Learning to Drop Points for LiDAR Scan Synthesis

Kazuto Nakashima and Ryo Kurazume

***Abstract*— 3D laser scanning by LiDAR sensors plays an important role for mobile robots to understand their surroundings. Nevertheless, not all systems have high resolution and accuracy due to hardware limitations, weather conditions, and so on. Generative modeling of LiDAR data as scene priors is one of the promising solutions to compensate for unreliable or incomplete observations. In this paper, we propose a novel generative model for learning LiDAR data based on generative adversarial networks. As in the related studies, we process LiDAR data as a compact yet lossless representation, a cylindrical depth map. However, despite the smoothness of real-world objects, many points on the depth map are dropped out through the laser measurement, which causes learning difficulty on generative models. To circumvent this issue, we introduce measurement uncertainty into the generation process, which allows the model to learn a disentangled representation of the underlying shape and the dropout noises from a collection of real LiDAR data. To simulate the lossy measurement, we adopt a differentiable sampling framework to drop points based on the learned uncertainty. We demonstrate the effectiveness of our method on synthesis and reconstruction tasks using two datasets. We further showcase potential applications by restoring LiDAR data with various types of corruption.**

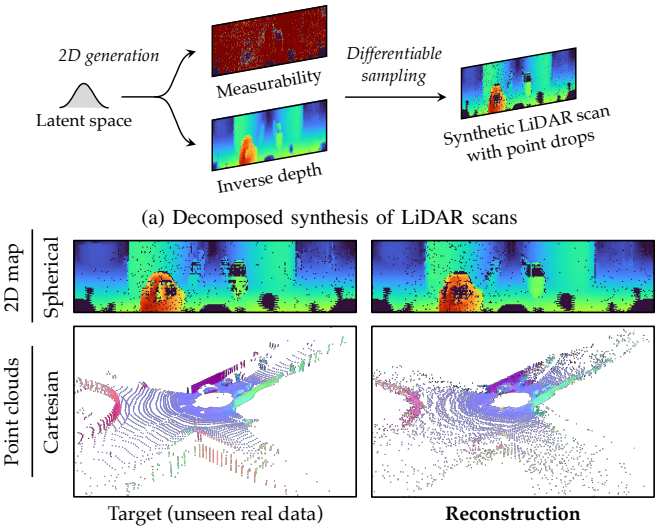
I. Introduction

3D scene understanding is indispensable for mobile robots to detect obstacle objects, find traversable paths, and explore the real world. A typical representation of 3D scenes is a point cloud, which can be measured by various range-scanning devices such as 3D LiDARs and RGB-D cameras. In particular, 3D LiDARs are widely used for autonomous driving systems to enable their perception capabilities such as SLAM, object detection, and semantic segmentation.

A 3D LiDAR calculates distances based on the time-of-flight of pulsed laser emitted and reflected at multiple elevation/azimuth angles. However, to gain the angular resolution and precision, it requires many scanner units and sufficient housing size, which could conflict with system requirements. Moreover, the measured points could have noises under adverse weather. The purpose of this study is to reconstruct high-quality point clouds from such limited or corrupted observations. However, it is non-trivial to manipulate a number of raw points with semantic consistency. One promising approach is to learn a generative model of point clouds as scene priors and to find the high-quality sample nearest to the observation.

Generative models have received considerable attention owing to recent advances in representation learning with deep neural networks. In particular, generative adversarial networks (GANs) [2] are a popular framework for building a generative model. GANs have been actively studied on 2D tasks such as generating photo-realistic natural images [3, 4], while they can also be used to model 3D data such as point clouds. Several studies [5, 6] proposed generative models that produce point clouds as a set of unordered points and demonstrated on data uniformly sampled from CAD models [7] In contrast, the point clouds by 3D LiDARs are often large, non-uniform, and sparse, as they are measured by radiated lasers centering on the sensor head. In other perception tasks with LiDAR data, projection-based 2D representations have also been studied, such as sensor-view depth images by spherical projection [8, 9] and bird's eye view [10]. The 2D approaches have advantages in computational efficiency: the processing can be designed with 2D convolutional networks, less problem on the sparsity, and the representation is compact yet lossless for spherical projection. In semantic point segmentation [9, 10] and object detection tasks [8], these approaches have gained performance in practice.

The goal of this study is to build a generative model of LiDAR data based on the 2D representation [8]–[12]. In particular, we use the sensor-view representation. Caccia *et al.* [11] is closely related to our task, which trained GANs and variational autoencoders (VAEs) with Cartesian points projected onto sensor-view grids. However, we found that training generative models on this 2D representation



(a) Decomposed synthesis of LiDAR scans

(b) Reconstruction of unseen data from KITTI [1] test set

Fig. 1. A concept of our proposed generative model for LiDAR data. (a) With only noisy training data, our proposed method can learn an underlying complete depth map and a measurability map that simulates dropout noises. (b) The trained model can be used to reconstruct unseen data.

Kazuto Nakashima and Ryo Kurazume are with Faculty of Information Science and Electrical Engineering, Kyushu University, 744 Motoka Fukuoka, Japan. k_nakashima@irvs.ait.kyushu-u.ac.jp, kurazume@ait.kyushu-u.ac.jp

is still challenging because of LiDAR-specific noises. As seen in Fig. 1(b), the projected image involves scattered dropout noises. The noises are due to the reflection failure of laser measurement, hereinafter called *point-drops*. Caccia *et al.* [11] handled this issue by interpolating all the point-drops by neighbor pixels, however, the heuristic imputation could cause unnatural structures.

In this paper, we introduce a novel GAN framework to synthesize *depth* with *uncertainty* from *styles*, named DUSty. Our key idea is to introduce measurement uncertainty into the generation process so that the underlying complete signals can be modeled implicitly. As shown in Fig. 1(a), our model produces a complete depth map and the corresponding measurability to sample realistic *dusty* point-drops. Our proposed model can be trained only with raw LiDAR data involving point-drops. We evaluated our method on two LiDAR datasets for driving scenes: KITTI [1] and MPO [13]. We demonstrate that our method successively synthesizes realistic LiDAR data, learning the underlying complete shape. We also demonstrate the reconstruction capability for unseen real data in various corruption settings. The code will be available at `https://github.com/kazuto1011/dusty-gan`.

The main contributions can be summarized as follows:

- We propose a noise-aware GAN framework DUSty, which models a complete depth map and the corresponding measurability from real LiDAR data.
- We introduce a differentiable relaxation to learn the discrete distribution of point-drops.
- We demonstrate the effectiveness of our approach on synthesis and reconstruction tasks on two LiDAR datasets.

II. Related Work

A. Synthesizing LiDAR Data

Generative modeling of point clouds is a challenging task emerged in recent years. Most studies focused on small point sets sampled from CAD objects [5], but rarely on LiDAR point clouds. There have been studies based on simulated environments [9, 12], but their diversity was limited to defined cases. Only Caccia *et al.* [11] worked on modeling LiDAR data with two popular frameworks for deep generative models: variational autoencoders (VAEs) and GANs. The authors revealed that 2D representation was much better for LiDAR data than for point sets. However, the study evaluated quantitative performance on reconstruction tasks only for VAEs. In this paper, we provide the results of both quality and diversity metrics for GANs on the synthesis task and demonstrate the improvement in the reconstruction task. In particular, we found that simulating *point-drops* was important for training GANs effectively.

For obstacle detection tasks, recent studies [9, 12] revealed that simulating point-drops could mitigate the domain gap between the real and simulated environments. Wu *et al.* [9] calculated the spatial prior from real LiDAR scans to sample dropout noises. However, the sampled noise is independent of the instance. Manivasagam *et al.* [12] trained U-Net to predict point-drops as a binary classification task. However, the drop prediction requires clean LiDAR data superimposed by multiple scans, and the classification probability is not calibrated. In contrast, our approach only requires noisy LiDAR data and can implicitly model the data-dependent uncertainty of point measurability. As we discuss in Section V, our learned generator can be used to estimate realistic point-drops from given observable points.

B. Modeling Image Noises

Some studies leveraged noisy training data for denoising images [14] and synthesizing novel clean images [15]–[17]. Lehtinen *et al.* [14] eliminated various types of synthetic noises, including multiplicative Bernoulli noise, which randomly masked a pixel to zero with a certain probability. Bora *et al.* [16] learned GANs from partially-zeroed images. We assume that point-drops in LiDAR sensing can be categorized as this type of noise. However, these studies [14, 16] assume a predefined noise model, such as pixel-level or patch-level dropouts with a fixed probability. Kaneko and Harada [15] proposed GAN architectures that can estimate noise distribution, but multiplicative binary noises are not covered. Li *et al.* [17] learn the distribution of binary noises, however, the noise is independent of signals and approximated with sigmoid outputs.

As in [14, 16, 17], we assume multiplicative Bernoulli noises to mimic the point-drops on the LiDAR scan map. However, we do not set the probability; instead, we aim to learn the pixel-wise probability model. Specifically, in our case, the noise distribution type is defined, but the noise level and the generative relationship to the depth modality are unknown owing to the complex physical factors in measurement. Meanwhile, generating binary masks with Bernoulli distribution is not differentiable, which means inability to learn the parameterized probability model by backpropagation. Therefore, this study employs the Gumbel-Softmax trick [18, 19], a reparametrization approach for discrete sampling.

III. Our Approach

A. Data Representation

This paper assumes a bijective image representation [11], which is directly acquired from horizontal scanning of multiple laser receptors aligned vertically. For example, a LiDAR that emits W pulses for H elevation angles produces $H \times W$ points with measured distances x, which can be considered as a cylindrical depth map with a size of $H \times W$. An existing study [11] trained GANs with the depth map representation. However, we found it difficult to learn stably without their preprocessing that narrows the spatial range of the depth map. Instead, we propose another approach for data representation motivated by monocular depth estimation. In a typical setting of LiDARs, most pixels of the acquired depth map represent a near-to-mid range, and most of the dynamic range is occupied by few distant points in a long-tail distribution. Therefore, to gain the dynamic range of the

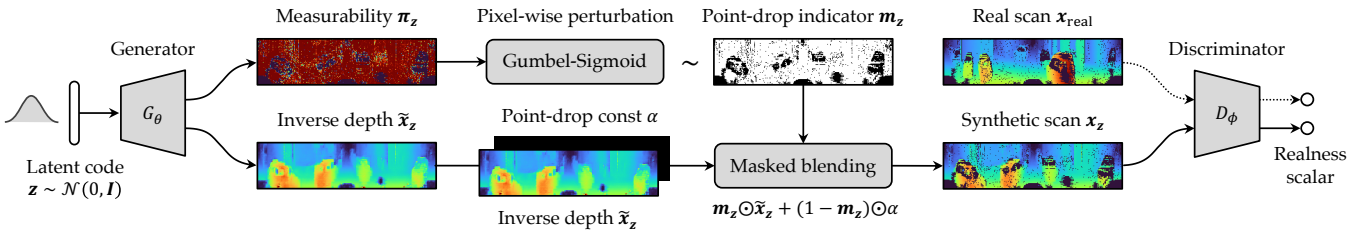


Fig. 2. Overview of our proposed GAN framework. Generator G_θ produces the inverse depth map $\tilde{x}_z$ and the corresponding measurability map π_z, from the sampled latent code $z \sim \mathcal{N}(0, I)$. The binary mask m_z indicating point-drops is sampled from the measurability map π_z. Discriminator D_ϕ distinguishes the processed inverse depth map x_z from the real data x_{real}.

majority regions, we transform the LiDAR depth map into an inverse form $1/x$, *i.e.*, an inverse depth map. We will discuss how we can synthesize the cylindrical inverse depth maps throughout this paper.

B. Decomposed Synthesis of LiDAR Scans

A GAN typically consists of two networks: a generator G_θ and a discriminator D_ϕ, where θ and ϕ are trainable parameters. In image synthesis tasks, G_θ maps a latent variable $z \sim N(0, I)$ to an image $x_z = G_\theta(z)$, whereas D_ϕ distinguishes the generated image x_z from sampled real images x_{real}. The networks are trained in an alternating fashion by minimizing the adversarial objective, *e.g.*, the following non-saturating loss [2]:

$$\mathcal{L}_D = -\mathbb{E}_x[\log D_\phi(x_{\text{real}})] - \mathbb{E}_z[\log(1 - D_\phi(G_\theta(z)))] \quad (1)$$

$$\mathcal{L}_G = -\mathbb{E}_z[\log D_\phi(G_\theta(z))]. \quad (2)$$

Here, G_θ is modeled as a decoder convolutional network that gradually upscales the spatial resolution toward the final image. Although synthesizing natural images [3] has been successfully achieved, we found it difficult to learn noisy depth maps with a stack of naïve convolutions.

Let us consider modeling a scene where a vehicle is moving, for example. If all the points are observable on the LiDAR receptor, the shape of the vehicle would be smoothly morphed on the depth map. In this case, generating a sequence of the depth maps is relatively easy if we move a sampling point in the continuous data space by manipulating a set of convolution kernels. However, in practice, some points are randomly dropped at the pixel level because of complex physical factors (such as mirror diffusion and material reflectance). In these cases, the scattered point-drops make the morphing discrete and cause learning difficulty. Our key idea is to simulate the point-drop phenomenon by a probabilistically invertible function to learn the underlying manifold of smooth depth separately.

To this end, we introduce a decomposed depth map representation into adversarial training. An overview of our approach is depicted in Fig. 2. First, we assume a dense inverse depth map $\tilde{x}_z \in \mathbb{R}^{H \times W}$ and a measurability map $\pi_z \in \mathbb{R}^{H \times W}$ that represents the probability of laser reflection (*e.g.*, mirror-like materials have low confidence in returning lasers due to the diffuse reflection, which results in point-drops). We sample a binary mask $m_z \in \{0, 1\}^{H \times W}$ from the measurability map π_z.

$$m_z \sim \text{Bernoulli}(\pi_z) \quad (3)$$

The final synthetic scan x_z is produced by masking the dense inverse depth $\tilde{x}_z$ using the binary mask m_z:

$$x_z = m_z \odot \tilde{x}_z + (1 - m_z) \odot \alpha, \quad (4)$$

where $\odot$ is an element-wise product, and α is a constant value representing the point-drop.

Note that sampling m_z is not differentiable, and the gradients cannot be propagated downstream. Therefore, we reparameterize m_z with the straight-through (ST) Gumbel-Sigmoid[1] distribution [18, 19] to estimate the gradients. We first introduce the continuous relaxation $\tilde{m}_z$ as follows:

$$\tilde{m}_z = \text{sigmoid}\left(\frac{e + g_1 - g_2}{\tau}\right), e = \ln\left(\frac{\pi_z}{1 - \pi_z}\right), \quad (5)$$

where e is a logit of π_z and $g_1, g_2 \in \mathbb{R}^{H \times W}$ are i.i.d. samples from Gumbel$(0, I)$, which perturbs e at the pixel level. Moreover, τ is a hyperparameter called temperature, which controls the slope of the sigmoid function. With a low value of τ, the soft mask $\tilde{m}_z$ approaches a binary mask, but the variance of the gradients is large. Finally, the soft mask $\tilde{m}_z$ is discretized into the binary mask m_z at each pixel location (i, j) as follows:

$$m_z^{i,j} = \begin{cases} 1 & \tilde{m}_z^{i,j} \geq 0.5 \\ 0 & \tilde{m}_z^{i,j} < 0.5 \end{cases} \quad (6)$$

ST Gumbel-Sigmoid [18, 19] approximates this thresholding by an identity function to enable gradient propagation: *i.e.*, we use the stochastic binary mask m_z in the forward step, and we approximate the gradients by $\tilde{m}_z$ during the backward step. In summary, instead of modeling raw data directly, our generator G_θ aims to jointly produce the dense inverse depth $\tilde{x}_z$ and the associated confidence of point measurability π_z. Discriminator D_ϕ only determines the realness of the processed inverse depth map x_z in Eq. 4.

C. Multilevel Gumbel Sampling

As introduced above, the pixel-level Gumbel sampling offers the differentiable binarizer. The stochastic behavior can learn the pixel-level uncertainty. However, we observed another level of uncertainty appear in real LiDAR data. For

[1]Gumbel-Sigmoid is a binary case of Gumbel-Softmax.

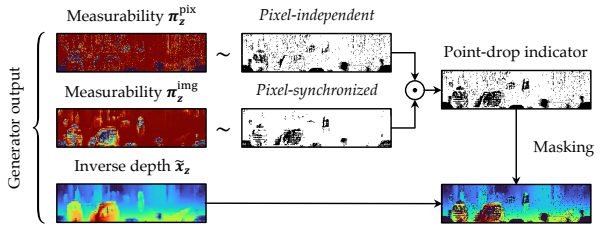


Fig. 3. Multilevel Gumbel sampling. $\odot$ denotes an element-wise product.

example, on the depth map of KITTI, some cars have point-drops on only the transparent windows, while others are missing most of the body. We assume colors and materials cause this object-level multimodality of point drops even for the same shape. The pixel-level Gumbel sampling is hard to catch this distribution, and as a result, the variation would be leaked into the latent space. We introduce a simple extension to separate the object-level uncertainty from the latent space.

Revisiting Eq. 5, we made *pixel-independent* perturbations to model the measurability. Here, we call the measurability and the sampled mask as $\boldsymbol{\pi}_z^{\text{pix}}$ and $\boldsymbol{m}_z^{\text{pix}}$, respectively. We now consider another measurability branch $\boldsymbol{\pi}_z^{\text{img}}$ to be added *pixel-synchronized* perturbations, where we sample scalars $g_1, g_2 \sim \text{Gumbel}(0, 1)$ in training and cancel them in testing to switch to a deterministic mask $\boldsymbol{m}_z^{\text{img}}$. The final mask $\boldsymbol{m}_z$ is determined by element-wise product $\boldsymbol{m}_z^{\text{pix}} \odot \boldsymbol{m}_z^{\text{img}}$.

IV. Synthesis by Latent Sampling

We first report the synthesis performance of our proposed methods by evaluating the similarity between the distributions of real data and sampled synthetic data. We further discuss the reconstruction performance in Section V.

A. Dataset

We built the inverse depth maps from two types of LiDAR datasets: KITTI [1] and MPO [13], which have different vertical resolutions and scenes. The LiDARs of both the KITTI and MPO datasets have the same distance range of $(0.9m, 120m)$, where we take the reciprocal of the distances and normalize the range into $(-1, 1)$ for the training stability of GANs. When reconstructing point clouds from the inverse depth map, we use the per-pixel average angles calculated from the training set.

1) KITTI: We used the KITTI odometry [1] dataset, which is a benchmark for odometry evaluation. The dataset provides 22 trajectories of LiDAR scans measured by the Velodyne HDL-64E[2] scanner, where each scan has 64 layers vertically. It includes 19,310 scans for training, 4,071 scans for validation, and 20,351 scans for testing. The provided data are processed sequences of Cartesian coordinate points ordered by azimuth and elevation angles. Most studies applied spherical projection to create a depth map from the point sets. However, this approach produces sampling artifacts due to the scanner's nonlinear vertical spacing[3]. As in Caccia *et al.* [11], we first chunk the ordered sequence into 64 sub-sequences, where each represents one elevation angle. We then subsample 256 points for each sub-sequence and stack them to form a 64×256 inverse depth map. Unlike [11], we do not clip the vertical angle and the distance range and do not fill in the point-drops with adjacent pixels[4].

2) MPO: We also use the Multimodal Panoramic 3D Outdoor dataset (MPO) [13]. In particular, we use a "sparse" dataset of MPO, which includes a total of 34,200 LiDAR scans from 60 trajectories. The dataset has diversity in geometry and point-drop distribution because it was constructed to classify six different outdoor scenes: coast, forest, indoor parking, outdoor parking, residential area, and urban area. Each scan is a sequence of Cartesian coordinate points with vertical angle IDs, obtained using Velodyne HDL-32E[5] scanner. In this study, we split the trajectories into 20,322 scans for training, 3,787 scans for validation, and 10,091 scans for testing. We chunk the sequence according to the vertical angle ID and create a 32×256 inverse depth map similar to the KITTI procedure.

TABLE I

Baseline architecture. H and W are the height and width of the inverse depth map, respectively. †The shape is replaced with $2 \times H \times W$ for DUSty-I and $3 \times H \times W$ for DUSty-II.

	Layer	Kernel size	Nonlinearity	Output shape
Generator G_θ	(Input)	–	–	$512 \times 1 \times 1$
	Transposed Conv.	$H/16 \times W/16$	Leaky ReLU	$512 \times H/16 \times W/16$
	Transposed Conv.	4×4	Leaky ReLU	$256 \times H/8 \times W/8$
	Transposed Conv.	4×4	Leaky ReLU	$128 \times H/4 \times W/4$
	Transposed Conv.	4×4	Leaky ReLU	$64 \times H/2 \times W/2$
	Transposed Conv.	4×4	Tanh	$1 \times H \times W^\dagger$
Discriminator D_ϕ	(Input)	–	–	$1 \times H \times W$
	Blur filtering [15]	3×3	–	$2 \times H \times W$
	Convolution	4×4	Leaky ReLU	$64 \times H/2 \times W/2$
	Convolution	4×4	Leaky ReLU	$128 \times H/4 \times W/4$
	Convolution	4×4	Leaky ReLU	$256 \times H/8 \times W/8$
	Convolution	4×4	Leaky ReLU	$512 \times H/16 \times W/16$
	Convolution	$H/16 \times W/16$	–	$1 \times 1 \times 1$

B. Models

1) Baseline: Table I shows the network architectures of our baseline GAN. The architecture design is inspired by the existing work [11]. However, we modify the output shape for our task, remove all normalization layers, and only use the Leaky ReLU nonlinearity with a negative slope of 0.2. We add vertical/horizontal blur filtering [15] as the first layer of the discriminator, to mitigate the learning difficulty of a discrete distribution. As in existing studies [20, 21], we replace the zero-padding of all convolutional layers with horizontal circular padding, where the right and left boundaries are padded with the pixels on the opposite sides. It enables the model to spread the receptive fields out of the boundaries so that it can process the cylindrical tensors.

2) DUSty-I (ours): For a fair comparison, in this method, we only modify the last convolutional layer of the baseline

[2] https://velodynelidar.com/products/hdl-64e/

[3] The HDL-64E scanner has several versions with different vertical spacing, but it was not unified in studies with KITTI.

[4] https://github.com/pclucas14/lidar_generation

[5] https://velodynelidar.com/products/hdl-32e/

generator. The last layer produces 2-channel outputs for inverse depth and measurability. The tanh nonlinearity is applied to only the inverse depth output. The measurability output is transformed into a binary mask with pixel-level Gumbel sampling, as introduced in Section III-B.

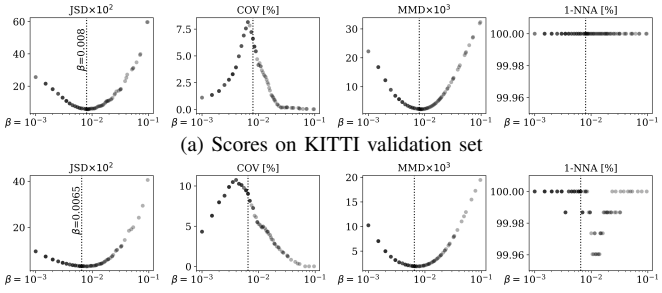
3) DUSty-II (ours): In this method, we add an extra channel to the measurability output of the DUSty-I model to decompose the Gumbel sampling into pixel-level and image-level, as introduced in Section III-C. The same temperature τ is used for both pixel-level and image-level Gumbel sampling.

C. Implementation Details

We set the maximum distance to the point-drop constant α in Eq. 4. Moreover, we observed that a low value of temperature τ in Eq. 5 slowed the convergence despite a better approximation of the binary masks. This study uses a fixed $\tau = 1$ in all experiments. For Gumbel sampling from measurability $\boldsymbol{\pi}$ in Eq. 5, our network directly outputs the logit $\boldsymbol{e}$, instead of $\boldsymbol{\pi} = \mathrm{sigmoid}(\boldsymbol{e})$. For adversarial training, we use the non-saturating loss in Eq. 1 and 2, with a R_1 gradient penalty [3]. The penalty coefficient was set to 1. All parameters were updated by Adam [22] optimizer for 25M iterations with a learning rate of 0.002 and a batch size of 32. We apply the equalized learning rate [4] for all trainable layers, where the parameters are initialized with $\mathcal{N}(0, 1)$ and scaled by He's initialization constant at runtime. Moreover, we apply DiffAugment by Zhao *et al.* [23], which composed of *color*, *translation*, and *cutout* augmentations to the discriminator inputs. For the *translation* augmentation, we circulate the inputs horizontally. DiffAugment was not required for training stability but greatly improved the quality of inverse depth maps. We take the exponential moving average for generator parameters θ. We implemented our networks in PyTorch and performed distributed training on two NVIDIA Titan RTX GPUs. The training required approximately 22 hours for each model. The code will be available at `https://github.com/kazuto1011/dusty-gan`.

D. Evaluation Metrics

We measured four types of distributional similarities between the sets of reference and generated point clouds [5]: Jensen–Shannon divergence (JSD) for quality, coverage (COV) for diversity, minimum matching distance (MMD) for quality, and 1-nearest neighbor accuracy (1-NNA) for both quality and diversity evaluation. We found that the synthesis evaluation on LiDAR point clouds had an extremely high cost, particularly for calculating the distance matrix for the sets of point clouds in COV, MMD, and 1-NNA. For efficiency, we first subsampled the series of test data to be 5,000 data in total, and for each, we also randomly subsampled 2,048 points from the full $H \times W$ points by farthest point sampling. We generated 5,000 data from each model and reduced the number of points in the same manner. We used the Chamfer distance to measure the pairwise similarity for the distance matrix of point clouds. Additionally, we computed the sliced Wasserstein distance (SWD) [4] to measure the patch-based image similarity for the inverse depth maps. For all metrics, we report the mean scores with the standard deviation over five runs with different latent codes.



(a) Scores on KITTI validation set

(b) Scores on MPO validation set

Fig. 4. Impact of relative tolerance β for the baseline method. We optimized β based on the weighted score and set 0.008 for KITTI and 0.0065 for MPO (dotted vertical lines).

E. Quantitative Results

It is non-trivial for the baseline model to produce values that exactly match the point-drop constant α. Therefore, we first optimize a relative tolerance β to decide if an interest pixel $\boldsymbol{x}_z^{i,j}$ is α based on

$$\frac{\|\boldsymbol{x}_z^{i,j} - \alpha\|}{x_{\max} - x_{\min}} \leq \beta, \tag{7}$$

where $x_{\max} - x_{\min}$ is a range of the inverse depth. β is optimized by HyperOpt [24] within $[10^{-3}, 10^{-1}]$ for 100 steps to minimize the following weighted 3D score, $\mathrm{JSD} \times 10 - \mathrm{COV} + \mathrm{MMD} \times 10^2 + 1\text{-NNA}$, on the validation set for each dataset. For efficiency, we reduce the number of points to 512. Fig. 4 shows trade-off results because the point-drop tolerance erodes the available range of inverse depth. We employ $\beta = 0.008$ for KITTI and $\beta = 0.0065$ for MPO. We note that our noise-aware model DUSty *does not* require the tolerance, *i.e.*, $\beta = 0$.

In Table II, we compare the synthesis performance among the baseline and our methods. On both KITTI [1] and MPO [13] datasets, our methods outperformed the baseline on all the 3D metrics. On the SWD score, the baseline was better than DUSty-I on KITTI, while our methods outperformed in the other cases. We consider that our sampling-based binary mask still has an "appearance" gap with the truth distribution of point-drops compared with the direct modeling by the baseline. Meanwhile, the 3D results on four metrics indicate that our decomposition approach has a positive effect on the disentangled modeling of inverse depth and point-drop.

F. Qualitative Results

Fig. 5 compares inverse depth maps generated from the baseline and our DUSty-I model on the KITTI dataset. Our method succeeds in representing the sharp jump edges around point-drops. However, the baseline has interpolated pixels that can be unexpected noise in point clouds. Moreover, we show some synthetic examples of all methods for

TABLE II

QUANTITATIVE COMPARISON OF SYNTHESIS PERFORMANCE. ↓: THE LOWER THE BETTER. ↑: THE HIGHER THE BETTER.

Dataset	Method	JSD $\times 10^2$ ↓ (3D quality)	COV [%] ↑ (3D diversity)	MMD $\times 10^3$ ↓ (3D quality)	1-NNA [%] ↓ (3D quality/diversity)	SWD ↓ (2D quality)
KITTI	Baseline	6.45 ± 0.06	4.99 ± 0.04	2.36 ± 0.03	99.99 ± 0.00	0.158 ± 0.011
	DUSty-I (**ours**)	**2.85** ± 0.01	**38.08** ± 0.55	1.14 ± 0.01	**93.69** ± 0.19	0.167 ± 0.010
	DUSty-II (**ours**)	3.54 ± 0.07	38.05 ± 0.65	**1.12** ± 0.01	94.62 ± 0.30	**0.151** ± 0.011
	Training set	0.93	35.02	0.87	96.72	0.182
MPO	Baseline	2.53 ± 0.03	6.10 ± 0.12	2.03 ± 0.01	99.31 ± 0.03	0.180 ± 0.013
	DUSty-I (**ours**)	**1.47** ± 0.02	22.85 ± 0.34	**1.53** ± 0.01	95.03 ± 0.16	0.174 ± 0.014
	DUSty-II (**ours**)	1.71 ± 0.02	**30.94** ± 0.34	1.54 ± 0.01	**94.84** ± 0.19	**0.160** ± 0.018
	Training set	0.74	34.88	1.52	87.09	0.158

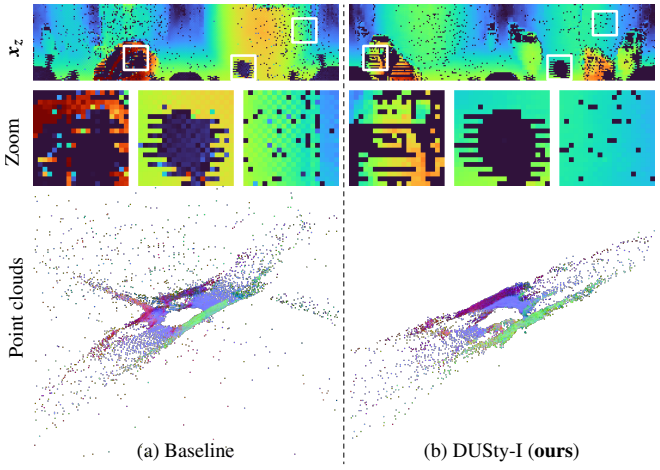


Fig. 5. Qualitative comparison of inverse depth outputs $\boldsymbol{x}_z$, zoomed-in regions (white boxes in $\boldsymbol{x}_z$), and points clouds, with and without the proposed measurability learning. Best viewed in color.

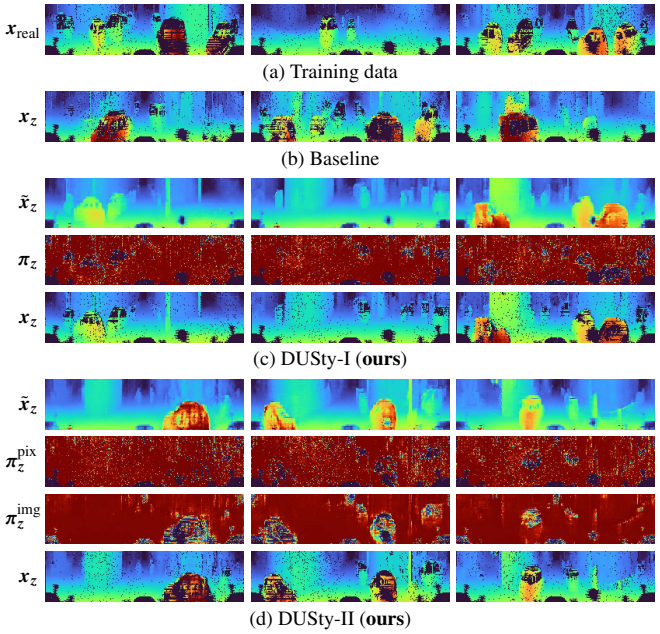


Fig. 6. Qualitative comparison on the KITTI [1] dataset. Each column is from a unique latent code. $\boldsymbol{x}_{\text{real}}$ and $\boldsymbol{x}_z$ denotes real LiDAR data and generated LiDAR data, respectively. $\tilde{\boldsymbol{x}}_z$ and $\boldsymbol{\pi}_z$ denotes the inverse depth and measurability outputs from our proposed models, respectively.

KITTI in Fig. 6 and MPO in Fig. 7. We can see that our methods DUSty-I and DUSty-II successfully learned a complete inverse depth maps in $\tilde{\boldsymbol{x}}_z$ and drop uncertainty in $\boldsymbol{\pi}_z$. Regarding $\boldsymbol{\pi}_z$, the results show that the sampling variation would occur on the boundaries of vehicle-like objects and the scattered noises. In contrast, the steady point-drops are represented as sufficiently low measurability, such as the ego-vehicle shadows in KITTI and the sky regions in MPO. Furthermore, DUSty-II separately learned the pixel correlation of the drop uncertainty in $\boldsymbol{\pi}_z^{\text{img}}$.

V. RECONSTRUCTION BY LATENT SPACE EXPLORATION

With trained GANs, we can reproduce the given data by optimizing a latent code [3, 25], where the task is called *GAN inversion*. In this section, we optimize the latent code $\boldsymbol{z}$ to reconstruct unseen LiDAR data from KITTI and MPO test sets. We demonstrate our decomposition approach is easier to find a matching latent code.

A. Objective

To reproduce a given inverse depth $\boldsymbol{x}_{\text{target}}$, we optimize the latent code $\boldsymbol{z}$ by minimizing the L1 distance between $\boldsymbol{x}_{\text{target}}$ and generated inverse depth $\tilde{\boldsymbol{x}}_z$ before masking:

$$\hat{z} = \arg\min_z \frac{\sum_{i,j} \mathbb{1}[\boldsymbol{x}_{\text{target}} \neq \alpha]^{i,j} \|\boldsymbol{x}_{\text{target}}^{i,j} - \tilde{\boldsymbol{x}}_z^{i,j}\|_1}{\sum_{i,j} \mathbb{1}[\boldsymbol{x}_{\text{target}} \neq \alpha]^{i,j}}, \quad (8)$$

where $\mathbb{1}[\boldsymbol{x}_{\text{target}} \neq \alpha] \in \{0,1\}^{H \times W}$ is a point-drop indicator of the target data $\boldsymbol{x}_{\text{target}}$, and (i, j) is the 2D location of the inverse depth map. The latent $\boldsymbol{z} \in \mathbb{R}^d$ is updated for 1,000 iterations by the Adam optimizer with a learning rate of 0.1. Considering the nature of high-dimensional Gaussian priors, the search space of the latent $\boldsymbol{z}$ is constrained to a surface of a hypersphere with a radius of $\sqrt{d}$ [25]. To avoid sticking in local minima, we add a Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.05t^2\boldsymbol{I})$, where t goes from 1 to 0 in iterations [3]. The optimization required approximately 4 seconds for each sample.

B. Quantitative Results

To assess the reconstruction performance, we compute standard metrics in depth estimation [1]: relative absolute

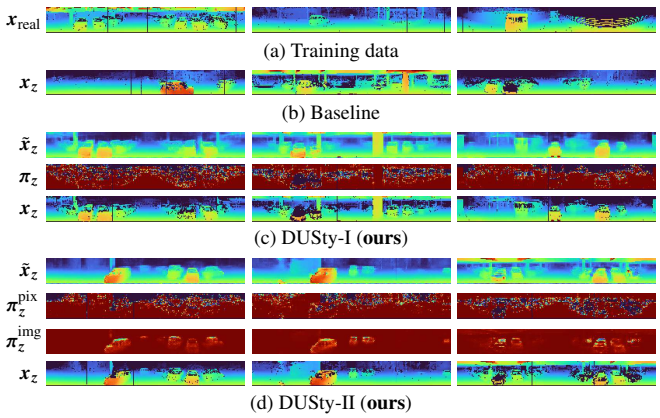


Fig. 7. Qualitative comparison on the MPO [13] dataset. Each column is from a unique latent code.

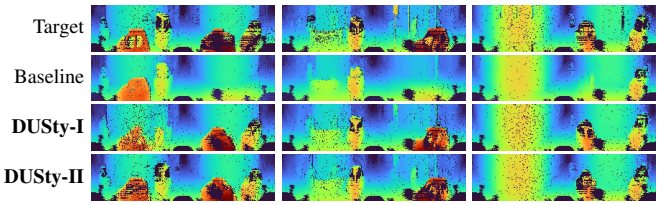


Fig. 8. Qualitative comparison of the reconstructed inverse depth maps. Our method is better at reproducing the geometric features of the given targets and generating realistic point-drops.

error (Abs Rel), relative squared error (Sq Rel), root mean squared error (RMSE), RMSE on logarithmic depth (RMSE log), and the ratio of pixels with relative error δ under the threshold ($\delta < 1.25^i, i = 1, 2, 3$). Moreover, we also compute the Chamfer distance (CD) between point clouds from $\boldsymbol{x}_{\text{target}}$ and $\boldsymbol{x}_{\hat{z}}$. Table III shows the results of the baseline and our proposed methods. Our method outperformed the baseline on all the metrics, and DUSty-I was slightly better than DUSty-II in most cases.

C. Qualitative Results

Fig. 8 compares the reconstructed inverse depth maps $\boldsymbol{x}_{\hat{z}}$ from the baseline and our methods. The baseline failed to reconstruct some foreground objects ($\hat{z} \leftarrow \boldsymbol{x}_{\text{target}}$), contrary to the synthesis results in Fig. 5 and 6 ($z \rightarrow \boldsymbol{x}_z$). In contrast, our method reproduced the details and also synthesized more realistic point-drops. This indicates that our decomposition trick yields a better embedding of the geometric styles of LiDAR scans.

D. Applications

In Fig. 9, we show some examples from DUSty-I, where the targets are with three types of strong corruptions: 90% of points are randomly dropped (Fig. 9(b)), 8 out of 64 horizontal lines are observable (Fig. 9(c)), and pixel-wise noise from $\mathcal{N}(0, 0.01)$ are added to the depth values (Fig. 9(d)). Despite very sparse or noisy targets $\boldsymbol{x}_{\text{target}}$, our method succeeded in reconstructing inverse depth maps with underlying complete surfaces $\tilde{\boldsymbol{x}}_{\hat{z}}$ and rendering realistic point-drops. On the other hand, we can also observe that the far objects and thin structures are still difficult to reconstruct in some cases.

VI. Conclusion

In this paper, we proposed DUSty, a GAN framework for LiDAR scan synthesis based on the decomposed image representation of inverse depth and point-drop. Our method showed effectiveness on both synthesis and reconstruction tasks compared to the baseline that directly models the LiDAR scans. The reconstruction experiments indicated potential applications of our method, such as restoration of corrupted data and upsampling of sparse scans from low-cost LiDARs. Our method would also be applicable to rendering realistic point-drops for fully measurable simulation data [9, 12] since our pixel-wise depth reconstruction can simultaneously produce a measurability map. Although this study used standard GAN architectures, further investigation on the network design could improve the generation quality. Future work includes introducing our point-drop learning into the other large networks [3, 4] and applying our synthetic data to perception tasks.

Acknowledgment

This work was supported by a Grant-in-Aid for JSPS Fellows Grant Number JP19J12159 and JSPS KAKENHI Grant Number JP20H00230.

References

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research (IJRR)*, vol. 32, no. 11, pp. 1231–1237, 2013.

[2] 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 (NIPS)*, 2014, pp. 2672–2680.

[3] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8110–8119.

[4] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[5] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4541–4550.

[6] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018, pp. 40–49.

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," Tech. Rep., Dec. 2015.

[8] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," *arXiv preprint arXiv:1608.07916*, 2016.

[9] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1887–1893.

[10] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, "Polarnet: An improved grid representation for online lidar point clouds semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9601–9610.

TABLE III

QUANTITATIVE COMPARISON OF RECONSTRUCTION PERFORMANCE. ↓: THE LOWER THE BETTER. ↑: THE HIGHER THE BETTER.

		3D error ↓	2D error ↓				2D accuracy [%] ↑		
Dataset	Method	CD $\times 10^4$	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
KITTI	Baseline	5.31	0.199	12.394	9.667	0.280	89.11	95.21	97.51
	DUSty-I (**ours**)	**1.64**	**0.074**	0.555	3.722	**0.155**	**92.09**	**96.94**	**98.69**
	DUSty-II (**ours**)	1.66	0.077	**0.546**	**3.663**	0.157	91.91	96.93	**98.69**
MPO	Baseline	9.33	0.197	9.008	10.523	0.308	84.15	92.26	95.61
	DUSty-I (**ours**)	**1.25**	**0.109**	**0.878**	**4.930**	**0.229**	**85.90**	**94.02**	**97.11**
	DUSty-II (**ours**)	1.28	0.111	1.013	5.079	0.232	85.83	93.89	97.00

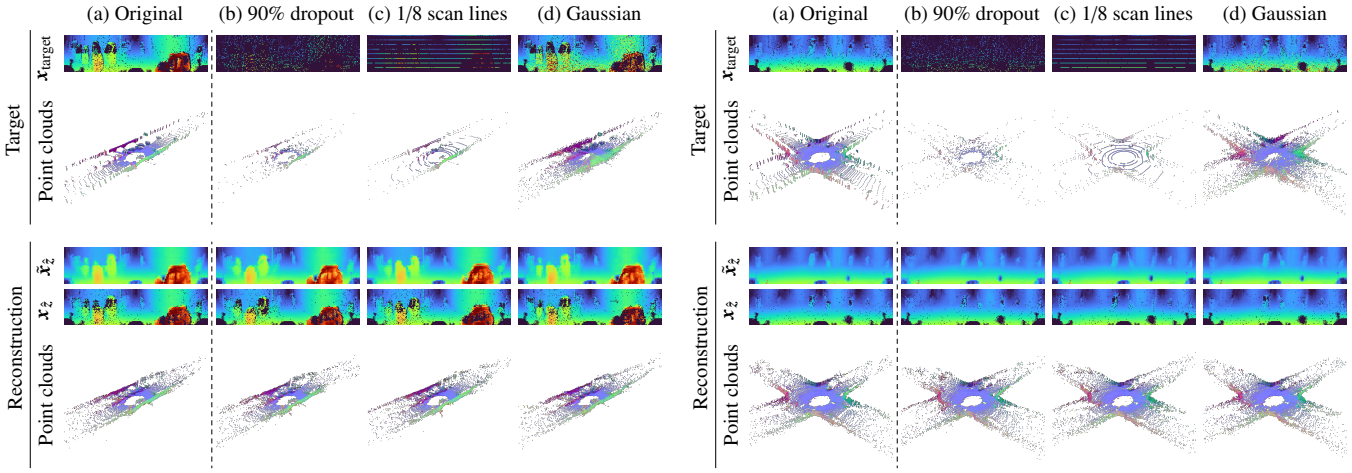


Fig. 9. Reconstruction examples for the target LiDAR scans $\boldsymbol{x}_{\text{target}}$ with three types of corruption. All results were generated by our DUSty-I trained on KITTI. (a) Sample without any corruption from the KITTI test set. (b) 90% points are randomly dropped. (c) Only 8 out of 64 horizontal lines are observable. (d) An additive Gaussian noise from $\mathcal{N}(0, 0.01)$ is applied to a depth value.

[11] L. Caccia, H. van Hoof, A. Courville, and J. Pineau, "Deep generative modeling of lidar data," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5034–5040.

[12] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma, and R. Urtasun, "Lidarsim: Realistic lidar simulation by leveraging the real world," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 167–11 176.

[13] O. Martinez Mozos, K. Nakashima, H. Jung, Y. Iwashita, and R. Kurazume, "Fukuoka datasets for place categorization," *The International Journal of Robotics Research (IJRR)*, vol. 38, no. 5, pp. 507–517, 2019.

[14] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2Noise: Learning image restoration without clean data," ser. Proceedings of Machine Learning Research, vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 2965–2974.

[15] T. Kaneko and T. Harada, "Noise robust generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8404–8414.

[16] A. Bora, E. Price, and A. G. Dimakis, "AmbientGAN: Generative models from lossy measurements," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[17] S. C.-X. Li, B. Jiang, and B. Marlin, "Misgan: Learning from incomplete data with generative adversarial networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[18] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[19] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[20] S. Schubert, P. Neubert, J. Pöschmann, and P. Protzel, "Circular convolutional neural networks for panoramic images and laser data," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 653–660.

[21] K. Nakashima, H. Jung, Y. Oto, Y. Iwashita, R. Kurazume, and O. M. Mozos, "Learning geometric and photometric features from panoramic lidar scans for outdoor place categorization," *Advanced Robotics*, vol. 32, no. 14, pp. 750–765, 2018.

[22] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[23] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han, "Differentiable augmentation for data-efficient gan training," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[24] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2013, pp. 115–123.

[25] S. Menon, A. Damian, S. Hu, N. Ravi, and C. Rudin, "PULSE: Self-supervised photo upsampling via latent space exploration of generative models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2437–2445.