] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *European conference on computer vision*, pages 634–651. Springer, 2014.

[28] S. Song and J. 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.

[29] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[30] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE Inter-*

national Conference on Computer Vision, pages 2686–2694, 2015.

[31] D. Z. Wang and I. Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, volume 1, 2015.

[32] W. Wang, R. Yu, Q. Huang, and U. Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018.

[33] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[34] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[35] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.