IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Use Your Imagination: A Detector-Independent Approach For LiDAR Quality Booster

Zeping Zhang[1], Tianran Liu[2] and Robert Laganière[3]

***Abstract*—Features from LiDAR and cameras are considered to be complementary. However, due to the sparsity of the LiDAR point clouds, a dense and accurate RGB/3D projective relationship is difficult to establish especially for distant scene points. Recent works try to solve this problem by designing a network that learns missing points or dense point density distributions to compensate for the sparsity of the LiDAR point cloud. In this work, we propose to use an imagine-and-locate process, called UYI. The objective of this module is to improve the point cloud quality and is independent of the detection network used for inference. We accomplish this task through a GAN based cross-modality module which uses an image as input to infer a dense LiDAR shape. Boosted by our UYI block, our experiments show a significant performance improvement in all tested baseline models. In fact, benefiting from the plug-and-play characteristics of our module, we were able to push the performance of existing state-of-the-art model to a new height. *Code will be made available*.**

I. Introduction

In recent years, many 3D detection solutions based on RGB cameras + LiDAR sensors [1], [2], [3], [4] have been proposed and achieved good performance in 3D detection. Modern 64-beam LiDAR sensors are able to guarantee high precision object positioning within a range of nearly 100 meters. However, compare with dense pixels information from RGB images, the distribution of cloud points generated becomes sparser as distance increases. Many previous works [5], [6], [7] have considered how to use the high-resolution characteristics of RGB images to compensate for the drawbacks of LiDAR sensors. For example, early work like MV3D [1] and its successor AVOD [2] proposed methods that use RGB features as ancillary information for points. Later works [8], [9] have focused on enhancing the point clouds, ContFuse [10] proposes a new way to aggregate information between 2 modalities to enrich the original point cloud. BtcDet [11] uses operations such as shape prototypes and symmetric flips to densify the point cloud of a vehicle object. However, considering the extreme sparsity of some objects, point recovery can be very difficult in some cases. Many works [12], [13] also attempt to make the network learn the missing points or the distribution of points to compensate for the sparsity of point cloud, but are confronted with the same issues. The purpose of UYI is to provide better quality foreground points for 3D detection task. For the previous state of the art method SFD_Net, the main limiting factor resides in the LIDAR branch of a multi-modal detection network. The 3D proposals are based entirely on the raw point cloud which still suffers from sparsity. This paper proposes a module to enhance a LiDAR point cloud by densify it to facilitate the detection of the road objects. To the best of our knowledge, this work is the first to accomplish this task with a GAN [14]-based cross-modal module that uses images as input to describe the shape of objects in LiDAR format. With this work, we aim at answering the following three questions:

1. How the sparsity of distant LiDAR point cloud affects LiDAR Pseudo-point cloud generation.
2. How to efficiently generate 3D object shape in lidar format using RGB information
3. How can we design this module to be plug-and-play on the widest possible range of models?

A. Sparse Representation to Pseudo Point Cloud

3D detectors inevitably suffer from point cloud sparsity, especially the methods [15], [16], [17] that use LiDAR-only information. Therefore, Chan et al. [18] proposed to use a dense pseudo point cloud generated by depth completion to enhance the LiDAR point cloud. SFD[19] extracts 3D geometry in a pseudo point cloud by designing a new RoI feature fusion method named 3D-GAF (3D Grid-wise Attentive Fusion). However, the pseudo lidar data generated by depth completion modules generally suffers from the long tail problem. For these difficult scenarios where there exist very few lidar points, so most, if not all, features the detector can acquire comes from pseudo points. These drawbacks limit the performance of SFDNet. In this paper we propose an effective object-level LiDAR pseudo-point cloud generation process that increases the density and quality of a captured point cloud, thereby not only improving the performance of 3D detection, but also reducing the number of pseudo-points to be generated compared to global-level completion.

B. 2D Point Cloud Colormap and Projection

To solve this problem, we transform the task of generating point clouds from RGB vehicle images into an image translation task. A GAN augmented with a self-attention module is introduced to convert the input RGB image into a 2D color point cloud image. We convert an input 3D vehicle point cloud box with depth into an RGB image by establishing a depth-color gradient correspondence as described in section III.A.

Manuscript received: February, 16, 2023; Revised April, 3, 2023; Accepted May, 22, 2023.

This paper was recommended for publication by Editor Pascal Vasseur at the Editor is the Senior Editor who communicated the decision.

[1]Zeping Zhang, [2]Tianran Liu and [3]Robert Laganière are with the VIVA Research Lab, University of Ottawa, 800 King Edward, Ottawa, ON K1N6N5, Canada (email: zeping.zhang@uottawa.ca, tliu157@uottawa.ca, laganier@uottawa.ca)

Corresponding author: Robert Laganiere

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

In previous works [20], [21] it has been shown that GANs are indeed capable of learning from color information. GANs are widely used in applications such as color paintings and other color reconstruction tasks [22], [23]. Here we want the GAN to learn the depth information of point clouds which will provide us with an approach to process and generate a 3D point clouds as an image.

C. Plug-and-Play Design

In this paper we opted for a design that could be used within as many models as possible. In BtcDet [11], the authors demonstrated that when the point clouds of vehicles are accurately densified, the average precision has the potential of reaching 99%. This demonstrates that the potential for improvement for current detection models actually lies fundamentally in the quality of the point cloud. The modules proposed in this paper have been designed to work in between the input data and the detection model and provide a higher-quality point cloud to the input side of the detection block to enhance detection.

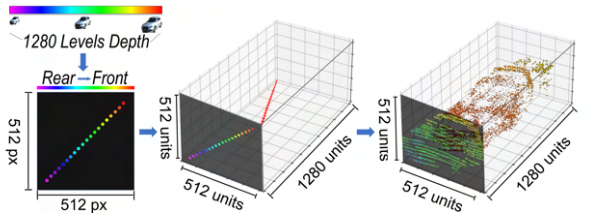

Fig. 1. Depth-Color Projection: The depth-color map correspondence is shown on the upper left. The example image on the left is mapped to a diagonally distributed depth point cloud (shown in the middle). On the right is an example of 3D shape after depth restoration of the model.

II. Related Work

A. LiDAR and RGB Fusion 3D Detectors

Due to the sparsity of point clouds, researchers typically consider multimodal approaches that utilize both images and point clouds. Some of the recent approaches started to use depth estimation in deep learning to generate pseudo-LiDAR point clouds. MMF [24] and ContFuse [10] both use anchorless methods, but MMF [24] also uses depth estimation for dense pseudo-LiDAR point clouds and achieves better performance. SFD-Net [19] uses the point cloud to depth-complement the pixel points on the image, making the point cloud denser. However, these pseudo-point cloud methods are still essentially point cloud completion, and establishing accurate RGB-3D dense projection relationships remains a challenge due to the sparsity of distant point clouds.

B. Previous Occlusion Processing Methods

Occlusion poses an immediate performance problem for many computer vision tasks, such as detection [25], instance segmentation [26] and 3D detection for autonomous driving [11]. In the case of previous 3D detection, many approaches have attempted to compensate for the occluded portion of the point cloud. For instance, Hu et al. [27] proposes a raycasting algorithm for efficiently computing object visibility and augmentation. Later BtcDet [11] was the first method to propose learning the occluded shapes in point cloud data by computing shape occupancy. However, both of them did not use RGB information for reconstruction. BtcDet complements the target using methods such as mirror symmetry and best-matching, which we designate as TFMM (Traditional Flip-Mirroring Method). We will compare our method with TFMM in the experiment section.

C. Depth completion GAN

An important upstream task for autonomous driving is the acquisition of dense depth information. Given its semi-supervised nature, GAN-based schemes treats depth as a fully generative task rather than a regression. In indoor scenes, condition-guided GAN [28], using sparse depth map, RGB image, or semantic layout as input, [29], [30], [31] can generate densified depth with SOTA performance. While in outdoor scenes, earlier researchers have shown that a well-designed GAN can perform full image-based depth estimation [32]. Treating the point cloud as a priori information of depth, [33], [34] upsample the initial point cloud to densify the point cloud for downstream tasks. In this paper, we extend this idea by using a texture-aware GAN with distance color coding to complete this cross-modal transition,

III. Use Your Imagination Block

A. From Depth Map to Color Space

To make object prediction from point clouds more effective, we propose a method to infer depth information from 2D images using color space mapping. We, therefore, propose to transform the depth information of a 3D vehicle point cloud box into color gradients frontal view projection. We do this by training a GAN model on the color mapping of the depth features. To this end, we created a 1280-level color/depth lookup table in which the color closer to a purple will be considered to be farther from the camera while the color becomes more reddish when closer. This color depth mapping is used to construct the training set of our GAN. To be more specific, let $\{v_1, v_2, ..., v_n\} \in V_i$ denote the points in i-th LiDAR point cloud frame from the dataset, and let p_i be the projection of 3D point v_i on the camera image.

In general, for every v_i in V, its corresponding pixel p_i in the camera view can be found as [35] (1),(2):

$$\mathbf{p_i} = f(v_i) = \mathbf{P^0}_{\text{rect}}\, \mathbf{R^0}_{\text{rect}}\, \mathbf{T}^{cam}_{velo} \mathbf{v_i} \tag{1}$$

$$\mathbf{T}^{cam}_{\text{velo}} = \begin{pmatrix} \mathbf{R}^{\text{cam}}_{\text{velo}} & \mathbf{t}^{\text{cam}}_{\text{velo}} \\ 0 & 1 \end{pmatrix} \tag{2}$$

where $\mathbf{P_0}_{\text{rect}}$ is the projection matrix in camera coordinates, $\mathbf{R_0}_{\text{rect}}$ and $\mathbf{R}^{\text{cam}}_{\text{velo}}$ refer to the rectified rotation matrix and the LiDAR to camera rotation matrix, respectively. And $\mathbf{t}^{\text{cam}}_{\text{velo}}$ is the translator vector from LiDAR to camera.

Let us now consider a given object bounding box projection on the camera reference view. The point with the minimum z-value is defined as the point of depth x_0, then the x values of the other points can be re-corrected as $x_i - x_0$. Considering

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

now the vehicle boxes in the KITTI dataset, we noted that, according to our statistics, the 95th percentile of vehicle box length in KITTI [35], [36] training set is about 4.52 meters. Assuming the length of most vehicles to be within 4.52 meters, we use [0, 4.52] as the standard depth range for generated vehicle point clouds. Therefore, for each point p_i in the bounding box, its corresponding corrected depth information d_i can be estimated as (3):

$$d_i = Depth(p_i) = (x_0 - x_i)/4.52 * 1279, (x_0 - x_i) \geq 0 \quad (3)$$

where x_i comes from 3D point v_i corresponding to p_i.

Now, to convert the corrected depth value into a color map, the RGB value for each pixel p_i in the bounding box of an object is converted according to the following transformation (4):

$$\begin{cases} (255 - d_i, 255, 255) & ,0 \leq d_i < 256 \\ (0, d_i - 255, 255) & ,256 \leq d_i < 512 \\ (0, 255, 255 - d_i + 512) & ,512 \leq d_i < 768 \\ (d_i - 768, 255, 0) & ,768 \leq d_i < 1024 \\ (255, 255 - d_i + 1024, 255) & ,1024 \leq d_i < 1280 \\ (0, 0, 0) & ,otherwise. \end{cases} \quad (4)$$

Through the above transformation, for each given bounding box we can calculate its RGB depth map. The process is shown in Figure 1. Note that this depth-color projection process can simply be reversed for the process of projecting the RGB depth map back to the 3D vehicle box. During training, we manually select vehicle boxes (including relatively complete vehicle shapes) as ground truth to make GAN learn the semantic information of translation from 2D images to RGB depth map. The relevant part of the experiment will be described in Section IV.

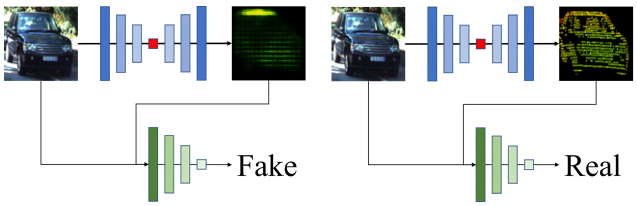

Fig. 2. Adversarial Learning Process for Conditional GAN: On the left is shown an instance where the discriminator successfully identified the image from the generator as fake. The right side shows an example where the generator successfully cheated the discriminator. The self-attention block is represented by the middle red block.

B. Sentient Generative Adversarial Network with Attention

1) Objective: In general, the generator of a conditional GAN in an image translation task learns a mapping from input image $real_a$ and a random noise vector z to output image $fake_b$, which would be compared with ground truth $real_b$ by the discriminator. Generator G tries to produce high-quality images that discriminators can not distinguish. The generator we used in our GAN provides a downsampler from 64 dimensions to 512 dimensions and the corresponding down-sampler with skip connections between the same dimensions. More specifically, our GAN accepts inputs of a fixed size of (256, 256, 3), and will be downsampled to (1,1,512) at the bottleneck. Then return to (256,256,3) after upsampling and transpose operations. In the middle of the U-Net structure, we use a self-attention block to better aggregate features. This procedure is described in Figure 2. Considering that the LiDAR information embedded in the RGB LiDAR map generated in the previous step behaves like texture features, we add a sentient loss $L_{sentient}$in our GAN. Based on a pre-trained VGG19 [37] model as a feature extractor, we map it to a pair of generated samples and real samples in the generator. This process is described in Algorithm 1, where N denotes the number of layers in the extractor, $E(real_b)$ and $feature_{real_b}$ denote features from $real_b$ generated by the extractor which is same for $fake_b$. $L_{sentient}$ denotes sentient loss, $feature_{real_b/fake_b}['relu_i']$ is the feature output of the i-th layer of the encoder, extracted from $real_b$ or $fake_b$.

Algorithm 1 Calculation of sentient loss in Generator

Require: $N \geq 0$, $real_a \neq \emptyset$, $real_b \neq \emptyset$
  Initialization:
  $L_{sentient} \leftarrow 0, E \leftarrow VGG19$
  $W \leftarrow [1.0/32, 1.0/16, 1.0/8, 1.0/4, 1.0]$
  $feature_{real_b} \leftarrow \mathrm{E}(real_b)$
  $feature_{fake_b} \leftarrow \mathrm{E}(fake_b)$
  while $i \leq N$ **do**
    $L_{sentient} \leftarrow L_{sentient} + W[j] * |feature_{real_b}['relu_i'] - feature_{fake_b}['relu_i']|$
  end while
  return $L_{sentient}$

The loss function of our generator can be expressed as

$$\begin{aligned} L_{Generator} =& L_{L1}(real_b, fake_b) \\ &+ L_{GAN}(D(fake_b), True) \\ &+ L_{Sentient}(real_b, fake_b) \end{aligned} \quad (5)$$

,where D represent the discriminator, L_{GAN} is the MSE loss from discriminator and L_{L1} denotes the L1 loss.

To test the importance of adding $L_{sentient}$ and self-attention block to the generator, we also define the situation when $L_{Generator}$ does not contain $L_{sentient}$ as

$$\begin{aligned} L_{Generator} =& L_{L1}(real_b, fake_b) \\ &+ L_{GAN}(D(fake_b), True) \end{aligned} \quad (6)$$

2) Model Design: As shown in Figure 3, our UYI Block acts as a point cloud augmentation module before the detection phase. It accepts RGB and LiDAR input, integrates a YOLOX-l[38] detector into the RGB image, converts the content of the obtained detection area into a vehicle RGB depth map and then forwards it to the GAN to generate a complete vehicle RGB depth map. The RGB depth map will then be processed using color mapping as explained in section III.A to generate vehicle point cloud box and project it back into the original point cloud. The weight of the YOLO-X[38] detector is obtained through transfer learning from ImageNet to KITTI. The reason we choose YOLOX-l as the detector is that during our training

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

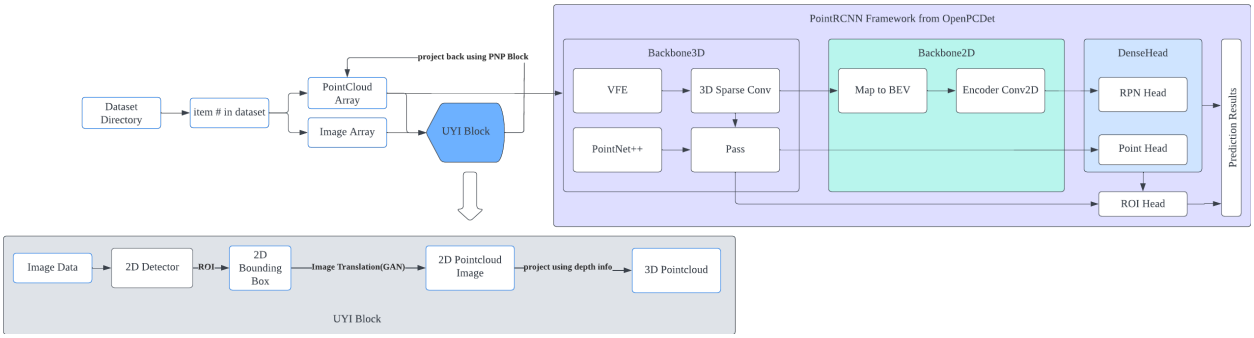

Fig. 3. The Workflow of UYI Module: The UYI module takes the RGB image and points cloud objects from the Data Loader, then projects the generated targets back to the original point clouds to produce high-quality point clouds which serve as input of the subsequent detection network.

on 2D objects on KITTI, YOLOX-l has achieved an AP of 87.66% which is very close to YOLOX-x's AP of 88.21%, but the size of YOLOX-l is only 54.69% of YOLOX-x. Therefore, YOLOX-l offers the best trade-off between performance and operating efficiency. These procedures enable us to obtain a high-quality point cloud. The enhanced high-quality point cloud can effectively improve the detection performance of the model, which we will be presented in the next section of experiments.

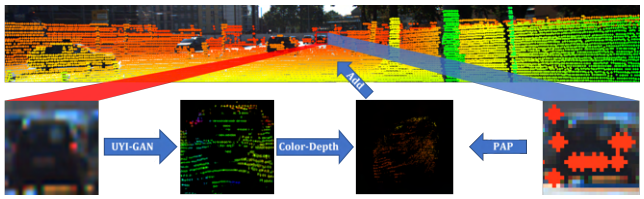

Fig. 4. Projection Awareness Positioning (PAP) Module: The leftmost of the second row is the ROI area of a vehicle output by the 2D detector, we generate its RGB point cloud map through UYI-GAN, and project it back to 3D space through color depth conversion to generate a 3D vehicle box. Finally, the target projection position of the 3D vehicle box in the original point cloud can be calculated according to the position of the points projected to the ROI area on the RGB image.

C. Projection Awareness Positioning Module

In order to project the generated vehicle box back accurately into the original point cloud, we need the (x, y, z) coordinates of the target projection location. In practice, the height coordinate y can be obtained as a direct reflection from RGB to the point cloud to obtain a frustum range. The mean-variance of the road height within a single sample is only 0.037m for the KITTI dataset, which enables us to simply interpret the lowest height value of the current point cloud as the reference value. The axis which needs more attention is the depth information, which is the value of the x-coordinate in the KITTI dataset. A typical approach is to use the principle of similar triangles, we have:

$$\begin{aligned} Estimated\ Distance = height_{real} \\ * focal_v / height_{pixel} \\ + (length/2) \end{aligned} \tag{7}$$

However, after calculating the positions of all the vehicle boxes in the KITTI training set using a similar triangle method we found that this approach produces an average error of 11.6 meters, which is clearly not accurate enough for positioning. Therefore we came up with the idea of using the point cloud corresponding to the original point in the 2D detection box as the position information. First, for each 2D detector that returns an ROI region, we transform it into a complete vehicle RGB depth map. In this case, if the number of real LiDAR points inside the 4.52 meters bounding box is sufficient, we calculate the geometric center of the resulting vehicle box. This process is shown in Figure 4. As it can be calculated that the average width of the vehicle boxes in the KITTI training set is 1.63 meters and is 1.80 meters at 95th percentile. Also, considering the 95th percentile of vehicle box length in the training set is about 4.52 meters (as mentioned in Section 3), thus the extent of the 3D area can be established as $O(x, y, z)$ using (8):

$$\begin{aligned} \{vi_x | O_x - 1.63/2 \leq vi_x \leq O_x + 1.63/2, vi \in V\}, while \\ \{vi_y | O_y - 4.52/2 \leq vi_y \leq O_y + 4.52/2, vi \in V\}\ and \\ \{vi_z | vi_z \in R, vi \in V\}. \end{aligned} \tag{8}$$

According to our statistics, 86% of the vehicle boxes can be localized using this method, with a localization accuracy ($IoU \geq 70\%$) of 94%. We found that 3 points is the minimum required to achieve this average accuracy or it would be too sparse for positioning an object properly. For the situation where the original point cloud projection on the target object area is less than 3 points, we trained a multinomial machine learning model based on multinomial linear regression [39] to predict the target depth and use the objects in KITTI's training set as ground truth. For each training sample, the inputs were defined as 2D bounding box width, 2D bounding box height, P2[1][1], 2D bounding box center (x, y), with the true depth as the training target and normalization performed on all the variables to (0,1]. Table I shows a significant improvement for depth estimation after this process, especially for the distant vehicle box, where we succeed in reducing the mean error to 1.56m. It could be observed that our MLR with polynomial features achieves the best results in both categories. We achieved an average error of only 3.06 meters for objects over 50 meters away and 1.56 meters for all objects. In this case, we align the obtained depth value as the geometric center point in the vehicle point cloud box generated by our GAN, thus making it possible to estimate the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Methods	Metrics	Easy	Medium	Hard
PV-RCNN Baseline	BEV_AP	92.86	88.93	**88.74**
	3D_AP	91.99	82.86	80.40
UYI(TFMM)+PV-RCNN	BEV_AP	93.93	89.43	88.53
	3D_AP	92.04	82.91	80.35
UYI(Original)+PV-RCNN	BEV_AP	**94.63**	**90.28**	88.43
	3D_AP	**92.97**	**83.11**	**82.38**
VOXEL-RCNN Baseline	BEV_AP	95.35	90.83	88.64
	3D_AP	92.06	82.64	80.10
UYI(TFMM)+VOXEL-RCNN	BEV_AP	93.37	90.65	88.77
	3D_AP	92.10	82.99	80.46
UYI(Original)+VOXEL-RCNN	BEV_AP	**95.76**	**91.06**	**88.84**
	3D_AP	**92.31**	**83.09**	**82.57**

TABLE II
COMPARISON OF DETECTION PERFORMANCE BETWEEN USING TFMM AND UYI MODULES FOR PSEUDO POINT CLOUD GENERATION ON PV-RCNN AND VOXEL-RCNN. IN THIS TABLE WE REPLACE OUR UYI MODULE BY A SIMPLE TFMM POINT CLOUD AUGMENTATION BLOCK.

Methods	Metrics	Easy	Medium	Hard
PV-RCNN Baseline	BEV_AP	92.86	88.93	**88.74**
	3D_AP	91.99	82.86	80.40
UYI(w/o attention and sentient module) + PV-RCNN	BEV_AP	93.40	**90.40**	88.53
	3D_AP	91.94	82.82	80.22
UYI(w/ attention and sentient module) + PV-RCNN	BEV_AP	**94.63**	90.28	88.43
	3D_AP	**92.97**	**83.11**	**82.38**
VOXEL-RCNN Baseline	BEV_AP	95.35	90.83	88.64
	3D_AP	92.06	82.64	80.10
UYI(without sentient loss) + VOXEL-RCNN	BEV_AP	94.88	91.05	88.76
	3D_AP	92.23	82.48	81.66
UYI(with sentient loss) + VOXEL-RCNN	BEV_AP	**95.76**	**91.06**	**88.84**
	3D_AP	**92.31**	**83.09**	**82.57**

TABLE III
COMPARISON OF DETECTION RESULTS BETWEEN ADDING ATTENTION AND SENTIENT MODULE OR NOT IN UYI MODULE FOR PSEUDO POINT CLOUD GENERATION ON PV-RCNN AND VOXEL-RCNN.

projected position of the target when the points information for positioning are not rich. Note that geometric calibration parameters are also provided on Waymo[40], Nuscences[41], etc., while other parameters such as the average length of bounding boxes can be computed, therefore we believe that the method proposed in this paper is not be limited to KITTI.

IV. EXPERIMENTS

In this section we present the results showing the performance improvements of typical baseline models by the high-quality point cloud enhanced with our UYI module. In part 1, we first describe the performance of our UYI module on VoxelRCNN [15] and PV-RCNN [16]. We compare our full UYI module with the UYI Module that replaces the point cloud generation block with the previous TFMM method. In part 2, we perform ablation studies on the performance improvement using UYI module on typical 3D models.

A. Comparision with traditional flipping and best matching method

In this section, we replace, in the UYI module, the point cloud generation block based on projection awareness positioning (PAP) with a simple TFMM point augmentation block as proposed in BtcDet (Section II.B).

We followed the TFMM method proposed in BtcDet to perform the best matching of the vehicle point cloud completion. A random scaling of [0.95, 1.05],was used and the number of point features was set to 4. The performance results of the point cloud enhanced for comparison with two variants of UYI module on Voxel-RCNN and PV-RCNN are in Table II. The improvement of the original UYI module is greater than that of the UYI using TFMM on the PV-RCNN and Voxel-RCNN baselines. The lead is most significant in 3D_AP in the Hard category, reaching +2.03% and +2.11% on PV-RCNN and VOXEL-RCNN, respectively.

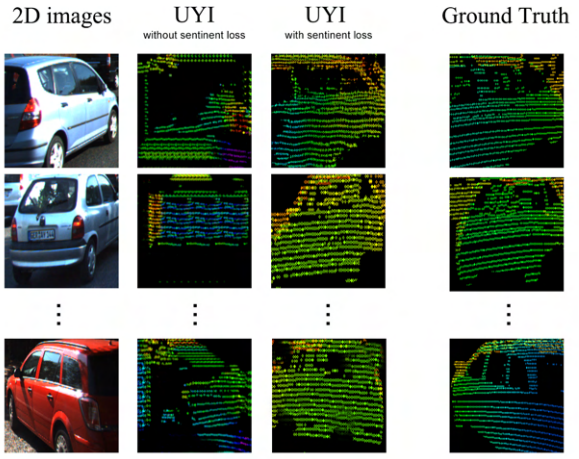

Fig. 5. Comparison of generated 2D point cloud images between UYI with and without attention and sentient module on validation set: On the left is the input 2D image, the second from the left is the output of UYI without attention and sentient module, the third from the left is the output of UYI with attention and sentient module, and the right is groundtruth. We want to minimize the gap between the output of UYI and the groundtruth, so the criterion for optimization is how similar the output of UYI is to its corresponding groundtruth.

B. Impact of attention and sentient module

In this section we assess the impact on vehicle detection results of using the attention and sentient modules in UYI. As in the previous section, we use PV-RCNN and VOXEL-RCNN because they are sufficiently strong baselines. The results are shown in Table III and Figure 5. The improvement of UYI with attention and sentient module is greater than that of the UYI without attention and sentient module. For PV-RCNN, in the category of 3D_AP we have a 2.16% increment for Hard objects, 0.29% in medium and 1.03% in easy. In VOXEL-RCNN we observe a more significant improvement for both BEV_AP and 3D_AP, with a 0.91% increase in 3D_AP for the hard case and with smaller boost in Medium and Easy. From the evaluations we can find that the attention and sentient modules can indeed generate better quality point cloud images, and these point cloud images of higher quality

Methods	Average Error(distance $\geq$ 50m)	Mean Squared Error(distance $\geq$ 50m)	Average Error(Full)	Mean Squared Error(Full)
Simple Linear Regression(MLR)	4.78 meters	45.71	2.03 meters	25.45
MLR+Polynomial Features	**3.06 meters**	**24.33**	**1.56 meters**	**21.91**
Previous Triangle Method	11.61 meters	321.00	4.09 meters	66.47

TABLE I
COMPARISON OF METHODS FOR ESTIMATING OBJECT DEPTH FROM RGB. CONSIDERING THAT THE ERROR INCREASES WITH THE DISTANCE OF THE OBJECT, THE RESULTS ARE DIVIDED INTO THE TEST OF LONG-DISTANCE OBJECTS GREATER THAN 50 METERS AND THE TEST OF ALL CASES.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Method	Reference	Metrics(R40)	UYI(Ours)			TWISE(Global level)			TWISE(Object level)			Baseline		
			Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
SECOND [17]	SENSORS 2018	Car BEV_AP	**92.27**	87.91	**86.63**	39.96	30.45	24.61	91.89	87.29	86.14	91.92	87.92	85.39
		Car 3D_AP	88.88	**79.15**	**76.07**	16.26	11.72	8.99	**89.39**	78.94	76.23	88.45	78.75	75.72
		Ped BEV_AP	**60.86**	**57.03**	**51.54**	3.78	3.03	3.19	53.33	47.08	42.81	58.67	53.20	48.64
		Ped 3D_AP	**53.94**	**49.66**	**45.00**	0.96	1.01	1.15	48.40	42.09	37.18	53.64	47.45	42.74
		Cyc BEV_AP	**85.63**	**66.08**	**61.93**	0.15	0.26	0.26	76.63	56.46	52.63	82.39	69.84	65.05
		Cyc 3D_AP	**78.04**	**60.34**	**56.54**	0.00	0.00	0.00	73.45	52.84	49.00	76.67	62.35	58.25
PointPillars [42]	CVPR 2019	Car BEV_AP	92.04	**87.93**	85.39	90.90	83.41	79.54	87.58	83.25	80.60	92.04	87.74	86.65
		Car 3D_AP	**88.51**	**78.78**	**75.72**	83.80	71.82	67.30	78.52	70.35	67.40	88.24	78.25	75.32
		Ped BEV_AP	**69.74**	**62.75**	**57.57**	48.87	43.08	39.56	60.41	54.21	49.82	59.50	54.37	50.13
		Ped 3D_AP	**63.50**	**56.79**	**51.54**	46.35	39.54	35.58	55.06	48.73	44.26	55.11	49.57	44.78
		Cyc BEV_AP	**88.23**	68.17	**64.34**	51.35	34.17	31.41	88.19	67.54	63.57	88.21	**68.63**	64.13
		Cyc 3D_AP	**87.75**	65.13	**61.94**	47.49	30.97	28.42	82.04	64.01	59.44	85.84	**65.89**	61.38
Point-RCNN [43]	CVPR 2019	Car BEV_AP	**93.13**	**88.96**	**86.76**	79.58	54.86	45.48	84.82	70.53	63.98	92.99	88.67	84.64
		Car 3D_AP	**89.18**	**79.82**	75.88	72.64	48.73	41.21	67.57	55.92	50.23	88.86	78.62	**77.38**
		Ped BEV_AP	**73.82**	**64.62**	**55.72**	50.65	39.30	34.01	65.76	56.58	48.00	64.17	57.81	51.16
		Ped 3D_AP	**69.41**	**58.79**	**50.04**	51.81	41.72	34.62	63.91	53.65	45.29	62.29	54.88	48.22
		Cyc BEV_AP	**92.66**	**75.75**	**71.10**	23.65	13.79	13.28	66.07	44.58	40.19	92.31	73.47	70.07
		Cyc 3D_AP	**91.97**	**72.58**	**68.04**	23.63	13.76	13.25	65.69	42.33	39.67	91.88	70.79	67.42
PartA2 [44]	TPAMI 2020	Car BEV_AP	**92.96**	**88.21**	**87.56**	59.32	44.67	37.38	90.44	83.74	81.55	90.86	86.26	85.80
		Car 3D_AP	**91.83**	**82.06**	**79.57**	46.67	32.75	26.56	84.14	72.62	69.98	89.93	80.24	77.89
		Ped BEV_AP	**71.97**	**65.65**	**60.53**	26.94	20.96	17.63	44.19	36.46	33.26	59.01	52.15	48.15
		Ped 3D_AP	**66.85**	**59.61**	**54.28**	20.21	15.67	12.54	40.11	32.92	28.89	54.62	48.32	43.41
		Cyc BEV_AP	**91.79**	77.66	**74.28**	13.99	8.28	6.81	62.64	40.85	37.62	89.81	**77.69**	73.40
		Cyc 3D_AP	**89.30**	72.06	**67.79**	11.22	6.35	6.16	59.71	38.61	35.32	86.12	**72.31**	67.46
PV-RCNN [16]	CVPR 2020	Car BEV_AP	**94.63**	**90.28**	88.43	68.85	57.83	51.89	93.87	87.30	85.04	92.86	88.93	88.74
		Car 3D_AP	**92.97**	**83.11**	**82.38**	52.92	42.12	37.45	90.38	81.12	78.69	91.99	82.86	80.40
		Ped BEV_AP	**70.95**	**63.64**	**58.61**	42.75	33.01	30.08	48.67	40.31	36.56	58.91	51.43	47.67
		Ped 3D_AP	**66.43**	**59.17**	**54.00**	42.71	32.90	29.29	45.43	37.53	33.33	55.78	48.42	43.87
		Cyc BEV_AP	**91.35**	**76.53**	**71.71**	24.96	14.75	13.53	68.96	45.13	42.33	87.18	70.35	66.21
		Cyc 3D_AP	**87.09**	**72.13**	**67.58**	24.96	14.43	13.17	67.67	43.46	40.70	86.85	68.00	63.83
VOXEL-RCNN [15]	AAAI 2021	Car BEV_AP	**95.76**	**91.06**	**88.84**	72.65	64.66	58.13	95.37	89.02	88.53	95.35	90.83	88.64
		Car 3D_AP	92.31	**83.09**	**82.57**	53.43	45.76	39.95	**92.49**	82.88	80.38	92.06	82.64	80.10
	UYI with YOLOX-s	Car BEV_AP	94.36	89.13	86.54	-	-	-	-	-	-	95.35	90.83	88.64
	UYI with YOLOX-s	Car 3D_AP	91.21	82.83	80.54	-	-	-	-	-	-	92.06	82.64	80.10
BtcDet [11]	AAAI 2022	Car BEV_AP	**93.89**	**91.79**	**89.28**	89.43	87.01	79.33	94.99	86.52	84.15	93.46	89.53	87.44
		Car 3D_AP	**92.81**	**85.80**	**83.08**	87.57	77.44	76.75	90.69	79.68	75.33	92.17	82.39	80.96
SFD [19]	CVPR 2022	Car BEV_AP	**96.10**	91.68	91.08	-	-	-	-	-	-	95.85	91.92	91.41
		Car 3D_AP	**95.64**	**88.42**	**85.70**	-	-	-	-	-	-	95.01	88.31	85.69

TABLE IV

PERFORMANCE IMPROVEMENT ON STATE-OF-THE-ART METHODS ON THE KITTI VALIDATION SET. FROM LEFT TO RIGHT: PERFORMANCE AFTER ADDING THE UYI MODULE, USING TWISE AS GLOBAL-LEVEL DEPTH COMPLETION INSTEAD OF UYI, USING TWISE TO REPLACE THE GAN MODULE FOR OBJECT LEVEL COMPLETION IN UYI, AND BASELINES.

Metrics(in average)	UYI(Ours)	TWISE(Global level)	TWISE(Object level)	Original KITTI
Inference speed	$\sim 32ms$	$\sim 41ms$		-
LiDAR points generated per frame	14053	192071	56117	-
Total LiDAR points per frame	133278	311296	175342	119225

TABLE V

COMPARISON OF INFERENCE SPEED OF UYI AND TWISE. THE SPEED IS CALCULATED AS AVERAGES FOR INFERENCE OVER ALL SAMPLES ON KITTI'S VALIDATION SET ON A SINGLE 3090.

produce better 3D point clouds, therefore leading to more accurate detections. In fact, UYI without attention and sentient module produces unsatisfactory results on most samples in the validation set. And we can observe that UYI with attention and sentient modules can make UYI to pay more attention to the texture characteristics of the reconstructed objects, making the reconstruction process more accurate.

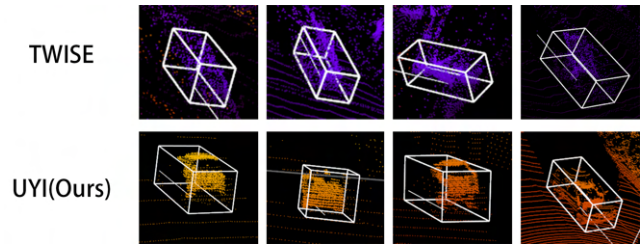

Fig. 6. Comparison between TWISE and UYI on long tail problem (best when viewed in color).

C. Comparison with State-of-the-art methods

To better illustrate the general improvements among previous models, we performed ablation experiments on several existing outstanding models, each of which was state-of-the-art when introduced. These are SECOND [17], PointPillars [42], PartA2 [44], PV-RCNN [16], Voxel-RCNN [15], BtcDet [11], SFD-Net [19]. The results are shown in Table IV and V. In columns 7 to 12 of the table, we present two sets of experiments to compare UYI with TWISE as another depth completion method:

- Use TWISE as the global point cloud completion method (the global level completion).
- Use foreground TWISE to replace the GAN in UYI as the point cloud generation method (object level completion)

Since the authors of SFD, BtcDet, and VOXEL-RCNN did not publish the training config for the Pedestrian and Cyclist class, we could not reproduce those metrics accurately. For this reason in the comparison of Pedestrians and Cyclists, these three models have been excluded. Also, since the TWISE-based pseudo point cloud generation method has been integrated into the SFDNet model, it is not used in the comparison experiment of TWISE. It can be seen that our UYI still achieved the most satisfying results, while the object-level TWISE achieved better overall performance than the global TWISE.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

From the comparison of UYI and Baseline, we can also observe that more significant progress is made in the Moderate and Easy categories. UYI is designed to improve the quality of point clouds and relies on 2D detectors. Compared to 3D detectors, 2D detectors are easier to detect distant objects. However, due to occlusion and shadows, etc., the actual available pixel area of some road objects is only a small part of the area. For closer objects, these regions are still large enough to generate accurate point clouds; as for distant objects, the quality of the generation is impacted when the 2D bounding box is small, thus making it difficult to generate satisfactory point clouds.

For the car category, it can be seen that the high-quality point clouds generated by the UYI module have led to significant improvements compared to the baseline. More notable results were brought by PartA2, PV-RCNN, and BtcDet, with also similar improvements in BEV_AP. For the pedestrian category, more notable results were brought by PartA2 and PV-RCNN, with also similar improvements in BEV_AP. We achieved 10.87%,11.29%, and 12.23% leap forward in the Hard, Moderate, and Easy categories of PartA2's 3D_AP. We also implemented an UYI module based on YOLOX-s as comparison to YOLOX-l on VOXEL-RCNN. The results show that the performance of the 2D detector affects the overall performance of UYI.

Note that the improvement in SFDNet is quite limited, we attribute it to the CPConvs in SFDNet which is actually designed for processing a small number of raw points with a large number of pseudo points. It is therefore not suitable for processing a large number of LiDAR points generated by GAN plus a large number of pseudo points. Given that pedestrian and cyclist classes are smaller objects than vehicles and suffer more from LiDAR sparsity, they should have larger performance gains, which is verified in Table IV: the improvement of the pedestrian category is more significant than that of vehicle category, with an average of 7.74%, 7.08% and 6.37% increment compared to baselines on easy/medium and hard category of 3D_AP among the models we evaluated. A similar trend is observed in the cyclist category as well.

According to our statistics in TABLE V, the UYI module only adds a latency of about 32ms to the model in general while the latency of using TWISE global completion is about 41ms. Object level completion latency using TWISE is almost the same. At the same time, point cloud completion using object-level methods can greatly reduce the number of generated points. Compared with the global TWISE point cloud completion, UYI as an object-level-based method reduces the number of generated point clouds by up to 92.68% in general, which makes us believe that object-level depth completion could be a promising direction for efficient 3D detection. In Figure 6, we present some generating samples of TWISE and UYI, it can be seen that with the native benefit of object level completion, our method can bring quite effective improvements to the long tail problem. Therefore our UYI Module could generally improve the point cloud quality and improving the quality of point clouds indeed leads to an improvement in the performance of 3D detection.

V. Conclusions

In this paper, we discussed how the sparsity of distant LiDAR point clouds affects point cloud completion tasks. To improve the quality of point clouds, we propose the UYI Module, which uses a GAN-based cross-modal point cloud generator to generate LiDAR information by taking 2D images as input and trying to learn the perceptual connection to point clouds with color encoding. This strategy improves the quality of the point cloud by passing the generated high-density vehicle point cloud through the PAP process. Thanks to the plug-and-play feature of the module, we can readily push the performance of existing state-of-the-art models to a new level. Although our work successfully demonstrates the benefits of higher point cloud quality for 3D detection models, there is still room for model efficiency optimizations. Designing models for occlusion removal and accelerated point cloud generation could be a promising direction in the future.

References

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," 07 2017, pp. 6526–6534.

[2] 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.

[3] 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 (CVPR)*, June 2018.

[4] L.-H. Wen and K.-H. Jo, "Fast and accurate 3d object detection for lidar-camera-based autonomous vehicles using one shared voxel-based backbone," *IEEE Access*, vol. 9, pp. 22 080–22 089, 2021.

[5] Z. Zhang, M. Zhang, Z. Liang, X. Zhao, M. Yang, W. Tan, and S. Pu, "Maff-net: Filter false positive for 3d vehicle detection with multi-modal adaptive feature fusion," *arXiv preprint arXiv:2009.10945*, 2020.

[6] L. Xie, C. Xiang, Z. Yu, G. Xu, Z. Yang, D. Cai, and X. He, "Pi-rcnn: An efficient multi-sensor 3d object detector with point-based attentive cont-conv fusion module," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 12 460–12 467.

[7] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[8] S. Imran, X. Liu, and D. Morris, "Depth completion with twin surface extrapolation at occlusion boundaries," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 2583–2592.

[9] J. Gu, Z. Xiang, Y. Ye, and L. Wang, "Denselidar: A real-time pseudo dense depth guided depth completion network," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1808–1815, 2021.

[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)*, September 2018.

[11] Q. Xu, Y. Zhong, and U. Neumann, "Behind the curtain: Learning occluded shapes for 3d object detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 2893–2901, Jun. 2022.

[12] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[13] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving," in *International Conference on Learning Representations*, 2020.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

[15] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel r-cnn: Towards high performance voxel-based 3d object detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, pp. 1201–1209, May 2021.

[16] 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 (CVPR)*, June 2020.

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

[18] D. Chan, H. Buisman, C. Theobalt, and S. Thrun, "A Noise-Aware Filter for Real-Time Depth Upsampling," in *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008*. Marseille, France: Andrea Cavallaro and Hamid Aghajan, Oct. 2008.

[19] X. Wu, L. Peng, H. Yang, L. Xie, C. Huang, C. Deng, H. Liu, and D. Cai, "Sparse fuse dense: Towards high quality 3d detection with depth completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 5418–5427.

[20] K. S. Kim, D. Kim, and J. Kim, "Hardness on style transfer deep learning for rococo painting masterpieces," in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2019, pp. 452–454.

[21] X. Gao, Y. Tian, and Z. Qi, "Rpd-gan: Learning to draw realistic paintings with generative adversarial network," *IEEE Transactions on Image Processing*, vol. PP, pp. 1–1, 08 2020.

[22] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Gaugan: semantic image synthesis with spatially adaptive normalization," 07 2019, pp. 1–1.

[23] X. Huang, A. Mallya, T.-C. Wang, and M.-Y. Liu, "Multimodal conditional image synthesis with product-of-experts GANs," in *ECCV*, 2022.

[24] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[25] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Occlusion-aware r-cnn: Detecting pedestrians in a crowd," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[26] P. Follmann, R. König, P. Härtinger, M. Klostermann, and T. Böttger, "Learning to see the invisible: End-to-end trainable amodal instance segmentation," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 1328–1336.

[27] P. Hu, J. Ziglar, D. Held, and D. Ramanan, "What you see is what you get: Exploiting visibility for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[28] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[29] H. Wang, M. Wang, Z. Che, Z. Xu, X. Qiao, M. Qi, F. Feng, and J. Tang, "Rgb-depth fusion gan for indoor depth completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6209–6218.

[30] D. S. Tan, C.-Y. Yao, C. Ruiz Jr, and K.-L. Hua, "Single-image depth inference using generative adversarial networks," *Sensors*, vol. 19, no. 7, p. 1708, 2019.

[31] Y. Li, Y. Wang, Z. Lu, and J. Xiao, "Depthgan: Gan-based depth generation of indoor scenes from semantic layouts," *arXiv preprint arXiv:2203.11453*, 2022.

[32] A. CS Kumar, S. M. Bhandarkar, and M. Prasad, "Monocular depth prediction using generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 300–308.

[33] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-gan: a point cloud upsampling adversarial network," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 7203–7212.

[34] H. Liu, H. Yuan, J. Hou, R. Hamzaoui, and W. Gao, "Pufa-gan: A frequency-aware generative adversarial network for 3d point cloud upsampling," *IEEE Transactions on Image Processing*, vol. 31, pp. 7389–7402, 2022.

[35] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[36] 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.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[38] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[39] J. Engel, "Polytomous logistic regression," *Statistica Neerlandica*, vol. 42, pp. 233 – 252, 04 2008.

[40] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[41] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[42] 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.

[43] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," 06 2019, pp. 770–779.

[44] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 8, pp. 2647–2664, 2021.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.