Transfer Your Perspective: Controllable 3D Generation from Any Viewpoint in a Driving Scene

Tai-Yu Pan[1], Sooyoung Jeon[1], Mengdi Fan[1], Jinsu Yoo[1], Zhenyang Feng[1], Mark Campbell[2], Kilian Q. Weinberger[2], Bharath Hariharan[2], Wei-Lun Chao[1]

[1]The Ohio State University, [2]Cornell University

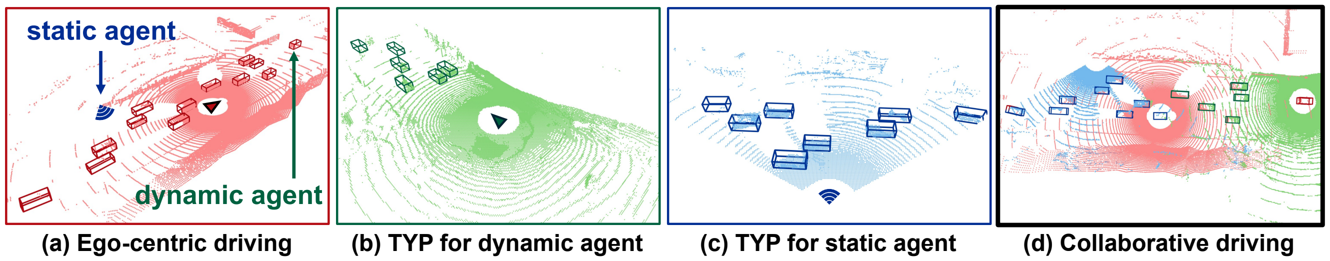

Figure 1. **Illustration of the proposed problem and solution, Transfer Your Perspective (TYP).** (a) A given sensory data captured by the ego-car (red triangle). (b) A generated sensory data by TYP, seeing from the viewpoint of another vehicle (green triangle) in the same scene. (c) A generated sensory data, seeing from an imaginary static agent like roadside units (blue icon). (d) Putting all the sensory data together, given or generated, TYP enables the development of collaborative perception with little or no real collaborative driving data.

Abstract

Self-driving cars relying solely on ego-centric perception face limitations in sensing, often failing to detect occluded, faraway objects. Collaborative autonomous driving (CAV) seems like a promising direction, but collecting data for development is non-trivial. It requires placing multiple sensor-equipped agents in a real-world driving scene, simultaneously! As such, existing datasets are limited in locations and agents. We introduce a novel surrogate to the rescue, which is to generate realistic perception from different viewpoints in a driving scene, conditioned on a real-world sample—the ego-car's sensory data. This surrogate has huge potential: it could potentially turn any ego-car dataset into a collaborative driving one to scale up the development of CAV. We present the very first solution, using a combination of simulated collaborative data and real ego-car data. Our method ***Transfer Your Perspective*** (**TYP**) *learns a conditioned diffusion model whose output samples are not only realistic but also consistent in both semantics and layouts with the given ego-car data. Empirical results demonstrate* TYP*'s effectiveness in aiding in a CAV setting. In particular,* TYP *enables us to (pre-)train collaborative perception algorithms like early and late fusion with little or no real-world collaborative data, greatly facilitating downstream CAV applications.*

1. Introduction

Seeing the world from an ego-centric perspective, a self-driving car risks being "narrow-sighted," limiting its ability to respond appropriately in dynamic driving environments. For instance, it should slow down or honk if a pedestrian is about to cross the road or if another vehicle is merging into its lane. However, these actions depend on the car's ability to detect traffic participants, which may be occluded by a large bus or a building at a sharp intersection. This occlusion problem can hardly be addressed by mounting a few more sensors on the car. We argue that a self-driving car must move beyond its ego-centric perspective.

One intuitive way is to collaborate with nearby "agents" that observe the scene from different angles, such as other sensor-equipped cars or static devices like roadside units (RSUs). When trained together to address GPS errors and synchronization delays [55], collaborative perception has been shown to significantly improve the accuracy of each agent's perception, particularly in detecting occluded or distant objects [3, 4, 14, 19, 23, 33, 42, 45, 50–52]. *However, collecting training data for collaborative perception is never easy.* Unlike single-agent data, which can be collected by simply driving a car on the road, collaborative data requires the simultaneous presence of multiple agents in the same driving scene. For dynamic agents like vehicles, precise coordination is needed to ensure they are within communication range. These challenges limit existing works in scale and the number of agents (typically just two). While one may leverage simulated data by game engines [7], they often fail to capture the diversity of real-world scenes. We thus ask:

Can we obtain realistic data for learning collaborative perception with much less effort, ideally as easily as single-agent data?

Specifically, given the vast amount of single-agent LiDAR data already collected across various driving environments, is it possible to convert each of them into collaborative perception data by generating additional point clouds from alternative reference viewpoints within the same scene?

At first glance, **this question may seem overly ambitious for three reasons.** First, for collaborative perception to be effective, the generated point cloud must provide information that the ego-car's point cloud cannot capture, such as occluded surfaces invisible to the ego-car. This creates a chicken-and-egg problem: if the ego-car's point cloud is all we initially know about the scene, how can we deduce any additional information? Second, for training purposes, the generated data must be realistic. It should replicate the constraints and data patterns of a real sensor as if it were positioned at the reference viewpoint, producing no points in occluded areas and fewer points in distant areas. Last but not least, in regions commonly perceivable from both the reference and ego-car's viewpoints, the generated data should align in layouts and semantics with the ego-car's data. On the surface, this may seem as simple as copying the ego-car's data, but doing so would violate the second requirement. In essence, the generated data should resemble a point cloud seen from the reference viewpoint, not the ego-car.

That said, after a deeper look at the question, **we argue that it is achievable with three key insights.** First, given semantic information around the ego-car for conditioning, existing research has shown promising progress in generating realistic point clouds [13, 27, 32, 44, 49, 62, 63]. This is especially encouraging, as most existing single-agent datasets provide 3D object labels, and they can be translated to obtain semantic information centered around the reference position. This further implies that we can generate occluded object surfaces as seen from the ego-car if they are visible from the reference viewpoint. Second, semantic information is easily editable: even if the translated object map has noticeable empty areas due to limited ego-car perception, object boxes can be manually added to make the map appear more realistic from the reference viewpoint. Lastly, in commonly perceivable regions where the generated data needs to meet two physical constraints, we could leverage simulators based on computational graphics and optics. Specifically, we can use simulated data from the two views to train a conditioned generative model that maps one viewpoint to the other.

Building upon these insights, we propose **Transfer Your Perspective (TYP)**, a novel research problem and the very first solution for *generating a realistic point cloud from any viewpoint in a scene, given real ego-car's point cloud and semantic labels as conditions.* TYP assumes access to **1)** a simulated collaborative driving dataset with multiple agents perceiving the same scenes from different viewpoints and **2)** a real single-agent driving dataset; both are labeled. Our solution involves a **conditioned latent diffusion model** [34] and a **dedicated two-stage training process**. In the first stage, we consider a single-agent scenario and train the model using real data conditioning only on object locations. This equips the model with the ability to generate diverse and realistic scenes. We denote the learned model by $P(\boldsymbol{x}|\boldsymbol{y})$, where $\boldsymbol{y}$ stands for the semantic condition and $\boldsymbol{x}$ for the point cloud. In the second stage, we incorporate the simulated data to learn *how to ground generation on data from another agent's viewpoint* so that the model can produce a semantically consistent reference point cloud given the ego-car's data. We learn a lightweight conditioning module to turn $P(\boldsymbol{x}|\boldsymbol{y})$ into $P(\boldsymbol{x}_r|\boldsymbol{x}_e, \boldsymbol{y}_r)$, where e and r indicate ego and reference views, respectively. We note that $\boldsymbol{x}_e$ was pre-translated to center around the reference position.

One challenge in TYP is the domain gap between real and simulated data. Due to differences in sensor configurations and placements, data collection environments, and the sim-to-real gap, the two sets of point clouds inevitably exhibit discrepant distributions, patterns, and densities. To address this, *we insert a domain adaptation step between the two training stages*. We train a separate encoder-decoder for the simulated data while enforcing a constraint to make the encoded features of simulated and real data indistinguishable [40]. This step allows us to learn $P(\boldsymbol{x}_r|\boldsymbol{x}_e, \boldsymbol{y}_r)$ in a space with reduced domain discrepancy during the second stage.

Once trained, we pair $P(\boldsymbol{x}_r|\boldsymbol{x}_e, \boldsymbol{y}_r)$ with the real data's encoder-decoder to generate real-style point clouds grounding on real ego-car's data, so that *we can develop collaborative perception algorithms without real collaborative data.* More specifically, given a real ego-car's perception $\boldsymbol{x}_e$ and label $\boldsymbol{y}_e$, we first translate them to center around the reference position, followed by optionally injecting object labels into $\boldsymbol{y}_e$ to make it realistic from the reference viewpoint.

We extensively validate TYP on multiple datasets, all in an *offline* setting. Empirical results demonstrate TYP's effectiveness in generating high-quality reference data to aid in the development of collaborative perception. In particular, we show that a conditioned diffusion model trained solely on simulated data (*e.g.*, OPV2V [51]) can already turn a real single-agent dataset into a real-alike collaborative one. As such, one can train collaborative perception algorithms for real test data without real training data. We further generate the "ColWaymo" dataset by training TYP on real single-agent Waymo data [39] and simulated OPV2V data [51], thus turning the former into a collaborative one. Collaborative perception backbones pre-trained on the large-scale, semisynthetic ColWaymo data demonstrate a remarkable transferability. They notably improve collaborative perception developed in the downstream tasks (*e.g.*, V2V4Real [52]), even in a many-shot setting.

In sum, our key contributions are:

- We propose a new research direction to aid in the development of collaborative perception, *generating real-looking sensory data from potentially any viewpoint in a real driving scene*, capable of turning a single-agent dataset into a collaborative one. This has the potential to scale up collaborative autonomous driving (CAV).
- We present TYP, a two-stage training recipe with domain adaptation in between to learn the generative model using simulated collaborative data and real single-agent data.
- Extensive experiments demonstrate TYP's effectiveness in aiding in collaborative perception across various scenarios.

2. Related Work

Collaborative autonomous driving (CAV) offers significant benefits, including extended perception range by sharing sensor information to detect objects beyond a single vehicle's field of view. Despite these benefits, collecting large-scale, real-world datasets for CAV poses significant challenges due to the complexity and cost of deploying multiple instrumented vehicles in diverse environments. Existing datasets often have limitations that restrict the scope of CAV research. For instance, OPV2V [51] and V2X-Sim [20] rely on simulations using the CARLA simulator [7], allowing controlled scenarios with multiple vehicle agents, yet lacking real-world variability. V2V4Real [52], on the other hand, provides real-world data, yet it only contains two collaborating agents, which restricts the exploration of more complex multi-agent interactions. Similarly, the DAIR-V2X [57] dataset also offers real-world data including vehicle-to-infrastructure (V2I) and V2V, but it mainly focuses on V2I scenarios and still has a limited number of collaborating vehicles. To overcome current limitations, we propose a new research direction—generating realistic point clouds given the ego's perspective.

3D generation. Various works have explored 3D scene generation, such as LiDARGen [62], R2DM [27], LiDM [32], UltraLiDAR [48], and RangeLDM [13]. Additionally, LidarDM [63] and Text2LiDAR [44] investigate conditional scene generation, with the former relying on hand-crafted map layouts and the latter conditioning on text inputs. However, all these methods focus on ego-centric generation. We propose a new research direction for CAV, aiming to generate realistic and consistent scenes conditioned on the ego agent's real point clouds—an area that remains largely unexplored.

Diffusion models. Diffusion models have recently advanced generative modeling for high-quality LiDAR point clouds and images. Denoising Diffusion Probabilistic Models (DDPMs) [12] outperform traditional Generative Adversarial Networks (GANs) [10] and further enhance efficiency

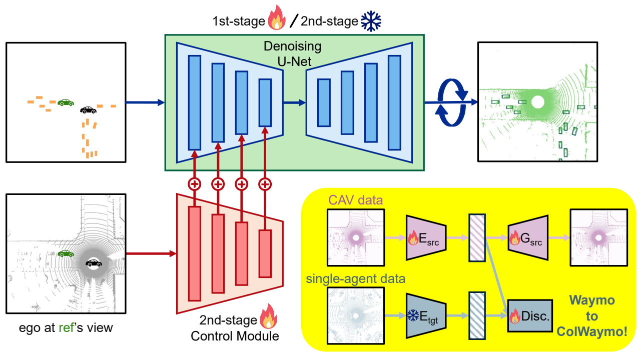

Figure 2. **Illustration of TYP's conditioned generative model and training process.** We propose a two-stage training procedure. The first stage maximizes the generation capability by conditioning solely on object locations (using real single-agent target data), while the second stage grounds the generation on the ego-car's perspective to match semantics and layouts (using simulated CAV data). Additionally, we introduce a discriminator to adapt simulated CAV features to the real target domain, making the trained model readily applicable to the target domain after the second stage.

for LiDAR generation. In autonomous driving applications, methods like RangeLDM [13], LidarDM [63], and LiDARGen [62] apply diffusion models for realistic LiDAR scene generation. In this paper, we utilize its generation capability to study our proposed problem.

3. Transfer Your Perspective (TYP)

In this paper, we introduce a new research direction to advance collaborative autonomous driving (CAV): *generating LiDAR point clouds from different perspectives within the same scene as the ego agent*, aiming to reduce the tedious efforts of collecting data for CAV. We begin by defining the proposed problem in Sec. 3.1. In Sec. 3.2, we discuss the representations of the inputs, including point clouds and semantic information. Sec. 3.3 outlines the pipeline developed to address this problem. Finally, in Sec. 3.4, we demonstrate how this capability can be applied to datasets that only have labeled ego agents, *e.g.*, Waymo Open Dataset (WOD) [39].

3.1. Problem Setup

Given the perception data $\boldsymbol{x}_e$ from an **e**go agent, we aim to build a model $P(\boldsymbol{x}_r|\boldsymbol{x}_e)$ to generate new perception data $\boldsymbol{x}_r$ seen from a different **r**eference location and perspective within a communication range. Here, $\boldsymbol{x}$ represents a LiDAR point cloud. This task presents several challenges, including potential information gaps between the two views, the need for alignment within the commonly perceivable area, and ensuring realism in the generated data, as discussed in Sec. 1.

To address these challenges, we extend the original problem by incorporating semantic information, such as object bounding boxes, represented as $P(\boldsymbol{x}_r|\boldsymbol{x}_e, \boldsymbol{y}_r)$. We assume the availability of this information around the ego agent, denoted by $\boldsymbol{y}_e$. For example, most of the existing single-agent datasets provide object labels. This semantic information can be easily translated and edited to become $\boldsymbol{y}_r$ by tools like traffic re-players [11], making it a key to bridge the information gap between the ego and reference views, enabling the reference agent to "see" those objects and surfaces beyond the ego's view—a core concept in CAV. Additionally, by aligning generated point clouds with object locations, this extension facilitates our goal of scaling up CAV development. In essence, the generated $(\boldsymbol{x}_r, \boldsymbol{y}_r)$ pairs can be used flexibly alongside $(\boldsymbol{x}_e, \boldsymbol{y}_e)$ in various CAV applications, like directly using them to train collaborative perception algorithms.

In the following sections, we describe how $\boldsymbol{x}$ and $\boldsymbol{y}$ are encoded, followed by our approach to tackling the problem.

3.2. Representations and Embeddings

LiDAR. There are multiple ways to represent point clouds in continuous 3D space, such as coordinates (*i.e.*, x, y, and z) [30, 31, 37], range images [25, 30, 43], and voxelization [17, 24, 53, 59]. To better align with spatial control from object locations, we follow [49] to voxelize point clouds using a pre-defined grid and record voxel occupancy. [49] also highlights that this representation can naturally handle varying point densities and only minimally impacts LiDAR generation, with some precision trade-offs during voxelization. In short, we represent a point cloud by $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$.

To avoid the computational cost of 3D convolutions, we convert $\boldsymbol{x}$ into Bird's-Eye-View (BEV) images by treating the height dimension (*i.e.*, C) as feature channels for 2D convolutions. This approach has been widely adopted in self-driving perception [5, 48, 58], allowing the use of 2D image-based model architectures and algorithms.

Semantic information. Given that the point clouds are represented as BEV images, it is also intuitive to represent object locations in BEV. We create binary object maps by considering only the x and y coordinates of 3D bounding boxes, resulting in $\boldsymbol{y} \in \mathbb{R}^{H \times W \times 1}$. (We could further extend it by including category information.)

Feature embedding. Following the literature [34], we encode input tensors using a Vector Quantized-Variational Autoencoder (VQ-VAE) [41], comprising an encoder E, a quantization function Q for feature vectors on spatial grids, and a decoder G. Formally, we obtain $\boldsymbol{x}^f = E_x(\boldsymbol{x}) \in \mathbb{R}^{h \times w \times c_x}$ by the encoder, map each c_x-dimensional vector to a learnable code to obtain $\boldsymbol{x}^z = Q_x(\boldsymbol{x}^f) \in \mathbb{R}^{h \times w \times c_x}$, and generate outputs using the decoder $\hat{\boldsymbol{x}} = G_x(\boldsymbol{x}^z) \in \mathbb{R}^{H \times W \times C}$, where h and w are spatial resolutions of the feature map downsampled from H and W, and c_x is the number of channels.

The VQ-VAE model is trained end-to-end by minimizing:

$$\mathcal{L}_{\text{vq}} = \mathcal{L}_{\text{rec}} + \|\text{sg}[E_x(\boldsymbol{x})] - \boldsymbol{x}^z\|_2^2 + \|\text{sg}[\boldsymbol{x}^z] - E_x(\boldsymbol{x})\|_2^2, \quad (1)$$

where sg$[\cdot]$ and $\mathcal{L}_{\text{rec}}$ denote the stop-gradient operation and the reconstruction loss, respectively. As our point cloud representation is binary occupancy $\boldsymbol{x} \in \{0, 1\}^{H \times W \times C}$, binary cross-entropy is a natural choice for $\mathcal{L}_{\text{rec}}$. However, due to the sparsity of point clouds, this results in an imbalanced loss. To address this, we adopt the Focal Loss (FL) [36]:

$$\ell_{\text{FL}}(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i) = \begin{cases} -(1 - \hat{\boldsymbol{x}}_i)^\gamma \log(\hat{\boldsymbol{x}}_i) & \text{if } \boldsymbol{x}_i = 1 \\ -\hat{\boldsymbol{x}}_i{}^\gamma \log(1 - \hat{\boldsymbol{x}}_i) & \text{otherwise}, \end{cases} \quad (2)$$

$$\mathcal{L}_{\text{rec}} = \sum_{i=0}^{M} \ell_{\text{FL}}(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i), \quad (3)$$

where i is the voxel index and M is the number of voxels.

For object locations $\boldsymbol{y}$, we use the same approach to train a separate VQ-VAE model E_y, Q_y, and G_y. In the subsequent subsections, we use $\boldsymbol{x}^f \in \mathbb{R}^{h \times w \times c_x}$ and $\boldsymbol{y}^f \in \mathbb{R}^{h \times w \times c_y}$ as feature embeddings for learning a latent diffusion model.

3.3. Transfer Your Perspective by Generation

We model the distribution $P(\boldsymbol{x}_r | \boldsymbol{x}_e, \boldsymbol{y}_r)$ defined in Sec. 3.1 by a conditioned generative model and *train it in two stages*. We assume access to $(\boldsymbol{x}_r, \boldsymbol{x}_e, \boldsymbol{y}_r)$ tuples as training data. We detail the training data preparation in Sec. 3.4 and Sec. 4.

In the first stage, we aim to maximize the generation capability by providing minimal conditions, *i.e.*, only bounding boxes, without further constraints on the scene. This encourages the model to produce $P(\boldsymbol{x}|\boldsymbol{y})$ with high flexibility. Second, building on this model, we incorporate the ego agent's point cloud as an additional cue and ground the generation process on it to ensure semantic and layout consistency between the generated and ego agent's point clouds. Put together, this two-stage training procedure enables the learned $P(\boldsymbol{x}_r | \boldsymbol{x}_e, \boldsymbol{y}_r)$ model to generate perception data at areas beyond the commonly visible regions between two views. The model and training process is illustrated in Fig. 2.

More importantly, this approach enables us to leverage existing labeled single-agent datasets such as KITTI [9], NuScenes [2], and Waymo Open Dataset (WOD) [39], since the first-stage training only requires ego-centric LiDAR point clouds and their corresponding semantic information (see Sec. 3.4). The $(\boldsymbol{x}_e, \boldsymbol{x}_r)$ data pairs seen from the ego and the reference agents are needed only in the second stage.

In the following, we elaborate on each training stage.

Stage 1: Generation with semantic information. The goal of this stage is to equip the model with strong generation capabilities of point clouds given spatial conditions, preparing it for the next stage. As discussed in Sec. 3.2, the point clouds and object locations are embedded by VQ-VAEs [41], denoted as $\boldsymbol{x}^f = E_x(\boldsymbol{x}) \in \mathbb{R}^{h \times w \times c_x}$ and $\boldsymbol{y}^f = E_x(\boldsymbol{y}) \in \mathbb{R}^{h \times w \times c_y}$. The image-like feature maps allow us to adopt existing generation algorithms for 2D images [6, 12, 16]. In this paper, we apply one of the most popular generative models, the Latent Diffusion Model (LDM) [34], for conditioned generation $P(\boldsymbol{x}|\boldsymbol{y})$. LDM seeks to model a data distribution by iteratively denoising variables that are initially sampled from a Gaussian. The objective is:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\boldsymbol{x}^f, \boldsymbol{y}^f, \epsilon \sim \mathcal{N}(0,1), t} \left[\left\| \epsilon - \epsilon_\theta \left(\boldsymbol{x}_t^f, \boldsymbol{y}^f, t \right) \right\|_2^2 \right], \quad (4)$$

where, ϵ_θ is a UNet [35] backbone and t is the timestamp.

Stage 2: Generation for new perspectives. In this stage, we aim to ground the generation of point clouds by incorporating the ego agent's perception. To retain the generation capability from stage 1, we freeze the learned $P(\boldsymbol{x}|\boldsymbol{y})$ and introduce a learnable lightweight control module to inject additional cues, following T2I-Adapter [26], as shown in Fig. 2.

Formally, let $\boldsymbol{x}_{e'}$ denote the *translated* and *rotated* ego-agent's point cloud centered around the reference location and aligned with the reference orientation, as shown in Fig. 2; $\boldsymbol{x}_{e'}^f = E_x(\boldsymbol{x}_{e'}) \in \mathbb{R}^{h \times w \times c_x}$ is the corresponding embedding. Let F_{AD} denote the learnable control module, which takes $\boldsymbol{x}_{e'}^f$ as input and outputs

$$\mathbf{F}_c = F_{AD}(\boldsymbol{x}_{e'}^f), \quad (5)$$

where $\mathbf{F}_c = \{\mathbf{F}_c^1, \mathbf{F}_c^2, \mathbf{F}_c^3, \mathbf{F}_c^4\}$ matches the size of the multi-scale features $\mathbf{F}_{enc} = \{\mathbf{F}_{enc}^1, \mathbf{F}_{enc}^2, \mathbf{F}_{enc}^3, \mathbf{F}_{enc}^4\}$ extracted from the frozen encoder of the UNet ϵ_θ. With these ingredients, T2I-Adapter influences the generation process by

$$\mathbf{F}_{enc}^i \leftarrow \mathbf{F}_{enc}^i + \mathbf{F}_c^i, \ i \in \{1, 2, 3, 4\}, \quad (6)$$

which injects $\boldsymbol{x}_{e'}^f$ embedding into each of the denoising steps. The second-stage objective for learning F_{AD} is:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\boldsymbol{x}_r^f, \boldsymbol{y}_r^f, \boldsymbol{x}_{e'}^f, \epsilon \sim \mathcal{N}(0,1), t} \left[\left\| \epsilon - \epsilon_\theta \left(\boldsymbol{x}_{r,t}^f, \boldsymbol{y}^f, \boldsymbol{x}_{e'}^f, t \right) \right\|_2^2 \right], \quad (7)$$

where the encoder of ϵ_θ is modified as defined in Eq. (6).

3.4. From Single Agent to Collaborative Datasets

There are many publicly accessible large-scale real-world autonomous driving datasets for ego-centric 3D perception, like WOD [39], NuScenes [2], KITTI-360 [21], and PandaSet [46], whereas much fewer datasets exist for CAV due to the challenges in data collection as discussed in Sec. 1.

This limitation inspired a bold idea: *Can we transfer the generation capability learned from paired CAV data to these existing labeled single-agent datasets?* Achieving this would scale up CAV datasets substantially, offering immense benefits to the research community. However, this goal is challenging due to domain gaps between datasets, such as variations in LiDAR sensors, point cloud densities, data patterns, and data collection environments. Below, we describe our approach to addressing these challenges.

Problem setup. Our target domain is a single-agent dataset, while the source domain consists of paired perception and semantic information from ego and reference perspectives, denoted as $(\boldsymbol{x}_r, \boldsymbol{x}_e, \boldsymbol{y}_r)$. We demonstrate that we can transfer knowledge from a simulated CAV dataset (e.g., OPV2V [51]) to a real-world ego-centric dataset (e.g., WOD [39]).

Two-Stage training is the key. As described in Sec. 3.3, we decompose the generative training into two stages: first learning $P(\boldsymbol{x}|\boldsymbol{y})$, followed by $P(\boldsymbol{x}_r|\boldsymbol{x}_e, \boldsymbol{y}_r)$. The second stage keeps $P(\boldsymbol{x}|\boldsymbol{y})$ frozen by adding an adapter. This strategy enables us to use target domain data in the first stage. The resulting $P(\boldsymbol{x}|\boldsymbol{y})$ would generate target-like point clouds.

Minimizing domain gaps. In the first stage, we have learned VQ-VAE encoder-decoders and $P(\boldsymbol{x}|\boldsymbol{y})$ in the target domain (single-agent). In the second stage, we bring in CAV data to guide the generation on how to condition on ego-agent's perception. To minimize the gap of $\boldsymbol{x}^f$ between two domains—*$\boldsymbol{x}^f$ is what the generative model is optimized to generate*—we adopt a GAN-style discriminator [8, 10, 22, 40]. Specifically, we aim to make feature embeddings extracted from the source data and the target data indistinguishable; namely, to confuse a discriminator D trained to differentiate them. We freeze all parameters of the target domain's encoder E_{tgt}, codebook Q_{tgt}, and decoder G_{tgt} in the VQ-VAE. For the source domain, we initialize its E_{src} and G_{src} by E_{tgt} and G_{tgt} while *reusing the frozen codebook* Q_{tgt}. We then learn the discriminator D and the source domain's encoder E_{src} and decoder G_{src} in an interleaving fashion by minimizing the following objective functions:

$$\text{For } D\text{: } \mathcal{L}_D = \begin{array}{l} \mathbb{E}_{\boldsymbol{x} \sim P_{tgt}} [\max(0, 1 - D(E_{tgt}(\boldsymbol{x})))] \\ + \mathbb{E}_{\boldsymbol{x} \sim P_{src}} [\max(0, 1 + D(E_{src}(\boldsymbol{x})))] \, ; \end{array} \quad (8)$$

$$\text{For VQ-VAE: } \mathcal{L}_{\text{vq}} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{src} + \|\text{sg}[\boldsymbol{x}^z] - E_{src}(\boldsymbol{x})\|_2^2, \quad (9)$$

$$\text{where } \mathcal{L}_{src} = -\mathbb{E}_{\boldsymbol{x} \sim P_{src}} D(E_{src}(\boldsymbol{x})). \quad (10)$$

The other terms in Eq. (9) follow those in Eq. (1), where for $\boldsymbol{x} \sim P_{src}$, $\hat{\boldsymbol{x}} = G_{src} \circ Q_{tgt} \circ E_{src}(\boldsymbol{x})$ in $\mathcal{L}_{rec}$. We note that $\mathcal{L}_D$ takes in two streams of data (*i.e.*, source and target) and is defined by the Hinge Loss [22] on $\boldsymbol{x}^f$; $\mathcal{L}_{\text{vq}}$ takes in only the source CAV data. We also note that embeddings for object locations does not require adaptation.

After this adaptation step, we proceed into the second stage of generative training, where we drop the discriminator and train the control module defined in Eq. (5) by minimizing Eq. (7). The feature embeddings are produced by E_{src} on top of $(\boldsymbol{x}_r, \boldsymbol{x}_e, \boldsymbol{y}_r) \sim P_{src}$.

Enhancement in target domains. With the discriminator and adversarial training, we reduce the gap between source and target domain data, successfully adapting TYP to the single-agent dataset, as shown in Sec. 4.5. To further improve the conditioned generation quality in the target domain, a further fine-tuning stage in it is desired. However, *how can we do so without having reference agents' point clouds in the target domain?* Here, we propose two simple solutions.

Method	Train Data	AP 0.5			
		s	m	l	all
No Fusion	ego's gt only	0.67	0.41	0.13	0.40
Early Fusion [4]	gt **(oracle)**	0.76	0.42	0.31	0.49
	TYP **(ours)**	0.75	0.38	0.29	0.46
Late Fusion [51]	gt **(oracle)**	0.74	0.57	0.36	0.55
	TYP **(ours)**	0.71	0.49	0.32	0.50
AttFuse [51]	gt **(oracle)**	0.94	0.80	0.62	0.77
	TYP **(ours)**	0.90	0.73	0.56	0.72
V2X-ViT [50]	gt **(oracle)**	0.87	0.71	0.50	0.71
	TYP **(ours)**	0.84	0.65	0.40	0.65

Table 1. **Results on OPV2V.** We compare the performance using reference generated by TYP versus ground-truth (oracle). The comparable results demonstrate the quality of our generated data.

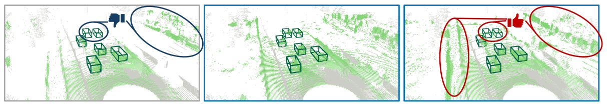

Figure 3. **Qualitative results on enhancement in the target domain.** Generated point cloud (green) has better quality with the enhancement given ego (gray) from Waymo (*cf*. Sec. 3.4).

Recall that in training $P(\boldsymbol{x}_r|\boldsymbol{x}_e, \boldsymbol{y}_r)$ using the source CAV data, point clouds show up in both the input as conditions and the outputs as supervisions. In other words, the domain gap exists on both sides, and we aim to reduce it by further fine-tuning *using the target single-agent dataset*. First, to enhance the output generation in the target domain, we set ego's point clouds to empty, *i.e.*, $\boldsymbol{x}_e = 0^{H\times W\times C}$ for $(\boldsymbol{x}_r, \boldsymbol{y}_r) \sim P_{tgt}$. Then, to enable adaptation on inputs, we apply self-training [18, 28, 60, 61], randomly sampling a reference location in the scene to generate pseudo-point clouds for fine-tuning, *i.e.*, $(\hat{\boldsymbol{x}}_r, \boldsymbol{x}_e, \boldsymbol{y}_r)$ for $(\boldsymbol{x}_e, \boldsymbol{y}_r) \sim P_{tgt}$. We use the above target domain data and the source domain data to jointly fine-tune the control module (*cf*. Sec. 3.3, Eq. (6), and Eq. (7)) further. Empirical results in Sec. 4.5 show the effectiveness of the proposed solution in turning single-agent datasets into collaborative datasets for CAV development.

4. Experiments

4.1. Setup

Datasets. In this study, we primarily use three datasets: OPV2V [51], V2V4Real [52], and WOD [39]. OPV2V is a simulation-based CAV dataset that includes over 70 diverse driving scenes and more than 11,000 frames. We utilize this dataset initially to validate our proposed research problem and later to guide the generation training on single-agent datasets. V2V4Real is a real-world CAV dataset with approximately 20,000 LiDAR frames, which we use to validate TYP in a real-world CAV context, treating it as a single-agent dataset (i.e., without using any labels to train TYP). Lastly, we adopt WOD to scale up TYP, one of the largest real-world single-agent datasets available.

Evaluation metrics. Following standard benchmarks [51, 52], we report Average Precision (AP 0.5) across three distance ranges: 0-30 meters (short, denoted as s), 30-50 meters (medium, m), and 50-100 meters (long, l) and overall (all).

Method	Train Data	AP 0.5			
		s	m	l	all
No Fusion	ego's gt only	0.71	0.33	0.08	0.46
Early Fusion [4]	gt **(oracle)**	0.79	0.42	0.45	0.60
	TYP **(ours)**	0.76	0.36	0.30	0.53
Late Fusion [51]	gt **(oracle)**	0.79	0.53	0.56	0.67
	TYP **(ours)**	0.77	0.48	0.51	0.63
AttFuse [51]	gt **(oracle)**	0.80	0.45	0.36	0.61
	TYP **(ours)**	0.75	0.45	0.26	0.56
V2X-ViT [50]	gt **(oracle)**	0.81	0.49	0.30	0.61
	TYP **(ours)**	0.76	0.38	0.17	0.48

Table 2. **Results on V2V4Real.** We compare the performance using reference generated by TYP versus ground-truth (oracle). Note that *we do not use V2V4Real data to train the generation*, treating it as a single-agent dataset. The comparable results demonstrate the quality of our generated data and the transferability of TYP.

Implementations. For TYP training, we voxelize point clouds within the range $[-51.2, 51.2]$ meters in x and y dimensions and 4 meters in z (range is dataset-dependent) to create 3D volumes, $\boldsymbol{x} \in \mathbb{R}^{512\times512\times20}$. This is then encoded to $\boldsymbol{x}^f \in \mathbb{R}^{64\times64\times8}$ with VQ-VAE [41]. Similarly, semantic information $\boldsymbol{y} \in \mathbb{R}^{512\times512\times1}$ is encoded to $\boldsymbol{y}^f \in \mathbb{R}^{64\times64\times3}$. We train VQ-VAE for $\boldsymbol{x}$ 120 epochs and for $\boldsymbol{y}$ 10 epochs, with a batch size of 6 per GPU. Next, we train LDM with a batch size of 48 per GPU for 200 epochs in the first stage and 80 epochs in the second stage. During the second stage, we freeze the U-Net from the first stage and fine-tune only the control module, following [26]. For adaptation, we initialize the weights of encoder-decoder for CAV data (fake) with the model trained on single-agent data (real), freeze its codebook, and fine-tune it with a discriminator comprising 4 convolutional layers. The model processes an equal amount of fake and real per batch (i.e., 3 samples each). All training is conducted on 4 NVIDIA H100 GPUs. For experiments on benchmarks, we follow the convention to report results with Pointpillars [17]. We train / fine-tune all models with 8 P100, with batch size 4 per GPU. For V2X-ViT, we use 4 H100 due to resources required.

Inference of generation and post-processing. We use DDIM scheduler [38] for 200 steps. For post-processing, we apply modified Gumbel-Softmax [15] to the logits. Specifically, we define two thresholds, low and high. Scores below the low threshold are dropped, those above the high threshold are retained, and values in between are perturbed with Gaussian noise on logits. Hyperparameters are tuned on a small hold-out set. Finally, we convert the volume back to coordinates by using the centers of the occupied voxels.

4.2. Do We Need Collaborative Perception?

To explore this question, we consistently report the performance without collaborative agents (denoted as "ego's gt

Method	Pre-Train	FT. on 0 Scene				FT. on 5 Scene				FT. on 10 Scene				FT. on 32 Scene			
		s	m	l	all	s	m	l	all	s	m	l	all	s	m	l	all
No Fusion	ego's gt only					0.44	0.16	0.06	0.29	0.66	0.29	0.08	0.42	0.71	0.34	0.08	0.47
Early Fusion [4]	scratch					0.44	0.18	0.16	0.30	0.73	0.28	0.24	0.48	0.79	0.42	0.45	0.60
	OPV2V	**0.54**	0.20	0.07	0.31	0.32	0.18	0.15	0.25	0.73	0.32	0.31	0.50	0.80	0.47	**0.54**	**0.65**
	ColWaymo (**ours**)	0.50	**0.24**	**0.24**	**0.35**	**0.68**	**0.34**	**0.32**	**0.51**	**0.78**	**0.39**	**0.33**	**0.57**	**0.83**	**0.48**	0.50	**0.65**
Late Fusion [51]	scratch					0.44	0.24	0.33	0.34	0.72	0.42	**0.51**	0.58	0.79	0.53	0.56	0.67
	OPV2V	**0.60**	**0.27**	**0.28**	**0.44**	0.55	0.26	0.38	0.42	0.73	0.42	**0.51**	0.59	0.78	0.55	0.58	0.67
	ColWaymo (**ours**)	0.40	0.19	0.15	0.25	**0.60**	**0.28**	**0.44**	**0.47**	**0.76**	**0.43**	**0.51**	**0.61**	**0.82**	**0.58**	**0.61**	**0.71**
AttFuse [51]	scratch					0.40	0.15	0.10	0.28	0.70	0.30	0.17	0.47	0.80	0.45	0.36	0.61
	OPV2V	0.51	0.19	0.05	0.31	0.54	0.22	0.11	0.37	0.77	0.40	0.21	0.54	0.83	0.53	0.40	0.65
	ColWaymo (**ours**)	**0.66**	**0.35**	**0.11**	**0.45**	**0.65**	**0.29**	**0.16**	**0.46**	**0.83**	**0.46**	**0.33**	**0.61**	**0.88**	**0.58**	**0.53**	**0.72**
V2X-ViT [50]	scratch					0.43	0.15	0.12	0.31	0.70	0.28	0.17	0.43	0.81	0.49	0.30	0.61
	OPV2V	0.51	0.24	0.07	0.33	0.48	0.23	0.16	0.35	0.76	0.38	0.22	0.53	0.81	0.49	0.35	0.61
	ColWaymo (**ours**)	**0.60**	**0.28**	**0.10**	**0.34**	**0.66**	**0.28**	**0.22**	**0.46**	**0.79**	**0.48**	**0.26**	**0.58**	**0.84**	**0.57**	**0.44**	**0.67**

Table 3. **Results with pre-training on collaborative Waymo.** We scale up TYP with exiting large-scale single-agent dataset WOD [39], creating its collaborative version "ColWaymo". These data are used for pre-training and subsequently fine-tuned on V2V4Real [52]. Results show significant improvement over training from scratch, highlighting the potential to reduce data collection efforts, scale up, and accelerate CAV development. Comparisons to simulated OPV2V [51] pre-training further demonstrate the realism of our generated point clouds.

Method	Disc.	ST.	Dum.	JSD (↓)	MMD (↓)
No Adaptation				0.26	4.61×10^{-4}
Adaptation	✓			0.16	1.10×10^{-4}
	✓	✓		0.17	1.30×10^{-4}
	✓	✓	✓	0.16	1.17×10^{-4}

Table 4. **Ablation of proposed adaption method**. Results with adaptation outperform no adaptation, demonstrating an improvement in generation quality. Abbreviations in title refer to discriminator, self-training, and dummy ego, respectively (*cf*. Sec. 3.4).

Num. Steps	Epochs 1st	Epochs 2nd	JSD (↓)	MMD (↓)
1	–	280	0.11	6.45×10^{-5}
2	200	80	**0.10**	$\mathbf{5.28 \times 10^{-5}}$

Table 5. **2-Stage training.** Results demonstrate the benefit of the proposed 2-stage training in generation quality on OPV2V [51].

only" in Tab. 1, Tab. 2, and Tab. 3). For example, in the V2V4Real dataset, a well-trained detector struggles with objects beyond 50 meters, achieving only 0.08 AP at that range, though it performs well on nearby objects with an AP of 0.71 (*cf*. first row of Tab. 3). simply combining the predictions from two under-trained detectors, we observe a notable boost in performance to 0.33 (*cf*. "scratch" in late fusion, Tab. 2), suggesting the significance of CAV.

4.3. Point Clouds Generation on OPV2V

Setting. To validate our research concept, we begin with the simulated dataset OPV2V [51]. We split the original training set of 44 scenes into two halves, using only the first 22 scenes to train our generation model. For the remaining 22 scenes, we generate point clouds for reference agents based on the ego agent's point clouds and compare these results to ground-truth point clouds. We follow the benchmark to report performance on the validation set from Culver City.
Baselines. Following the literature [51, 52], we adopt three common fusion algorithms for CAV: early, late, and intermediate fusion, which fuse input point clouds, extracted features, and predicted bounding boxes, respectively. For intermediate fusion, we adpot AttFuse [51] and V2X-VIT [50].
Comparison to using ground-truth LiDAR. In Tab. 1, we show that training CAV with point clouds generated by TYP achieves performance comparable to that obtained with ground-truth (oracle) across all methods (*e.g*., 0.46 *vs*. 0.49 with early, 0.50 *vs*. 0.55 with late, and 0.77 *vs*. 0.72 with AttFuse [51], etc.) This result highlights the quality of the generated point clouds. In the supplemental material, we further demonstrate that with additional labeled data (from the first half of the split used for training generation), the gap to oracle performance can be further minimized.

4.4. Single-Agent to Collaborative Multi-Agent

Setting. Building on our findings with simulated data (*cf*. Sec. 4.3), we validate our approach on the real-world dataset V2V4Real [52], *treating it as a single-agent dataset without access to reference agents' point clouds*. Specifically, we use the model from Sec. 4.3 to generate reference car's point clouds from the ego's perspective and compare performance against ground-truth point clouds (oracle). Note that *no V2V4Real data is used to train the generation*.
Results on V2V4Real. In Tab. 2, we show the performance using generated reference point clouds is comparable to the oracle, supporting our idea of *translating a single-agent dataset to multi-agent CAV*. Xu *et al*. previously explored OPV2V-to-V2V4Real adaptation, noting significant performance drops even with adaptation algorithms. Here, TYP offers a new solution to this challenge – in a generative way, providing a fresh perspective on domain adaptation.

4.5. ColWaymo: Scaling with Large-Scale Data

Scaling CAV to the next level. In Sec. 4.4, we show the potential to create realistic reference's point clouds on real-

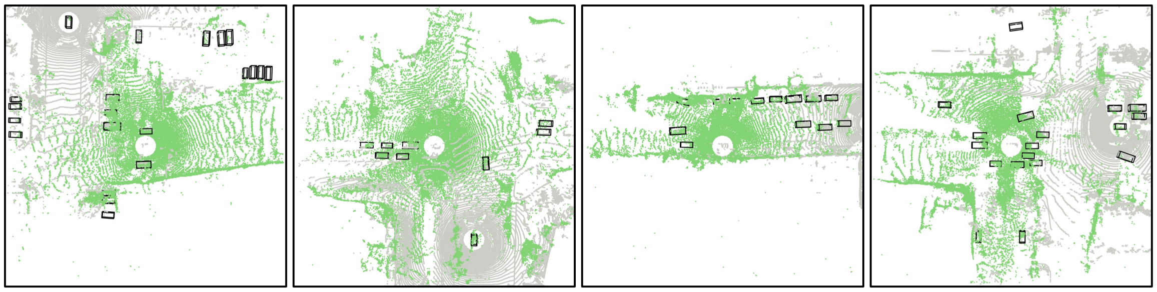

Figure 4. **Qualitative results on Collaborative Waymo.** The gray point clouds are from the original single-agent dataset and the green are generated by TYP conditioning on them.

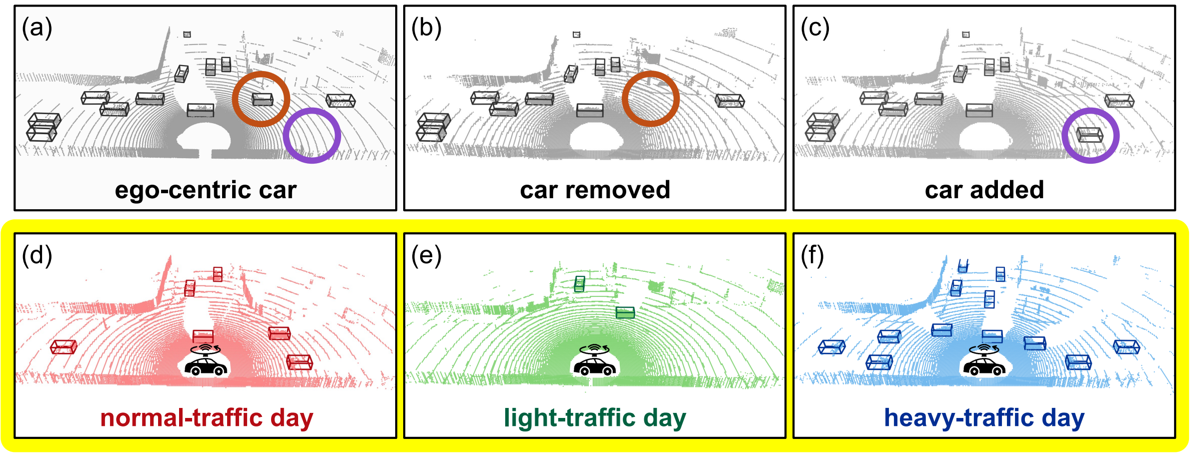

Figure 5. **Qualitative results.** Our proposed TYP is capable of scene editing, by inputting the same point cloud but different object locations. We (b) remove and (c) add a car from (a) the original point cloud. Inspired by the idea of past traversals [56], we apply completely different traffic conditions and generate (d), (e), and (f), to imagine driving through the same intersection.

world dataset V2V4Real [52], through a simulation-guided training of TYP. Here, our goal is broader: to transform an existing large-scale single-agent dataset (*e.g.*, WOD [39]) into a collaborative version. Using the pipeline described in Sec. 3 and illustrated in Fig. 2, we train the generative model on $17,400$ samples from the WOD training set. We then use its validation set to generate approximately $36,000$ collaborative samples by sampling reference locations from labeled vehicles. These generated samples serve to pre-train a model, which we subsequently fine-tune on V2V4Real [52]. For comparison, we also conduct pre-training on OPV2V [51].

Results of fine-tuning on V2V4Real. Following prior work [1, 29, 47, 54], we perform pre-training followed by fine-tuning on limited labeled data to validate the potential of TYP for scaling up CAV development. As shown in Tab. 3, pre-training on our generated data yields significantly better performance than training from scratch. Additionally, compared to pre-training on the simulated dataset OPV2V, ColWaymo consistently achieves higher performance, highlighting the realism of the generated point clouds. These results demonstrate that TYP effectively transforms an ego-only dataset into a collaborative version, and potentially reducing the need for extensive CAV data collection.

Generation quality by proposed adaptation. To assess the quality of generated data, we use two common metrics: Jensen-Shannon Divergence (JSD) and Maximum Mean Discrepancy (MMD). As shown in Tab. 4, our adaptation method improves JSD from 0.26 to 0.16 and MMD from 4.61 to 1.17 compared to the baseline without adaptation. Additionally, Fig. 3 visually illustrates this improvement.

4.6. Qualitative Results

We illustrate the proposed problem in Fig. 1, which highlights the benefits of collaboration by allowing an agent to "see" beyond the limitations of its own sensors. TYP generates realistic point clouds from various viewpoints, enabling the agents to perceive previously occluded objects while preserving the overall scene semantics. Fig. 4 shows our ColWaymo, and Fig. 5 further demonstrates the model's capability for scene editing.

5. Conclusion

We introduce a new research direction in collaborative driving and present the first solution. Empirical results show our approach can significantly reduce data collection and development efforts, advancing safer autonomous systems.

Limitations and future work. Following existing benchmarks [51, 52], TYP focuses on vehicle-like objects. Future work could extend it to broader objects and static entities (*e.g.*, traffic signs, signals) essential for real-world traffic.

Acknowledgment

This research is supported in part by grants from the National Science Foundation (IIS-2107077, IIS-2107161). We are thankful for the generous support of the computational resources by the Ohio Supercomputer Center.

References

[1] Alexandre Boulch, Corentin Sautier, Björn Michele, Gilles Puy, and Renaud Marlet. Also: Automotive lidar self-supervision by occupancy estimation. In *CVPR*, 2023. 8

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 4, 5

[3] Qi Chen, Xu Ma, Sihai Tang, Jingda Guo, Qing Yang, and Song Fu. F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019. 1

[4] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. In *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019. 1, 6, 7

[5] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017. 4

[6] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 4

[7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *CoRL*, 2017. 2, 3

[8] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016. 5

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 4

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 3, 5

[11] Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. In *NeurIPS*, 2024. 4

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 3, 4

[13] Qianjiang Hu, Zhimin Zhang, and Wei Hu. Rangeldm: Fast realistic lidar point cloud generation. In *ECCV*, 2025. 2, 3

[14] Yue Hu, Shaoheng Fang, Zixing Lei, Yiqi Zhong, and Siheng Chen. Where2comm: Communication-efficient collaborative perception via spatial confidence maps. In *NeurIPS*, 2022. 1

[15] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017. 6

[16] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *NeurIPS*, 2021. 4

[17] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 4, 6

[18] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013. 6

[19] Yiming Li, Shunli Ren, Pengxiang Wu, Siheng Chen, Chen Feng, and Wenjun Zhang. Learning distilled collaboration graph for multi-agent perception. In *NeurIPS*, 2021. 1

[20] Yiming Li, Dekun Ma, Ziyan An, Zixun Wang, Yiqi Zhong, Siheng Chen, and Chen Feng. V2x-sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving. In *Robotics and Automation Letters*, pages 10914–10921. IEEE, 2022. 3

[21] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE TPAMI*, 45(3):3292–3310, 2022. 5

[22] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 5

[23] Yen-Cheng Liu, Junjiao Tian, Nathaniel Glaser, and Zsolt Kira. When2com: Multi-agent perception via communication graph grouping. In *CVPR*, 2020. 1

[24] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015. 4

[25] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *IROS*, 2019. 4

[26] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *AAAI*, 2024. 5, 6

[27] Kazuto Nakashima and Ryo Kurazume. Lidar data synthesis with denoising diffusion probabilistic models. In *ICRA*, 2024. 2, 3

[28] Tai-Yu Pan, Qing Liu, Wei-Lun Chao, and Brian Price. Towards open-world segmentation of parts. In *CVPR*, 2023. 6

[29] Tai-Yu Pan, Chenyang Ma, Tianle Chen, Cheng Perng Phoo, Katie Z Luo, Yurong You, Mark Campbell, Kilian Q Weinberger, Bharath Hariharan, and Wei-Lun Chao. Pre-training lidar-based 3d object detectors through colorization. In *ICLR*, 2023. 8

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 4

[31] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 4

[32] Haoxi Ran, Vitor Guizilini, and Yue Wang. Towards realistic scene generation with lidar diffusion models. In *CVPR*, 2024. 2, 3

[33] Zaydoun Yahya Rawashdeh and Zheng Wang. Collaborative automated driving: A machine learning-based method to enhance the accuracy of shared information. In *International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018. 1

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2, 4

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 5

[36] T-YLPG Ross and GKHP Dollár. Focal loss for dense object detection. In *CVPR*, 2017. 4

[37] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 4

[38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 6

[39] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 2, 3, 4, 5, 6, 7, 8

[40] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017. 2, 5

[41] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *NeurIPS*, 2017. 4, 6

[42] Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In *ECCV*, 2020. 1

[43] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *ICRA*, 2018. 4

[44] Yang Wu, Kaihua Zhang, Jianjun Qian, Jin Xie, and Jian Yang. Text2lidar: Text-guided lidar point cloud generation via equirectangular transformer. In *ECCV*. Springer, 2024. 2, 3

[45] Hao Xiang, Zhaoliang Zheng, Xin Xia, Runsheng Xu, Letian Gao, Zewei Zhou, Xu Han, Xinkai Ji, Mingxi Li, Zonglin Meng, et al. V2x-real: a largs-scale dataset for vehicle-to-everything cooperative perception. *arXiv preprint arXiv:2403.16034*, 2024. 1

[46] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. Pandaset: Advanced sensor suite dataset for autonomous driving. In *International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021. 5

[47] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pretraining for 3d point cloud understanding. In *ECCV*, 2020. 8

[48] Yuwen Xiong, Wei-Chiu Ma, Jingkang Wang, and Raquel Urtasun. Learning compact representations for lidar completion and generation. In *CVPR*, 2023. 3, 4

[49] Yuwen Xiong, Wei-Chiu Ma, Jingkang Wang, and Raquel Urtasun. Ultralidar: Learning compact representations for lidar completion and generation. In *CVPR*, 2023. 2, 4

[50] Runsheng Xu, Hao Xiang, Zhengzhong Tu, Xin Xia, Ming-Hsuan Yang, and Jiaqi Ma. V2x-vit: Vehicle-to-everything cooperative perception with vision transformer. In *ECCV*, 2022. 1, 6, 7

[51] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *ICRA*, 2022. 2, 3, 5, 6, 7, 8

[52] Runsheng Xu, Xin Xia, Jinlong Li, Hanzhao Li, Shuo Zhang, Zhengzhong Tu, Zonglin Meng, Hao Xiang, Xiaoyu Dong, Rui Song, et al. V2v4real: A real-world large-scale dataset for vehicle-to-vehicle cooperative perception. In *CVPR*, 2023. 1, 3, 6, 7, 8

[53] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 4

[54] Junbo Yin, Dingfu Zhou, Liangjun Zhang, Jin Fang, Cheng-Zhong Xu, Jianbing Shen, and Wenguan Wang. Proposalcontrast: Unsupervised pre-training for lidar-based 3d object detection. In *ECCV*, 2022. 8

[55] Jinsu Yoo, Zhenyang Feng, Tai-Yu Pan, Yihong Sun, Cheng Perng Phoo, Xiangyu Chen, Mark Campbell, Kilian Q Weinberger, Bharath Hariharan, and Wei-Lun Chao. Learning 3d perception from others' predictions. In *ICLR*, 2025. 1

[56] Yurong You, Katie Z Luo, Xiangyu Chen, Junan Chen, Wei-Lun Chao, Wen Sun, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Hindsight is 20/20: Leveraging past traversals to aid 3d perception. In *ICLR*, 2022. 8

[57] Haibao Yu, Yizhen Luo, Mao Shu, Yiyi Huo, Zebang Yang, Yifeng Shi, Zhenglong Guo, Hanyu Li, Xing Hu, Jirui Yuan, and Zaiqing Nie. Dair-v2x: A large-scale dataset for vehicle-infrastructure cooperative 3d object detection. In *CVPR*, 2022. 3

[58] Chris Zhang, Wenjie Luo, and Raquel Urtasun. Efficient convolutions for real-time semantic segmentation of 3d point clouds. In *3DV*, 2018. 4

[59] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 4

[60] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. Rethinking pretraining and self-training. In *NeurIPS*, 2020. 6

[61] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018. 6

[62] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *ECCV*, 2022. 2, 3

[63] Vlas Zyrianov, Henry Che, Zhijian Liu, and Shenlong Wang. Lidardm: Generative lidar simulation in a generated world. *arXiv preprint arXiv:2404.02903*, 2024. 2, 3