Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection

Zhixin Wang and Kui Jia

***Abstract*— In this work, we propose a novel method termed *Frustum ConvNet (F-ConvNet)* for amodal 3D object detection from point clouds. Given 2D region proposals in a RGB image, our method first generates a sequence of frustums for each region proposal, and uses the obtained frustums to group local points. F-ConvNet aggregates point-wise features as frustum-level feature vectors, and arrays these feature vectors as a feature map for use of its subsequent component of fully convolutional network (FCN), which spatially fuses frustum-level features and supports an end-to-end and continuous estimation of oriented boxes in the 3D space. We also propose component variants of L-ConvNet, including a FCN variant that extracts multi-resolution frustum features, and a refined use of L-ConvNet over a reduced 3D space. Careful ablation studies verify the efficacy of these component variants. L-ConvNet assumes no prior knowledge of the working 3D environment, and is thus dataset-agnostic. We present experiments on both the indoor SUN-RGBD and outdoor KITTI datasets. L-ConvNet outperforms all existing methods on SUN-RGBD, and at the time of submission it outperforms all published works on the KITTI benchmark. We will make the code of L-ConvNet publicly available.**

I. Introduction

Detection of object instances in 3D sensory data has tremendous importance in many applications including autonomous driving, robotic object manipulation, and augmented reality. Among others, RGB-D images and LiDAR point clouds are the most representative formats of 3D sensory data. In practical problems, these data are usually captured by viewing objects/scenes from a single perspective; consequently, only partial surface depth of the observed objects/scenes can be captured. The task of *amodal* 3D object detection is thus to estimate oriented 3D bounding boxes enclosing the full objects, given partial observation of the object surface. In this work, we focus on object detection from point clouds, and assume the availability of accompanying RGB images.

Due to the discrete, unordered, and possibly sparse nature of point clouds, detecting object instances from them is challenging and requires learning techniques that are different from the established ones [1]–[3] for object detection in RGB images. In order to leverage the expertise in 2D object detection, existing methods convert 3D point clouds either into 2D images by view projection [4]–[6], or into regular grids of voxels by quantization [7]–[10]. Although 2D object detection can be readily applied to the converted images or volumes, these methods suffer from loss of critical 3D information in the projection or quantization process.

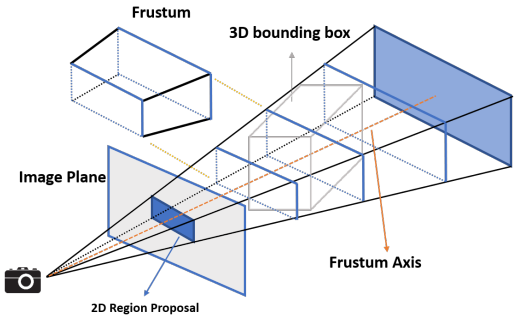

Fig. 1: Illustration for how a sequence of frustums are generated for a region proposal in a RGB image.

With the progress of point set deep learning [11], [12], recent methods [13], [14] resort to learning features directly from raw point clouds. For example, the seminal work of F-PointNet [13] first finds local points corresponding to pixels inside a 2D region proposal, and then uses PointNet [11] to segment from these local points the foreground ones; the amodal 3D box is finally estimated from the foreground points. Performance of this method is limited due to the reasons that (1) it is not of end-to-end learning, and (2) final estimation relies on too few foreground points which themselves are possibly segmented wrongly. Methods of VoxelNet style [14]–[16] overcome both of the above limitations by partitioning 3D point cloud into a regular grid of equally spaced voxels; voxel-level features are learned and extracted, again using methods similar to PointNet [11], and are arrayed together to form feature maps that are processed subsequently by convolutional (conv) layers; amodal 3D boxes are estimated in an end-to-end fashion using spatially convolved voxel-level features. For the other side of the coin, due to unawareness of objects, sizes and positions of grid partitioning in VoxelNet [14] methods do not take object boundaries into account, and their settings usually assume prior knowledge of the 3D environment (e.g., roads in the KITTI dataset [17]), which, however, are not always available.

Motivated to address the limitations in [13], [14], we propose in this paper a novel method of amodal 3D object detection termed *Frustum ConvNet (F-ConvNet)*. Similar to [13], our method assumes the availability of 2D region proposals in RGB images, which can be easily obtained from off-the-shelf object detectors [1], [2], and identifies 3D

Zhixin Wang and Kui Jia are with the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China. Email: wang.zhixin@mail.scut.edu.cn, kuijia@scut.edu.cn.

points corresponding to pixels inside each region proposal. Different from [13], our method generates for each region proposal a sequence of (possibly overlapped) *frustums* by sliding along the *frustum axis* [1] (cf. Fig. 1 for an illustration). These obtained frustums define groups of local points. Given the sequence of frustums and point association, our F-ConvNet starts with lower, parallel layer streams of PointNet style to aggregate point-wise features as a frustum-level feature vector; it then arrays *at its early stage* these feature vectors of individual frustums as 2D feature maps, and uses a subsequent fully convolutional network (FCN) to down-sample and up-sample frustums such that their features are fully fused across the frustum axis at a higher frustum resolution. Together with a final detection header, our proposed F-ConvNet supports an end-to-end and continuous estimation of oriented 3D boxes, where we also propose a FCN variant that extracts multi-resolution frustum features. Given an initial estimation of 3D box, a final refinement using the same F-ConvNet often improves the performance further. In this work, we present careful ablation studies that verify the efficacy of different components of L-ConvNet. On the SUN RGBD dataset [18], our method outperforms all existing ones. On the KITTI benchmark [17], our method outperforms all published works at the time of submission, including those working on point clouds and those working on a combination of point clouds and RGB images. We summarize our contributions as follows.

- We propose a novel method termed *Frustum ConvNet (F-ConvNet)* for amodal 3D object detection from point cloud. Given a sequence of frustums generated for each region proposal, F-ConvNet aggregates point-wise features as frustum-level feature vectors, and arrays *at its early stage* these feature vectors as a feature map for use of a subsequent FCN, which spatially fuses frustum-level features and supports an end-to-end and continuous estimation of oriented boxes in the 3D space.
- We also propose component variants of L-ConvNet, including a FCN variant that extracts multi-resolution frustum features, and a refined use of L-ConvNet over a reduced 3D space. Careful ablation studies verify the efficacy of these components and variants.
- L-ConvNet assumes no prior knowledge of the working 3D environment, and is thus dataset-agnostic. On the indoor SUN-RGBD dataset [18], L-ConvNet outperforms all existing methods; on the outdoor dataset of KITTI benchmark [17], it outperforms all published works at the time of submission.

II. Related Works

In this section, we briefly review existing methods of amodal 3D object detection. We organize our reviews into two categories of technical approaches, namely those based on conversion of 3D point clouds as images/volumes, and those admitting operation directly on raw point clouds.

Methods based on data conversion MV3D [5] projects LiDAR point clouds to bird eye view (BEV), and then employs a Faster-RCNN [2] for 3D object detection. AVOD [6] extends MV3D by aggregating the multi-modal features to generate more reliable 3D object proposals. Some existing methods also use depth images as converted data of point clouds. Deng et al. [19] directly estimate 3D bounding boxes from RGB-D images based on the Fast-RCNN framework [1]. Luo et al. [20] explore the SSD pipeline [3] to fuse RGB and depth image for 3D bounding box estimation. DSS [21] encodes a depth image as a grid of 3D voxels by TSDF, and uses 3D CNNs for classification and box estimation. PIXOR [9] also encodes point clouds as grids of voxels. The above methods based on data conversion can leverage the expertise in mature 2D detection, but the projection or quantization process would cause loss of critical information.

Methods working on raw point clouds We have reviewed in introduction the seminal work of [13] that works directly on raw point clouds but is not of end-to-end learning, and the methods of VoxelNet style [14]–[16] that resolve this issue but with the shortcoming of object unawareness in 3D point clouds. We note that PointPillars [16] explores pillar shape instead of voxel design to aggregate point-wise features. For multi-stage methods [13], [22], [23], subsequent works of IPOD [22] and PointRCNN [23] explore different proposal methods. Our proposed F-ConvNet is motivated and designed to combine the benefits of both worlds.

III. The Proposed Frustum ConvNet

In this section, we present our proposed Frustum ConvNet (F-ConvNet) that supports end-to-end learning of amodal 3D object detection. Design of F-ConvNet centers on the notion of *square frustum*, and a sequence of frustums along the same *frustum axis* connect a cloud of discrete, unordered points with an FCN that enables oriented 3D box estimation in a continuous 3D space. Fig.2 give an illustration. By assuming the availability of 2D region proposals in RGB images, we will first introduce our way of point association with sequences of (possibly overlapped) frustums that are obtained by sliding along frustum axes determined by 2D region proposals, and compare with alternative ways of point association/grouping. We will then present the architecture of F-ConvNet, and specify how point-wise features inside individual frustums are aggregated and re-formed as 2D feature maps for a continuous frustum-level feature fusion and 3D box estimation. We finally explain how an F-ConvNet can be trained using losses borrowed from the literature of 2D object detection.

A. Associating Point Cloud with Sequences of Frustums

Learning semantics from point clouds grounds on extraction of low-level geometric features that are defined over local groups of neighboring points. Due to the discrete, unordered nature of point cloud, there exists no an oracle way to associate individual points into local groups. In the

[1]For any image region, a square pyramid passing though the image region can be specified by the viewing camera and the farthest plane that is perpendicular to the optical axis of the camera. Starting from the image plane, a *frustum* is formed by truncating the pyramid with a pair of parallel planes perpendicular to the optical axis, which is also the *frustum axis*.

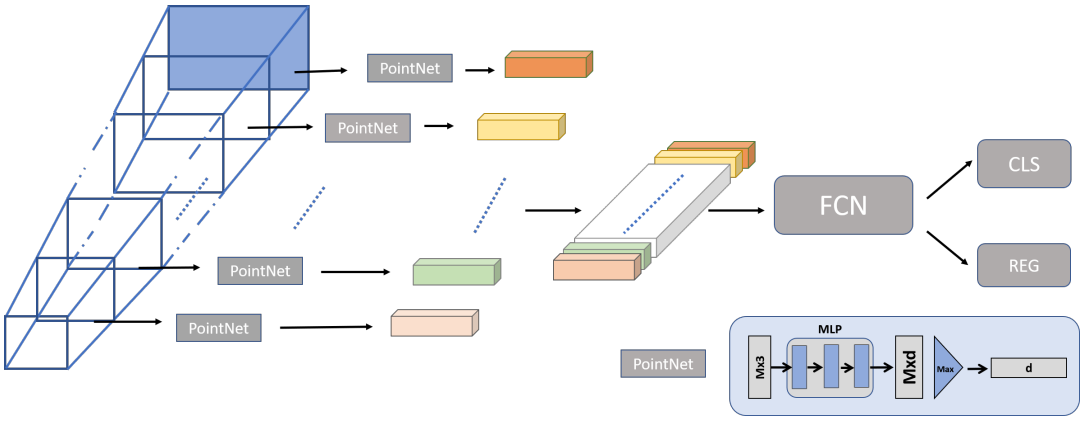

Fig. 2: The whole framework of our F-ConvNet. We group points and extract features by PointNet from a sequence of frustums, and frustum-level features are re-formed as 2D feature map for our fully convolutional network (FCN) and detection header (CLS and REG) for 3D box estimation.

literature of point set classification/segmentation [12], [24], local groupings are formed by searching nearest neighbors, with seed points sampled from a point cloud using farthest point sampling (FPS). FPS can efficiently cover a point cloud, but it is unaware of object positions; consequently, grouping based on FPS is not readily useful for tasks concerning with detection of object instances from a point cloud. Rather than sampling seed points for local groupings, methods of VoxelNet style [14]–[16] define a regular grid of equally spaced voxels in the 3D space, and points falling in same voxels are grouped together. Although voxels may densely cover the entire 3D space, their sizes and grid positions do not take object boundaries into account. In addition, their settings usually assume prior knowledge of the 3D environment and the contained object categories (e.g., car and pedestrian in the KITTI dataset [17]), which, however, are not always available.

To address these limitations, we are inspired by [13] and propose the following scheme to group local points. We assume that a RGB image is available accompanying the 3D point cloud, and that 2D region proposals are also provided by off-the-shelf object detectors [1]–[3]. A sequence of (possibly overlapped) frustums can be obtained by sliding a pair of parallel planes along the frustum axis with an equal stride, where the pair of planes are also perpendicular to the frustum axis. Fig.3 gives an illustration. In this work, we generate such a sequence of frustums for each 2D region proposal, and we use thus obtained frustum sequences to group points, i.e., points falling inside same frustums are grouped together. Assuming that 2D region proposals are accurate enough, our frustums mainly contain foreground points, and are aware of object boundaries. We note that for each 2D region proposal, a single frustum of larger size (defined by the image plane and the farthest plane) is generated in [13] and all points falling inside this frustum are grouped together; consequently, an initial stage of foreground point segmentation has to be performed before amodal 3D box estimation. In contrast, we generate for each region

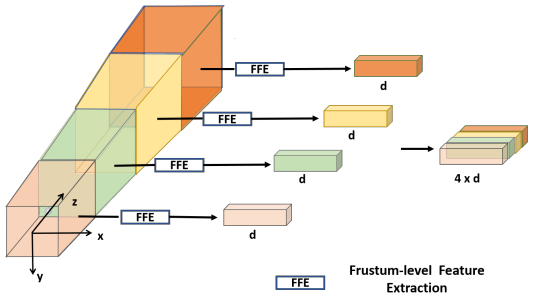

Fig. 3: A toy illustration of our frustums (coded as different colors) for point association and frustum-level feature aggregation. A sequence of non-overlapped frustums are shown here for simplicity.

proposal a sequence of frustums whose feature vectors are arrayed as a feature map and used in a subsequent FCN for an end-to-end estimation of oriented boxes in the continuous 3D space, as presented shortly.

B. The Architecture of Frustum ConvNet

Given a sequence of frustums generated from a region proposal, the key design of an F-ConvNet is to aggregate *at its early stage* point-wise features inside each frustum as a frustum-level feature vector, and then array as a 2D feature map these feature vectors of individual frustums for use of a subsequent FCN, which, together with a detection header, supports an end-to-end and continuous estimation of oriented 3D boxes. Fig. 2 gives the architecture. We present separate components of the F-ConvNet as follows.

Frustum-level feature extraction via PointNet

For a 2D region proposal in a RGB image, assume a sequence of T frustums of height u are generated by sliding along the frustum axis with a stride s. For any one of them, assume it contains M local points whose coordinates in the camera coordinate system are denoted as $\{\mathbf{x}_i = (x_i, y_i, z_i)\}_{i=1}^{M}$ — for simplicity, we assume optical axis of the camera is perpendicular to this 2D region, which suggests an initial

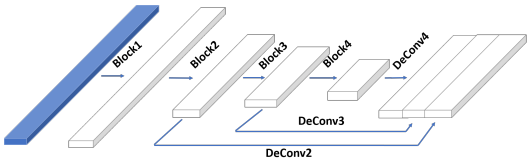

Fig. 4: The architecture of fully convolutional network (FCN) in Frustum ConvNet. Each convolutional layer is followed by Batch Normalization and ReLU nonlinearity. Blue color represents 2D feature map of arrayed frustum-level feature vectors.

adjustment of camera view has already been performed, as illustrated in Fig.3. To learn and extract point-wise features, we use PointNet [11] in this work that stacks several fully-connected (FC) layers, followed by a final layer that aggregates features of individual points as a frustum-level feature vector via element-wise max pooling. We note that choices other than PointNet are applicable as well, such as PointCNN [24]. Instead of using $\{\mathbf{x}_i\}_{i=1}^{M}$ as input of PointNet directly, we use relative coordinates $\{\bar{\mathbf{x}}_i = (\bar{x}_i, \bar{y}_i, \bar{z}_i)\}_{i=1}^{M}$ that are obtained by subtracting each $\mathbf{x}_i$ with the centroid $\mathbf{c}$ of the frustum, i.e., $\bar{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{c}$ for $i = 1, \ldots, M$.

We apply a PointNet to each frustum, and the T duplicate PointNets, with shared weights, form the lower, parallel streams of our L-ConvNet. Our used PointNet consists of three FC layers which respectively output features of dimensions $d/2$, $d/2$, and d.

Fully convolutional network

Denote the extracted frustum-level feature vectors as $\{\mathbf{f}_i\}_{i=1}^{L}$, with $\mathbf{f}_i \in \mathbb{R}^d$. We array these L vectors to form a 2D feature map $\mathbf{F}$ of the size $L \times d$, which will be used as input of a subsequent FCN. As shown in Fig.4, our FCN consists of blocks of conv layers, and de-conv layers corresponding to each block. Convolution in conv layers is applied across the frustum dimension by using kernels of the size $3 \times d$. The final layer of each of the conv blocks, except the first block, also down-samples (halves) the 2D feature map at the frustum dimension by using stride-2 convolution. Convolution and down-sampling fuse features across frustums and produce at different conv blocks *virtual frustums* of varying heights (along the frustum axis direction). Given output feature map of each conv block, a corresponding de-conv layer is used that up-samples at the frustum dimension the feature map to a specified (higher) resolution $\tilde{L}$; outputs of all de-conv layers are then concatenated together along the feature dimension. It outputs a feature map of the size $\tilde{L} \times 3d$. Feature concatenation from virtual frustums of varying sizes provides a hierarchical granularity of frustum covering, which would be useful to estimate 3D boxes of object instances whose sizes are unknown and vary. In this work, we use an FCN of 4 conv blocks and 3 de-conv layers for KITTI, and an FCN of 5 conv blocks and 4 de-conv layers for SUN-RGBD. Layer specifics of these FCNs are given in the appendix.

A multi-resolution frustum feature integration variant

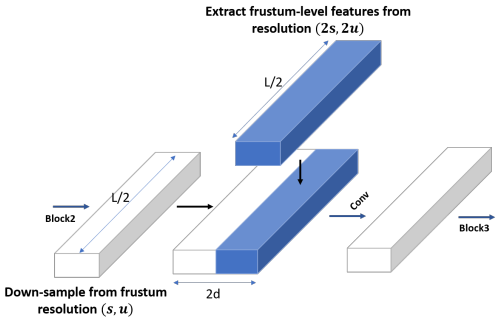

Fig. 5: Illustration of our multi-resolution frustum feature integration. We show an example between Block2 and Block3. We apply it between Block3 and Block4, Block4 and DeConv4. Including first resolution, we have in total four kinds of resolution in KITTI dataset.

We already know that output feature maps of conv blocks in FCN are of reduced resolutions by a power of 2 at the frustum dimension. Take the one with the size of $L/2 \times d$ as the example. For the same 2D region proposal, a new sequence of $T/2$ frustums can be generated by sliding along the frustum axis with a stride $2s$. Applying PointNet to each of the generated frustums and arraying the resulting feature vectors produce a new feature map of the same size $L/2 \times d$. When frustum height is doubled as $2u$, the new sequence covers the same 3D space at a half coarser resolution, while its feature map being compatible with its corresponding one in FCN. We then concatenate along the feature dimension the two feature maps of the same size, giving rise to a new one of the size $L/2 \times 2d$. A final conv layer is used to resize it back as a feature map of size $L/2 \times d$, so that it can be placed back in FCN with no change of other FCN layers. Fig.5 illustrates the above procedure. The procedure can be used for each down-sampled feature maps in FCN. We refer to this scheme as a multi-resolution frustum feature integration variant of F-ConvNet. Ablation studies in Section IV-B verify its efficacy.

C. Detection Header and Training of Frustum ConvNet

On top of FCN is the detection header composed of two, parallel conv layers, as shown in Fig.2. They are respectively used as the classification and regression branches. The whole F-ConvNet is trained using a multi-task fashion, similar to those in 2D object detection [1], [2].

Suppose we have K object categories. The classification branch is trained to output a $\tilde{L} \times (K+1)$ frustum-wise probability map of object categories, plus the background one. In this work, we use focal loss [25] for classification branch to cope with imbalance of foreground and background samples.

Ground truth of oriented 3D bounding boxes is parameterized as $\{x_c^g, y_c^g, z_c^g, l^g, w^g, h^g, \theta^g\}$, where $\{x_c^g, y_c^g, z_c^g\}$ denote coordinates of box centroid, $\{l^g, w^g, h^g\}$ denote three side lengths of the box, and θ^g denotes the yam angle that means the in-plane rotation perpendicular to the gravity direction.

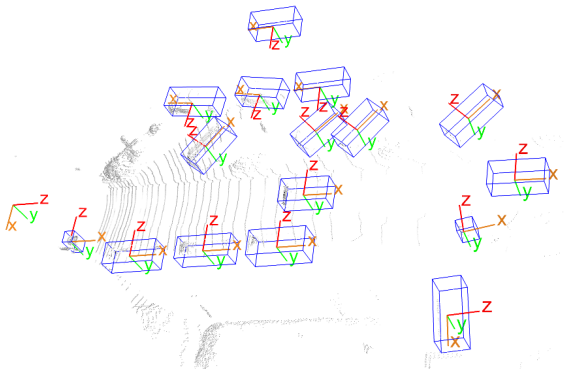

Fig. 6: Normalization of point coordinates for refinement Network. Leftmost is the camera coordinate system. Normalized coordinate systems are shown with each cuboid box, where origin of each system is located at each cuboid center and its x-axis is aligned with optical axis of the camera.

We discretize the range $[-\pi, \pi)$ of yam angles into N bins, and define for each frustum KN anchor boxes, i.e., N ones per foreground category. For any one of them parameterized as $\{x_c^a, y_c^a, z_c^a, l^a, w^a, h^a, \theta^a\}$, we use centroid of the frustum as $\{x_c^a, y_c^a, z_c^a\}$, compute from training samples the category-wise averages of the three side lengths as $\{l^a, w^a, h^a\}$, and set θ^a as one of the bin centers of yam angles. This gives the following offset formulas:

$$\begin{aligned} &\Delta x = x_c^g - x_c^a, \Delta y = y_c^g - y_c^a, \Delta z = z_c^g - z_c^a, \\ &\Delta l = \frac{l^g - l^a}{l_a}, \Delta w = \frac{w^g - w^a}{w_a}, \Delta h = \frac{h^g - h^a}{h^a}, \\ &\Delta\theta = \theta^g - \theta^a. \end{aligned} \tag{1}$$

Regression loss is based on the Euclidean distance of the above offsets. Besides, we also use a corner loss [13] to regularize box regression of all parameters. Together with the focal loss for classification branch, the whole F-ConvNet is trained using a total of three losses.

D. Final Refinement

We have assumed for now that the 2D region proposals are accurate enough. In practice, region proposals provided by object detectors do not bound object instances precisely. As a remedy, we propose a refinement that applies the same F-ConvNet architecture to points falling inside the oriented 3D boxes that we have just estimated. Specifically, we first expand each estimated box by a specified factor — we set the factor as 1.2 in this work, and normalize points inside the expanded box by translation and rotation, as shown in Fig. 6. These normalized points are used as input of a second F-ConvNet for a final refinement. Ablation studies show that this refinement well resolves the issue caused by inaccurate 2D object proposals.

IV. Experiments

A. Datasets and Implementation Details

KITTI The KITTI dataset [17] contains 7,481 training pairs and 7,518 testing pairs of RGB images and point clouds of three object categories (i.e., Car, Pedestrian, and Cyclist). For each category, detection results are evaluated based on three levels of difficulty (i.e., easy, moderate, and hard). We train two separate F-ConvNets respectively for Car and Pedestrian/Cyclist. Since ground truth of the test set is unavailable, we follow existing works [5] and split the original training set into the new training and validation ones respectively of 3,712 and 3,769 samples. We conduct our ablation experiments using this splitting, and our final results on the KITTI test set are obtained by server submission. We use the official 3D IoU evaluation metrics of 0.7, 0.5, and 0.5 respectively for the categories of Car, Cyclist, and Pedestrian.

SUN-RGBD The SUN-RGBD dataset [18] contains 10,355 RGB-D images (5,285 training ones and 5,050 testing ones) of 10 object categories. We convert the depth images as point clouds for use of our method. Results are evaluated on the 10 categories under 0.25 3D IoU threshold. For this dataset, we do not use the final refinement of our method.

Implementation details Our use of 2D object detectors is as follows. For the KITTI dataset, we use RRC [26] for the car category and MSCNN [27] for the pedestrians and cyclist categories. We directly use the release models provided by these methods. As for SUN-RGBD, we train Faster-RCNN [2] with the backbone network of ResNet-50 [28]. We do data augmentation to the obtained 2D region proposals by translation and scaling. We randomly sample from 3D points corresponding to each region proposal to have a fixed number 1,024 for KITTI and 2,048 for SUN-RGBD. For final refinement, we use a fixed number of 512. We also do random flipping and shifting to these points, similar to [13].

To prepare positive and negative training samples, we shrink ground-truth boxes by a ratio of 0.5, and count anchor boxes whose centers fall in the shrunken ground-truth boxes as foreground ones, count the others as background. We ignore the anchor boxes whose centers fall between the shrunken boxes and ground-truth boxes. We consider a depth range of [0, 70] meters in KITTI and that of [0, 8] in SUN-RGBD. We train F-ConvNets with a mini-batch size 32 on 1 GPU. We use ADAM optimizer with weight decay of 0.0001. Learning rates start from 0.001 and decay by a factor of 10 every 20^{th} epoch of the total 50 epoches. For KITTI, we use 4 frustum resolutions of $u = [0.25, 0.5, 1.0, 2.0]$ and $s = [0.25, 0.5, 1.0, 2.0]$ for the car category, with $d = 128, L = 280$, and $\tilde{L} = 140$, and 4 frustum resolutions of $u = [0.1, 0.2, 0.4, 0.8]$ and $s = [0.1, 0.2, 0.4, 0.8]$ for the pedestrian and cyclist categories, with $d = 128, L = 560$, and $\tilde{L} = 280$. For SUN-RGBD, we use 5 frustum resolutions of $u = [0.1, 0.2, 0.4, 0.8, 1.6]$ and $s = [0.1, 0.2, 0.4, 0.8, 1.6]$, with $d = 128, L = 80$, and $\tilde{L} = 40$.

B. Ablation studies

In this section, we verify components and variants of our proposed F-ConvNet by conducting ablation studies on the train/val split of KITTI. We follow the convention and use the car category that contains the most training examples. Before

Method	Easy	Moderate	Hard
MV3D [5]	71.29	62.68	56.56
VoxelNet [14]	81.97	65.46	62.85
F-PointNet [13]	83.76	70.92	63.65
AVOD-FPN [6]	84.41	74.44	68.65
ContFusion [10]	86.32	73.25	67.81
IPOD [22]	84.1	76.4	75.3
PointRCNN [23]	88.88	78.63	**77.38**
Ours	**89.02**	**78.80**	77.09

TABLE I: 3D object detection AP on KITTI val set.

Method	Easy	Moderate	Hard
MV3D [5]	86.55	78.10	76.67
VoxelNet [14]	89.60	84.81	78.57
F-PointNet [13]	88.16	84.92	76.44
ContFusion [10]	**95.44**	87.34	82.43
IPOD [22]	88.3	86.4	84.6
Ours	90.23	**88.79**	**86.84**

TABLE II: BEV detection AP on KITTI val set.

individual studies, we first report in Tab.I and Tab.II our results of 3D detection and BEV detection on the validation set. For the most important "Moderate" column, our method outperforms existing ones on both of the two tasks.

Influence of 2D region proposal Our method relies on accuracy of 2D region proposals. To investigate how much it affects the performance, we use three 2D object detectors with increased practical performance, namely a baseline Faster-RCNN [2] with a backbone network of ResNet-50 [28], a detector provided by [13], and an oracle one of ground-truth 2D boxes. Results in Tab.III confirm that better performance of 2D detection positively affects our method.

2D Detection			3D Detection		
Easy	Moderate	Hard	Easy	Moderate	Hard
96.19	87.51	77.41	85.19	74.05	64.77
96.48	90.31	87.63	86.51	76.57	68.17
100.00	100.00	100.00	87.68	85.47	78.19

TABLE III: Influence of 2D region proposal. Each line corresponds to results from a different 2D object detector.

Effect of frustum feature extractor We use PointNet [11] to extract and aggregate point-wise features as frustum-level feature vectors. Other choices such as PointCNN [24] are applicable as well. To compare, we replace the element-wise max pooling in PointNet by the X-Conv operation in PointCNN for feature aggregation. Tab.IV shows that PointCNN is also a possible choice of frustum feature extractor; however, its performance is not necessarily better than the simple PointNet. We use one resolution for this experiment.

	Easy	Moderate	Hard
PointNet	84.09	75.32	67.45
PointCNN	81.91	73.83	66.37

TABLE IV: Comparison between frustum feature extractors.

Effect of the multi-resolution frustum feature integration variant To investigate the effect of this variant, we plug in various resolution combinations into F-ConvNet. Results in Tab.V confirm the efficacy.

s1	s2	s3	s4	Easy	Moderate	Hard
✓				84.09	75.32	67.45
✓	✓			84.19	74.88	66.95
✓		✓		85.41	75.63	67.44
✓			✓	86.12	76.04	67.97
✓	✓	✓		86.21	76.12	67.96
✓		✓	✓	86.69	76.30	68.02
✓	✓	✓	✓	86.51	76.57	68.17

TABLE V: Investigation of the multi-resolution frustum feature integration variant.

Effect of Focal Loss and Final Refinement We use focal loss to cope with imbalance of foreground and background training samples. We also propose a final refinement step to cope with less accurate 2D region proposals. The effects of these two components are clearly demonstrated in Tab.VI.

	Easy	Moderate	Hard
w/o FL and w/o RF	83.78	74.05	65.96
w/o RF	86.51	76.57	68.17
Ours	89.02	78.80	77.09

TABLE VI: Effects of focal loss (FL) and final refinement (RF).

C. Comparisons with The State of The Art

The KITTI Results Tab.VII shows the performance of our method on the KITTI test set, which is obtained by server submission. Our method outperforms all existing published works, and at the time of submission it ranks 4^{th} on the KITTI leaderboard. We also show performance of our method on 3D object localization in Tab. VIII. For this detection task, the 3D bounding boxes are projected to bird-eye view plane and IoU is evaluated on oriented 2D boxes. Representative results of our method are visualized in Fig.7.

The SUN-RGBD Results We also apply our proposed F-ConvNet to the indoor environment of SUN-RGBD. Our results in Tab.IX are better than those of all existing methods, showing the general usefulness of our proposed method.

V. Conclusion

We have presented a novel method of Frustum-ConvNet for amodal 3D object detection in an end-to-end and continuous fashion. The proposed method is dataset-agnostic and demonstrates state-of-the-art performance on both the indoor SUN-RGBD and outdoor KITTI datasets. The method is useful for many applications such as autonomous driving and robotic object manipulation. In future research, we will investigate more seamless ways of integrating point-wise and RGB features and we expect even better performance would be achieved.

Method	Cars			Pedestrians			Cyclists		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D [5]	71.09	62.35	55.12	-	-	-	-	-	-
VoxelNet [14]	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
F-PointNet [13]	81.20	70.29	62.19	51.21	44.89	40.23	71.96	56.77	50.39
AVOD-FPN [6]	81.94	71.88	66.38	50.80	42.81	40.88	64.00	52.18	46.61
SECOND [15]	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
IPOD [22]	79.75	72.57	66.33	**56.92**	44.68	**42.39**	71.40	53.46	48.34
PointPillars [16]	79.05	74.99	68.30	52.08	43.53	41.49	75.78	59.07	52.92
PointRCNN-v1.1 [23]	**85.94**	75.76	**68.32**	49.43	41.78	38.63	73.93	59.60	53.59
Ours	85.88	**76.51**	68.08	52.37	**45.61**	41.49	**79.58**	**64.68**	**57.03**

TABLE VII: 3D object detection AP (%) on KITTI test set.

Method	Cars			Pedestrians			Cyclists		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D [5]	86.02	76.90	68.49	-	-	-	-	-	-
VoxelNet [14]	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
F-PointNet [13]	88.70	84.00	75.33	58.09	50.22	**47.57**	75.38	61.96	54.68
AVOD-FPN [6]	88.53	83.79	77.90	58.75	51.05	47.54	68.06	57.48	50.77
SECOND [15]	88.07	79.37	77.95	55.10	46.27	44.76	73.67	56.04	48.78
IPOD [22]	86.93	83.98	77.85	**60.83**	**51.24**	45.40	77.10	58.92	51.01
PointPillars [16]	88.35	**86.10**	**79.83**	58.66	50.23	47.19	79.14	62.25	56.00
PointRCNN-v1.1 [23]	89.47	85.68	79.10	55.92	47.53	44.67	81.52	66.77	**60.78**
Ours	**89.69**	83.08	74.56	58.90	50.48	46.72	**82.59**	**68.62**	60.62

TABLE VIII: 3D object localization AP (BEV) (%) on KITTI test set.

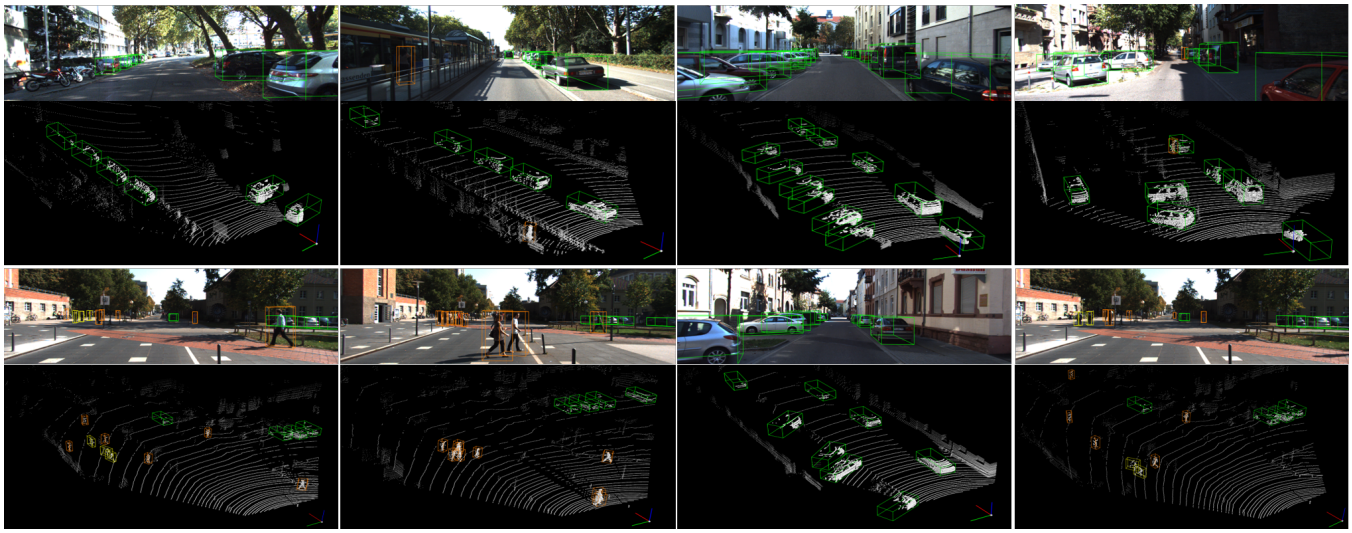

Fig. 7: Qualitative results on the KITTI test set (best view in color with zoom-in). Different color bounding boxes denote different categories, with green for car, orange for pedestrian, and yellow for cyclist.

Method	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	soft	table	toilet	mean
DSS [21]	44.2	78.8	11.9	61.2	20.5	6.4	15.4	53.5	50.3	78.9	42.1
COG [29]	58.26	63.67	31.80	62.17	**45.19**	15.47	27.36	51.02	51.29	70.07	47.63
2Ddriven3D [30]	43.45	64.48	31.40	48.27	27.93	25.92	41.92	50.39	37.02	80.40	45.12
PointFusion [31]	37.26	68.57	**37.69**	55.09	17.16	23.95	32.33	53.83	31.03	83.80	45.38
Ren et al. [32]	**76.2**	73.2	32.9	60.5	34.5	13.5	30.4	60.4	**55.4**	73.7	51.0
F-PointNet [13]	43.3	81.1	33.3	64.2	24.7	32.0	58.1	61.1	51.1	**90.9**	54.0
Ours	61.32	**83.19**	36.46	**64.40**	29.67	**35.10**	**58.42**	**66.61**	53.34	86.99	**57.55**

TABLE IX: 3D detection results on the SUN-RGBD test set (IoU 0.25).

REFERENCES

[1] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[4] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," in *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

[5] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *IEEE CVPR*, vol. 1, no. 2, 2017, p. 3.

[6] 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)*. IEEE, 2018, pp. 1–8.

[7] B. Li, "3d fully convolutional network for vehicle detection in point cloud," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 1513–1518.
[8] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection." in *Robotics: Science and Systems*, vol. 1, no. 3, 2015.
[9] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
[10] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 641–656.
[11] 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*, vol. 1, no. 2, p. 4, 2017.
[12] 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*, 2017, pp. 5105–5114.
[13] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 918–927.
[14] 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*, 2018, pp. 4490–4499.
[15] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.
[16] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Encoders for object detection from point clouds," *arXiv preprint arXiv:1812.05784*, 2018.
[17] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
[18] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite." vol. 5, p. 6, 2015.
[19] Z. Deng and L. J. Latecki, "Amodal detection of 3d objects: Inferring 3d bounding boxes from 2d ones in rgb-depth images," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017, p. 2.
[20] Q. Luo, H. Ma, Y. Wang, L. Tang, and R. Xiong, "Single multi-feature detector for amodal 3d object detection in rgb-d images," *arXiv preprint arXiv:1711.00238*, 2017.
[21] 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*, 2016, pp. 808–816.
[22] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "Ipod: Intensive point-based object detector for point cloud," *arXiv preprint arXiv:1812.05276*, 2018.
[23] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," *arXiv preprint arXiv:1812.04244*, 2018.
[24] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in Neural Information Processing Systems*, 2018, pp. 828–838.
[25] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.
[26] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, "Accurate single stage detector using recurrent rolling convolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5420–5428.
[27] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European Conference on Computer Vision*. Springer, 2016, pp. 354–370.
[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[29] 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*, 2016, pp. 1525–1533.
[30] J. Lahoud and B. Ghanem, "2d-driven 3d object detection in rgb-d images," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4632–4640.
[31] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 244–253.
[32] Z. Ren and E. B. Sudderth, "3d object detection with latent support surfaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 937–946.

APPENDIX

Layer specifics of the FCN component of L-ConvNet for the KITTI and SUN-RGBD datasets are respectively given in Tab.X and Tab.XI.

Name	Kernel size/Filter no./Striding/Padding
Block1	3×128 / 128 / 1 / 1
Block2	3×128 / 128 / 2 / 1 3×128 / 128 / 1 / 1
Block3	3×128 / 256 / 2 / 1 3×256 / 256 / 1 / 1
Block4	3×256 / 512 / 2 / 1 3×512 / 512 / 1 / 1
Deconv2	1×128 / 256 / 1 / 0
Deconv3	2×256 / 256 / 2 / 0
Deconv4	4×512 / 256 / 4 / 0

TABLE X: Layer specifics of the FCN component of L-ConvNet for KITTI.

Name	Kernel size/Filter no./Striding/Padding
Block1	3×64 / 64 / 1 / 1
Block2	3×64 / 128 / 2 / 1 3×128 / 128 / 1 / 1
Block3	3×128 / 256 / 2 / 1 3×256 / 256 / 1 / 1
Block4	3×256 / 512 / 2 / 1 3×512 / 512 / 1 / 1
Block5	3×512 / 512 / 2 / 1 3×512 / 512 / 1 / 1
Deconv2	1×128 / 256 / 1 / 0
Deconv3	2×256 / 256 / 2 / 0
Deconv4	4×512 / 256 / 4 / 0
Deconv5	8×512 / 256 / 8 / 0

TABLE XI: Layer specifics of the FCN component of L-ConvNet for SUN-RGBD.