2021 IEEE Intelligent Transportation Systems Conference (ITSC)
Indianapolis, USA. September 19-21, 2021

Faraway-Frustum: Dealing with Lidar Sparsity for 3D Object Detection using Fusion

Haolin Zhang*, Dongfang Yang*, Ekim Yurtsever*, Keith A. Redmill and Ümit Özgüner

***Abstract*— Learned pointcloud representations do not generalize well with an increase in distance to the sensor. For example, at a range greater than 60 meters, the sparsity of lidar pointclouds reaches a point where even humans cannot discern object shapes from each other. However, this distance should not be considered very far for fast-moving vehicles: a vehicle can traverse 60 meters in under two seconds while moving at 70 mph. For safe and robust driving automation, acute 3D object detection at these ranges is indispensable. Against this backdrop, we introduce faraway-frustum: a novel fusion strategy for detecting faraway objects. The main strategy is to depend solely on the 2D vision sensor for recognizing and classifying an object, as object shape does not change drastically with an increase in depth, and use pointcloud data for object localization in the 3D space for faraway objects. For closer objects, we use learned pointcloud representations instead, following state-of-the-art practices. This strategy alleviates the main shortcoming of object detection with learned pointcloud representations. Experiments on the KITTI dataset demonstrate that our method outperforms state-of-the-art methods by a considerable margin for faraway object detection in bird's-eye view and 3D. Our code is open-source and publicly available: `https://github.com/dongfang-steven-yang/faraway-frustum`.**

I. INTRODUCTION

3D/bird's-eye view (BEV) object detection is a critical task for many robotics applications. Existing lidar-based methods show good performance for close to medium range objects. However, a closer look at the state-of-the-art exposes an inherent problem: learned pointcloud representations do not generalize well with an increase in sparsity. This is not a surprising phenomenon. At a range greater than sixty meters, lidar pointcloud sparsity reaches a point where even humans cannot discern object shapes from each other. For example, in the KITTI 3D/BEV object detection benchmark [1], the state-of-the-art 3D object detection performance is remarkable. But when these high performing models face objects that are located at 60 meters and beyond, mean average precision drops to almost *zero*. We believe this is an important issue for automated driving. For instance, detecting faraway objects can offer more time for the automated vehicle to make better decisions.

3D/BEV object detection for faraway objects is challenging, and state-of-the-art (SOTA) lidar-based detectors [2], [3], [4], [5], [6] do not perform well for this task. We believe this is caused by sparsity and near-random scattering of the few points obtained from faraway objects. Learned representations from close to medium range objects do not generalize to faraway cases, and since SOTA approaches are primarily deep neural networks, they cannot learn the representations of faraway cases.

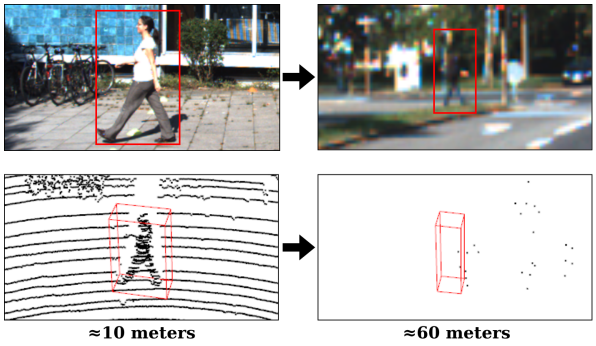

Fig. 1. Learned pointcloud representations do not generalize well with an increase in sparsity. This problem does not translate to the 2D RGB image domain in the same fashion, as object shape does not change drastically with an increase in depth. However, sparse points in the target object's vicinity can still be used to estimate depth. Our method utilizes these sparse points to estimate depth while using 2D RGB information to recognize shape and object-class.

RGB-pointcloud fusion is a common strategy [7], [8], [9], [10], [11], [12] to increase 3D object detection performance. For example, some works [7], [12] focus on using 2D detection results to generate frustum-based search spaces in pointclouds. As shown in Fig. 1, a faraway object in the RGB image domain usually contains around 400 pixels, which can be easier to recognize with a mature 2D detector. As such, a fusion-based approach can be a good candidate for faraway object detection. However, even though the aforementioned studies use RGB imagery to boost detection performance, they still depend exclusively on learned pointcloud representations to localize objects in 3D.

In this work, we propose an alternative 3D/BEV detector, *Faraway-Frustum*, to address the problem of faraway object detection. We follow the idea of frustum generation but use clustering instead of a neural network to estimate an initial object location in the cropped pointcloud for faraway objects. We still train a neural network to regress bounding box shape and refine depth. The overview of the proposed method is shown in Fig. 2. We first use 2D instance segmentation masks (or 2D bounding boxes) for each object in the RGB image space to generate frustums in the pointcloud space and find the corresponding lidar points for each object. Then, a pointcloud clustering technique is applied to estimate the 3D centroid of the object. By comparing the centroid distance with a faraway threshold, a decision is made to treat an

*Equal contribution

Authors are with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210, USA. Contact: [zhang.10749, yang.3455, yurtsever.2, redmill.1, ozguner.1] @osu.edu

2021 IEEE International Intelligent Transportation Systems Conference (ITSC) | 978-1-7281-9142-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ITSC48978.2021.9564990

object as either faraway or nearby. If faraway, a 3D bounding box is regressed by our Faraway Frustum Network (FF-Net) to the object based on the estimated centroid and the frustum pointcloud. Otherwise, instead of clustering the raw pointcloud, learned representations are directly used for 3D box fitting, following SOTA practices.

To evaluate the proposed method, we conducted benchmarking experiments using the KITTI dataset [1]. In KITTI, the average number of lidar points for each faraway object (e.g. pedestrians over 60 meters and cars over 75 meters) is ten or less, which supports our motivating assertion that an alternative approach is necessary for detecting faraway objects instead of directly using pointcloud-driven neural network approaches. The experimental results demonstrate that our method outperforms SOTA methods on faraway object detection, which indicates that our method effectively fuses the RGB data with a very sparse pointcloud. As shown in Fig. 6, our proposed method successfully detects faraway objects where SOTA methods (fusion or pointcloud only) fail. Our main contributions can be summarized as follows:

- Introduction of a novel fusion strategy: depending solely on 2D vision sensing for object-class recognition and using frustum-cropped pointcloud data with clustering for 3D object localization.
- Showing that using clustering with cropped, very sparse raw pointcloud data is a better strategy than using learned representations for faraway 3D object detection. As shown in Fig. 1, within very sparse pointclouds, the shape of objects changes drastically and randomly. As such, using representations learned mostly from closer objects is not effective.
- Demonstrating the failure of state-of-the-art 3D object detectors with objects at a distance over sixty meters in the KITTI dataset. The proposed faraway-frustum approach outperforms SOTA methods with a significant margin.

II. RELATED WORK

In this section, we briefly review state-of-the-art 3D/BEV object detection methods. We divide them into two main categories: pointcloud only methods and RGB-pointcloud fusion methods. In the second category, we mainly discuss feature-based fusion and frustum-based fusion. We also discuss their performance in detecting faraway objects.

Pointcloud only methods. One way of processing pointcloud data is based on voxels [3], [4], [2], [13]. Such methods first convert the pointcloud into voxel grids and then learn the representation of each voxel. 3D/BEV detection is achieved with the learned voxel representation. Alternatively, raw pointcloud data can be used for 3D/BEV object detection [14], [15], [6] by directly utilizing PointNet-based architectures [16]. These methods are robust for most objects. However, pointcloud only methods all have difficulty detecting faraway objects, because the lidar points of faraway objects are too sparse to be voxelized and learned, leading to no detection result in most cases.

Feature-based Fusion. Feature-based fusion methods try to make the pointcloud data and the RGB data complement each other. One way is to fuse the information from the pointcloud into the RGB image. For example, [8] fuses the features from a Region of Interest (RoI) in both 2D image and 2D depth map, and then conducts 3D box regression. MV3D [9] projects the lidar pointcloud to two image representations (bird's-eye view and front view). The features and information extracted from these two image representations and the RGB image are then fed into a region-based fusion network for 3D object detection. AVOD [10] first generates a BEV map from a voxel grid representation of the lidar pointcloud. The features extracted from both the BEV map and the RGB image are fused for 3D object detection through a first-stage region proposal network and a second-stage detector network. The main problem of these methods is the loss of the 3D geometric information in the lidar pointcloud caused by using only the pointcloud's 2D representations, leading to some errors in locating small objects such as pedestrians. Feature-based fusion can also be achieved by fusing the information from the image space into the pointcloud. For example, [11] extracts the geometric features in 3D and color features in 2D from RGB-D images and then fuses them for 3D object detection. MVX-Net [17] fuses the RGB image and pointcloud point-wise or voxel-wise. The features extracted from the RGB image by a pre-trained 2D CNN are fused with the pointcloud in a voxel-based network to do 3D object detection. PointPainting [18] assigns the semantic feature to each lidar point by fusing the 2D detection result from the RGB image, thus achieving better results in pointcloud-based neural network detector. Since these approaches heavily rely on the pointcloud features, they still can not generate good results for faraway objects with sparse lidar points.

Frustum-based Fusion. Frustum-based fusion methods use the detection results from 2D image to generate frustums for the pointcloud, hence reducing the search space in 3D. An early and classic method is Frustum PointNets [7]. This method first generates a frustum for each object detected in 2D, then applies a PointNet-based approach to do instance segmentation and 3D box estimation in each frustum. Some work [19], [20] have improved the process of frustum generation by filtering out some background noise, and there is work focusing on changing the content of frustums. For example, Frustum ConvNet [12] generates a sequence of sub-frustums via sliding in the original 3D frustum. Frustum Voxnet [21] voxelizes parts of the frustum instead of using the whole frustum space, which offers more accurate representations around the area of interest. Some other researchers aimed to provide more fusion information for improving the accuracy of 3D detection. For example, one work [22] combines the pointcloud features in the frustum with the image features in the 2D bounding box as early-fusion and then applies a PointNet-based detector. Another work [23] fuses their own BEV detection results with 3D/BEV results from Frustum PointNets as late-fusion. These perform well for most objects. But unfortunately, for faraway objects with sparse lidar points, pure neural network based approaches cannot generalize well.

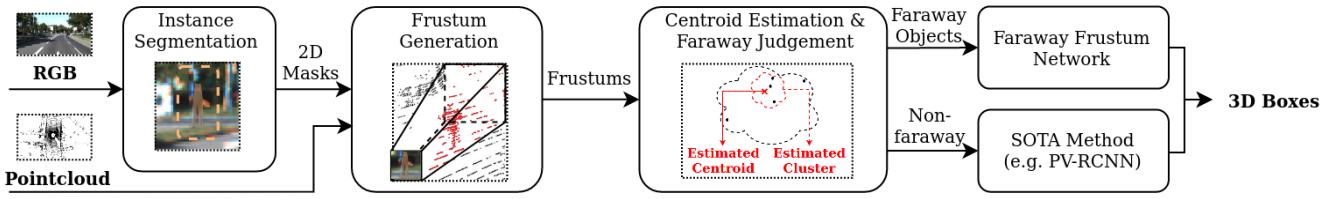

Fig. 2. Overview of the 3D/BEV object detection system based on our proposed method (*Faraway-Frustum*). It contains three main stages: frustum generation, centroid estimation, and box regression. First, the 2D object information (classification and 2D semantic mask) is extracted from the image by conducting instance segmentation, and then the 3D frustum is shaped by extruding the 2D semantic mask to the 3D coordinate system. Second, lidar pointcloud (red) points in the frustum are collected and clustered, and then the 3D object centroid is estimated. Finally, depending on the faraway/nearby decision, the 3D bounding box is predicted by our Faraway Frustum Network or a state-of-the-art method.

One recent work [24] achieved good 3D/BEV pedestrian detection results around 30 meters. In our work, we extend the range significantly, detecting pedestrians at 60 meters and beyond, where most SOTA approaches completely fail.

III. PROPOSED METHOD

An overview of our proposed method is shown in Fig. 2 and Algorithm 1. Our method takes both the RGB image and lidar pointcloud as input and outputs 3D/BEV bounding boxes $\mathbf{B}_i$ with class id c_i. There are three main stages in our method: frustum generation, centroid estimation, and depth-refinement with box regression. Each stage will be illustrated in detail in the following subsections.

A. Frustum Generation

2D instance segmentation. 2D instance segmentation serves as the basis of frustum generation. It takes an image as input and outputs the 2D object detection results containing 2D bounding boxes and semantic masks.

In this work, we use the 2D instance segmentation framework Mask R-CNN [25] to obtain 2D object information $\{R_i\}$ from image $\mathbf{I}$:

$$\{R_i\} = f_{\text{Mask R-CNN}}(\mathbf{I}) \tag{1}$$

where $f_{\text{Mask R-CNN}}$ represents the Mask R-CNN framework. $R_i = (c_i, \mathbf{b}_i, \mathbf{M}_i, s_i)$ is the instance segmentation result, which is a 4-tuple consisting of class label c_i, 2D bounding box $\mathbf{b}_i$, 2D semantic mask $\mathbf{M}_i$, and confidence score s_i for object i.

Frustum generation. We use the set of 2D results $\{R_i\}$ to generate frustums and to further identify the lidar points that correspond to each object i. With the known transformation $\mathbf{T}$ between the camera and the lidar, we use the semantic mask $\mathbf{M}_i$ for 2D-to-3D projection as shown in Fig. 4(b). Then the corresponding "frustrum pointcloud" $\mathbf{P}'_i$ can be identified from the raw lidar pointcloud $\mathbf{P}$ based on the frustum:

$$\mathbf{P}'_i = f_{\text{Mask-frustum}}(\mathbf{M}_i, \mathbf{T}, \mathbf{P}) \tag{2}$$

The mask-frustum based projection $f_{\text{Mask-frustum}}$ is our main approach. As shown in Fig. 4(c), we believe that using the semantic mask can exclude some noise points that do not belong to the target object, e.g., the points from occluded objects or the background. As an alternative, we also tested the box-frustum based projection $\mathbf{P}'_i = f_{\text{Box-frustum}}(\mathbf{b}_i, \mathbf{T}, \mathbf{P})$, which is used as a comparison with our main approach.

Algorithm 1: Faraway-Frustum($\mathbf{P}$, $\mathbf{I}$, $\mathbf{T}$, z_{th})

Input:
Lidar pointcloud $\mathbf{P} \in \mathbb{R}^{N,3}$.
RGB image $\mathbf{I} \in \mathbb{R}^{H,W,3}$.
Calibration matrix $\mathbf{T} \in \mathbb{R}^{4,4}$.
Faraway object threshold $z_{\text{th}} \in \mathbb{R}$
Output:
3D object bounding box $\mathbf{B}_i \in \mathbb{R}^7$.
Class id c_i.
Main algorithm:
$\{c_i, \mathbf{b}_i, \mathbf{M}_i, s_i\} = f_{\text{Mask R-CNN}}(\mathbf{I})$ $(i = 1, 2...n)$;
foreach $i(1, 2...n)$ **do**
  $\mathbf{P}'_i = f_{\text{Mask-frustum}}(\mathbf{M}_i, \mathbf{T}, \mathbf{P})$;
  $(x_i, y_i, z_i) = f_{\text{clustering}}(\mathbf{P}'_i)$;
  if $z_i \geq z_{th}$ **then**
    $\mathbf{P}''_i = f_{\text{projection}}(\mathbf{P}'_i, x_i, y_i, z_i)$;
    $(z'_i, w_i, l_i, h_i, \alpha_i) = f_{\text{FF-Net}}(\mathbf{P}''_i, c_i)$;
    $\mathbf{B}_i = (x_i, y_i, z'_i, \alpha_i, w_i, l_i, h_i)$;
  else
    $\mathbf{B}_i, c_i = f_{\text{SOTA}}(\mathbf{P}, \mathbf{I})$;
  end
end

B. Centroid Estimation

Centroid estimation. With $\mathbf{P}'_i$ obtained from the frustum, we then estimate the 3D centroid (x_i, y_i, z_i) for object i:

$$(x_i, y_i, z_i) = f_{\text{clustering}}(\mathbf{P}'_i) \tag{3}$$

The 3D object centroid plays two key roles in our method. One is to use the depth z_i to determine whether object i should be treated as a faraway object. The other is to further generate the 3D/BEV detection results for faraway objects.

Based on our observation in the KITTI dataset, regardless of whether the pointcloud in the frustum is dense or sparse, there are always some points on the object's surface. Thus, we adopt a fast clustering technique using histograms to estimate the 3D object centroid.

First, for all points in the pointcloud $\mathbf{P}'_i$, the histogram of all the coordinate values in each axis is generated (here we have 3 axes x, y, and z). For the histogram of each axis, we define the edges of every bin in the histogram as (e^l_j, e^r_j), $\forall j \in \{0, 1, \cdots, N\}$, and the count of values belonging

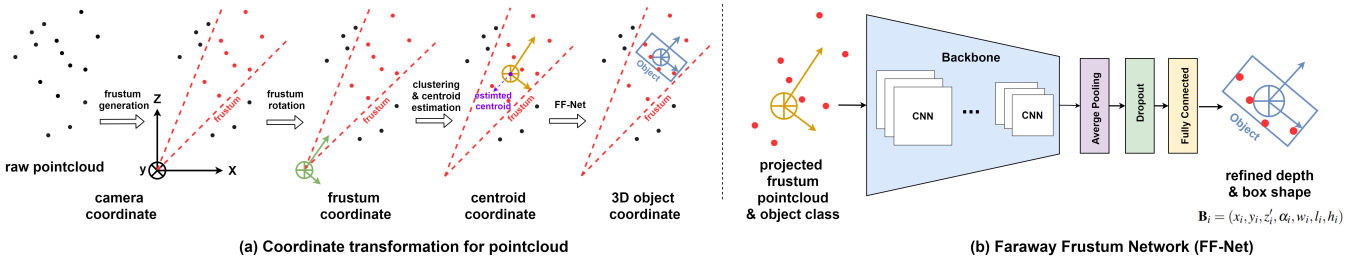

Fig. 3. An illustration of coordinate transformation for pointcloud and Faraway Frustum Network (FF-Net). (a) Illustrates the process of projecting the pointcloud into different coordinate systems in our method. After carrying out frustum generation, frustum rotation, clustering, and centroid estimation, the frustum pointcloud is projected into the centroid coordinate system. Our goal is to further localize the 3D object by the 2D projection of the frustum pointcloud and the FF-Net. (b) The FF-Net is essential to refine the object center, regress the box size, and resolve certain issues that may occur while creating the frustum. For example, due to errors in detection or segmentation, the cluster centroid may not be aligned well with the object. The FF-Net is trained to deal with such issues and refine the object localization.

to each bin as n_j, $\forall j \in \{0,1,\cdots,N\}$, where N is the number of bins. Then, we identify the bin with the largest count value. This indicates that most of the points are concentrated within this bin. The corresponding index will be obtained by $j^* = \arg\max_j (n_j)$. Finally, the centroid value of an axis, for example, the centroid of x-axis, x_i, can be obtained by $x_i = \frac{1}{2}(e^l_{j^*} + e^r_{j^*})$. The centroid values y_i and z_i for the other two axes is estimated in the same way.

Faraway threshold. Using the estimated centroid (x_i, y_i, z_i), we determine whether each object i is considered faraway or nearby. We select different faraway thresholds z_{th} for different object classes based on the statistics of the number of ground truth lidar points in each object. As shown in Fig. 5, we first draw a line for the objects that have 10 lidar points, then we approximately select the z_{th} such that most objects of distance larger than z_{th} have less than 10 points. To determine whether object i should be treated as a faraway object, we compare the estimated distance z_i with z_{th} of the corresponding object class c_i. If $z_i > z_{\text{th}}$, then it is a faraway object, otherwise it is not.

C. Box Regression

To obtain the 3D/BEV bounding box $\mathbf{B}_i$ for object i based on the estimated object centroid (x_i, y_i, z_i), we need to estimate the box shape: the length l_i, width w_i, height h_i, and orientation α_i. If object i is a faraway object, directly using learned representations from state-of-the-art models is not a good choice because they do not generalize well from dense pointclouds to very sparse pointclouds. Furthermore, the estimated object centroid (especially the depth) may still be quite far from the box center. As such, we propose to use a light model named Faraway Frustum Network (FF-Net) for faraway objects to refine the depth z'_i and regress the shape $(w_i, l_i, h_i, \alpha_i)$ of the 3D bounding box with the input of the object class c_i and a 2D projection $\mathbf{P}''_i$ of the frustum pointcloud, as shown in Fig. 3.

Pointcloud Projection. 2D projection $\mathbf{P}''_i$ of the frustum pointcloud is generated using a coordinate transformation as shown in Fig. 3(a). After conducting frustum generation for the raw pointcloud $\mathbf{P}$, the frustum pointcloud $\mathbf{P}'_i$ is obtained. First, the camera coordinate system is rotated to the center view of the frustum to build the frustum coordinate system. Second, after histogram-based clustering and centroid estimation, the frustum coordinate system is transformed to the centroid coordinate system with the estimated centroid at the origin. Finally, all lidar points inside of the frustum are projected into the centroid coordinate system in bird's-eye view. The projected frustum pointcloud in the centroid coordinate system is taken as the 2D projection $\mathbf{P}''_i$ of the frustum pointcloud.

Box Regression. FF-Net takes the object class c_i and a 2D projection $\mathbf{P}''_i = f_{\text{projection}}(\mathbf{P}'_i, x_i, y_i, z_i)$ of the frustum pointcloud $\mathbf{P}'_i$ whose origin is the estimated centroid (x_i, y_i, z_i) as input and combines a MobileNet-based [26] backbone network with a multi-output regression head as shown in Fig. 3(b).

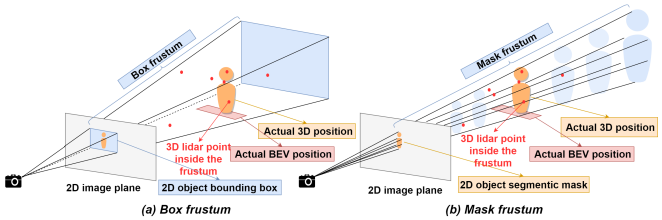

Fig. 4. An illustration of frustum generation. The main difference between box frustum and mask frustum is that box frustum uses the 2D bounding box as the projection source, while mask frustum uses the 2D semantic mask. Mask frustum gives a more compact search space alongside the outline of the object, and thus excludes some noise points caused by potential occlusions.

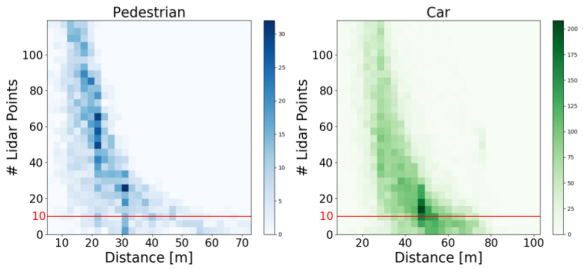

Fig. 5. The number of points belonging to an object (pedestrians and cars) versus distance from the sensor in the KITTI dataset. As the distance (x-axis) increases, the number of lidar points in an object (y-axis) decreases drastically and the pointcloud of each faraway object is very sparse. When the number of lidar points is less than 10, the shape of objects cannot be recognized. Thus, objects with fewer than 10 points are considered faraway objects. We use this distribution to decide the faraway decision threshold.

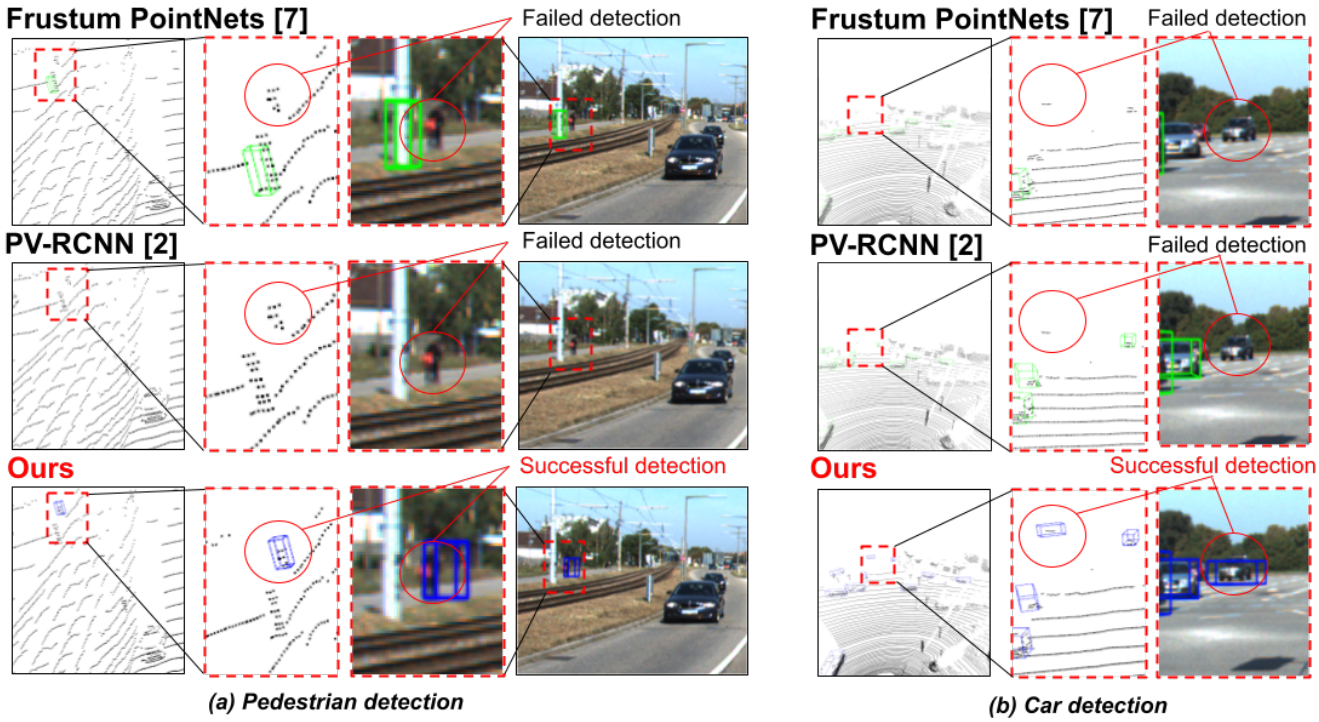

Fig. 6. Example 3D detection results of faraway objects from the KITTI test set. (a) Pedestrian detection. *Top row*: Frustum PointNets [7], which is based on fusing multiple modalities (RGB and pointcloud). *Middle row*: PV-RCNN [2], which uses only the pointcloud. *Bottom row*: Our proposed method. (b) Car detection. Same arrangement as in (a). In these examples, for both the faraway pedestrian and the faraway car, our proposed method successfully detects the targets. However, state-of-the-art methods (Frustum PointNets [7] and PV-RCNN [2]) all fail.

The estimated centroid (x_i, y_i, z_i) is considered as the origin of the input projection, and the goal of the FF-net is to shift this origin to the real center of the 3D bounding box, i.e. to transform a centroid coordinate to a 3D object coordinate as shown in Fig. 3(a). Furthermore, another goal is to regress the box shape, achieved by minimizing the loss of the regressed length, width and height of the box. We use mean absolute error (MAE) to compute the loss L_x,L_y,L_z,L_w,L_l,L_h,L_α of box centroid (x_i', y_i', z_i') and shape $(w_i, l_i, h_i, \alpha_i)$ respectively. By summing these losses, FF-Net is trained and optimized with multi-task losses L_{FF-Net}.

Finally, for a faraway object, we take the shifted depth z_i' and the regressed 3D bounding box shape $(w_i, l_i, h_i, \alpha_i)$ from the output of FF-Net and combine them with (x_i, y_i) from the estimated centroid. We assign a 3D bounding box to the faraway object i as:

$$\mathbf{B}_i = (x_i, y_i, z_i', \alpha_i, w_i, l_i, h_i). \tag{4}$$

It should be noted that the class id c_i is directly obtained with Mask R-CNN. If object i is not a faraway object, we switch to using learned representations following SOTA (e.g. Frustum-PointNets [7], PV-RCNN [2]) methods. In this case, the 3D bounding box and class id for a non-faraway object are obtained by $\mathbf{B}_i, c_i = f_{\text{SOTA}}(\mathbf{P}, \mathbf{I})$.

IV. EXPERIMENTS

We utilized the KITTI dataset [1] to conduct our experiments. We specifically extracted faraway objects in KITTI and investigated them separately. Details of dataset preparation, evaluation metrics, and implementation are described below.

Dataset preparation. First we analyzed the statistics of the original KITTI dataset by evaluating the distribution of the objects at different distances and having different numbers of lidar points, as shown in Fig. 5. It is obvious that as the distance increases, the number of lidar points in an object decreases. That is to say, the pointcloud is very sparse for faraway objects. We selected the faraway threshold z_{th} for cars as 75 meters and for pedestrians as 60 meters. We also split the KITTI dataset into the training set (3724 frames) and the validation set (3757 frames).

Evaluation metric. The major evaluation metric is the mean average precision (mAP) with a given IoU threshold, as suggested by the KITTI dataset. We use both the official benchmark and our specific benchmark for faraway objects. In the benchmark for faraway objects, we only evaluate faraway objects and we use a specific IoU threshold (0.1) for the mAP. We use a low IoU threshold because detecting an object with a small overlap with the ground truth is still better, at long distances, than no detection at all. The mAP results are computed with 11 recall positions which is the same as in [1].

We also evaluate the faraway objects using the average

IoU (*aIoU*), which is defined as $aIoU = \frac{\sum_{i=1}^{n} \text{IoU}}{n}$ where n is the total number of faraway objects and $\sum_{i=1}^{n} \text{IoU}$ is the sum of the IoU values calculated based on the ground truth and the predicted bounding box.

Implementation. Our method uses the instance-level semantic segmentation method (Mask R-CNN [25] pre-trained on the COCO dataset [27]) to generate 2D object information from the image space. Our first approach (ours1) uses 2D semantic masks to generate frustums in 3D. The second approach (ours2) uses 2D bounding boxes to generate frustums. As a baseline (ours3), we also use ground truth 2D bounding boxes provided by KITTI to generate frustums. All of our approaches are combined with PV-RCNN [2] for non-faraway objects. The Faraway Frustum Network (FF-Net) is trained using the Adam optimizer with early stopping. FF-Net is trained with the whole training set, but during inference only faraway objects are fed to the FF-Net.

We compared our proposed method with the following SOTA 3D/BEV object detectors: SECOND [3], PointPillars [4], PV-RCNN [2], and Frustum PointNets [7]. These SOTA methods are all trained from scratch using our data split, and we evaluated them with the same faraway metrics.

V. RESULTS

Quantitative results. Table I shows the average IoU results for BEV detection of faraway objects in the KITTI validation dataset. All of our methods outperform SOTA methods with higher average IoU of at least 0.051 and at most 0.157. And surprisingly, none of the methods except ours can achieve an average of 0.1 IoU for both pedestrians and cars. This result not only demonstrates the effectiveness of our method, but also underlines an important shortcoming of SOTA methods. Furthermore, we believe finding the exact shape of faraway objects is not a priority. As long as we obtain the 3D/BEV detection result with even a small IoU (e.g. 0.1), it can still be very useful for certain applications such as automated driving. In other words, a 0.1 IoU detection is better than a false negative. As such, we set the IoU threshold to 0.1 for faraway objects in the mAP comparison.

TABLE I
AVERAGE IOU COMPARISON OF FARAWAY BEV OBJECT DETECTION ON KITTI VAL DATASET

Method	BEV Ped.	BEV Car
	> 60 m	> 75 m
Frustum PointNets [7]	0.000	0.000
SECOND [3]	0.036	0.009
PointPillars [4]	0.072	0.000
PV-RCNN [2]	0.051	0.018
Ours1 (FF-net mask)	0.123	**0.157**
Ours2 (FF-net box)	**0.124**	0.150

* Name explanation: Ped. (Pedestrian).
** The **bold** result indicate the best in all methods, and the blue result represents the second place.

The mAP results of faraway 3D/BEV detection over KITTI validation dataset for pedestrians and cars are shown in Table II. For faraway pedestrians (over 60 meters), our methods (ours1 and ours2) outperform SOTA methods on 3D/BEV detection with large mAP margins (BEV: at least 22.14% and at most 45.45%, 3D: at least 9.55% and at most 44.54%). For faraway cars (over 75 meters), our methods (ours1 and ours2) outperform SOTA methods again with a higher mAP (BEV: at least 27.09% and at most 46.90%, 3D: at least 16.52% and at most 46.90%).

TABLE II
MAP COMPARISON OF FARAWAY 3D/BEV OBJECT DETECTION ON KITTI VAL DATASET

Method	3D Ped.	BEV Ped.	3D Car	BEV Car
	IoU threshold 0.1			
	Over 60 meters		Over 75 meters	
FP [7]	00.00	00.00	00.00	00.00
SE [3]	13.63	13.63	09.09	09.09
PP [4]	22.40	22.40	00.00	00.00
PV [2]	19.69	19.69	18.18	18.18
Ours1	**44.54**	44.54	34.70	45.27
Ours2	31.95	**45.45**	**46.90**	**46.90**

* Name explanation: FP (Frustum PointNets), SE (SECOND), PP (PointPillars), PV (PV-RCNN), Ours1 (FF-net mask), Ours2 (FF-net box), Ped. (Pedestrian).
** The **bold** result indicates the best in all methods, and the blue result represents the second place. We set the experimental IoU threshold as 0.1 for faraway pedestrians because in the current stage, it is extremely difficult to precisely locate faraway objects, while detecting faraway objects even with low IoU is still practical and useful.

The mAP results of 3D/BEV detection over the KITTI validation dataset for pedestrians and cars are shown in Table III. For the non-faraway official Easy/Mod/Hard benchmark, our method performs as well as the baseline SOTA method (PV-RCNN).

All the above results demonstrate that our method achieves better performance on faraway object detection without impairing the overall performance of SOTA methods.

Qualitative results. Fig. 6 shows a visual example of the results of different methods for faraway object detection. We compared our method with PV-RCNN [2] and Frustum PointNets [7]. In frame (a), Frustum PointNets mistakenly detects the pole as a pedestrian, while PV-RCNN provides no detection. However, for detecting the faraway pedestrian near the pole, only our detector succeeds. In frame (b), state-of-the-art methods all fail in detecting the faraway car. In contrast, our method successfully detects the car in 3D.

VI. CONCLUSION

In this paper, we proposed an alternative 3D/BEV detector, named *Faraway-Frustum*, to deal with lidar sparsity of faraway objects. Our method takes advantage of relatively dense image data to find faraway objects and circumvents the disadvantages of pointcloud-driven neural networks working on very sparse points. Moreover, our alternative detector can be flexibly combined with a state-of-the-art method to form an overall 3D/BEV object detection system via setting faraway thresholds.

TABLE III
MAP COMPARISON OF 3D/BEV OBJECT DETECTION ON KITTI VAL DATASET

Method	3D/BEV Pedestrian			3D/BEV Car		
	IoU threshold 0.5			IoU threshold 0.7		
	Easy	Mod	Hard	Easy	Mod	Hard
PV-RCNN [2]	69.53/73.32	66.02/67.42	62.91/65.70	96.73/97.53	93.18/94.77	85.76/94.78
Ours1 (FF-net mask + PV-RCNN)	71.65/73.02	67.29/68.73	62.08/66.87	96.73/97.54	93.17/94.75	85.76/94.77
Ours2 (FF-net box + PV-RCNN)	71.74/73.11	67.29/68.73	62.08/66.88	96.73/97.54	93.17/94.75	85.76/94.77

For the non-faraway official Easy/Mod/Hard benchmark, our method performs as well as the baseline SOTA method (PV-RCNN). This result shows that the proposed method can be used to improve the faraway detection performance without sacrificing the non-faraway detection performance.

The experiments demonstrated the feasibility of our approach, but they also exposed a significant shortcoming of state-of-the-art object detection methods: Relying on learned representations of very sparse lidar points to detect faraway objects is not a good strategy.

ACKNOWLEDGMENT

Material reported here was supported by the United States Department of Transportation under Award Number 69A3551747111 for the Mobility21 University Transportation Center. Amy findings, conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[2] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. 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*, 2020, pp. 10 529–10 538.

[3] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[4] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[5] Z. Liu, X. Zhao, T. Huang, R. Hu, Y. Zhou, and X. Bai, "Tanet: Robust 3d object detection from point clouds with triple attention." in *AAAI*, 2020, pp. 11 677–11 684.

[6] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 040–11 048.

[7] 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/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 918–927.

[8] Z. Deng and L. Jan Latecki, "Amodal detection of 3d objects: Inferring 3d bounding boxes from 2d ones in rgb-depth images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5762–5770.

[9] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[10] 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)*, 2018, pp. 1–8.

[11] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 808–816.

[12] Z. Wang and K. Jia, "Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1742–1749.

[13] M. Ye, S. Xu, and T. Cao, "Hvnet: Hybrid voxel network for lidar based 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1631–1640.

[14] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 770–779.

[15] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "Std: Sparse-to-dense 3d object detector for point cloud," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1951–1960.

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[17] V. A. Sindagi, Y. Zhou, and O. Tuzel, "Mvx-net: Multimodal voxelnet for 3d object detection," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7276–7282.

[18] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4604–4612.

[19] X. Du, M. H. Ang, S. Karaman, and D. Rus, "A general pipeline for 3d detection of vehicles," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3194–3200.

[20] K. Shin, Y. P. Kwon, and M. Tomizuka, "Roarnet: A robust 3d object detection based on region approximation refinement," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2510–2515.

[21] X. Shen and I. Stamos, "Frustum voxnet for 3d object detection from rgb-d or depth images," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 1698–1706.

[22] X. Zhao, Z. Liu, R. Hu, and K. Huang, "3d object detection using scale invariant and feature reweighting networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 9267–9274.

[23] P. Cao, H. Chen, Y. Zhang, and G. Wang, "Multi-view frustum pointnet for object detection in autonomous driving," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 3896–3899.

[24] M. Fürst, O. Wasenmüller, and D. Stricker, "Lrpd: Long range 3d pedestrian detection leveraging specific strengths of lidar and rgb," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.

[25] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[26] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.