This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Stable-Sim2Real: Exploring Simulation of Real-Captured 3D Data with Two-Stage Depth Diffusion

Mutian Xu[1] Chongjie Ye[2,3,1] Haolin Liu[4] Yushuang Wu[5] Jiahao Chang[1] Xiaoguang Han[1,2,3]*

[1]SSE, CUHKSZ [2]FNii-Shenzhen
[3]Guangdong Provincial Key Laboratory of Future Networks of Intelligence, CUHKSZ
[4]Tencent Hunyuan3D [5]ByteDance Games

Abstract

3D data simulation aims to bridge the gap between simulated and real-captured 3D data, which is a fundamental problem for real-world 3D visual tasks. Most 3D data simulation methods inject predefined physical priors but struggle to capture the full complexity of real data. An optimal approach involves learning an implicit mapping from synthetic to realistic data in a data-driven manner, but progress in this solution has met stagnation in recent studies. This work explores a new solution path of data-driven 3D simulation, called Stable-Sim2Real, based on a novel two-stage depth diffusion model. The initial stage finetunes Stable-Diffusion to generate the residual between the real and synthetic paired depth, producing a stable but coarse depth, where some local regions may deviate from realistic patterns. To enhance this, both the synthetic and initial output depth are fed into a second-stage diffusion, where diffusion loss is adjusted to prioritize these distinct areas identified by a 3D discriminator. We provide a new benchmark scheme to evaluate 3D data simulation methods. Extensive experiments show that training the network with the 3D simulated data derived from our method significantly enhances performance in real-world 3D visual tasks. Moreover, the evaluation demonstrates the high similarity between our 3D simulated data and real-captured patterns. Project page: mutianxu.github.io/stable-sim2real.

1. Introduction

In recent years, real-world 3D datasets have played a crucial role in addressing a wide range of tasks in 3D vision and robotics [1, 7, 11, 15, 23, 52, 69, 70]. However, collecting real 3D data is often labor-intensive and time-consuming, and is further complicated by the recent concerns regarding data privacy. In this scenario, synthetic (*i.e.*, simulated) data [8, 10, 12, 16, 68] emerges as an alternative data resource that is cost-effective, rapid, and scalable. Nevertheless, models trained using synthetic data may not perform robustly against the real-world signal.

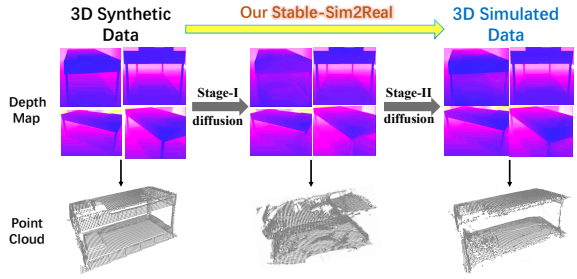

Figure 1. We delve back into a substantial yet stagnant problem of simulating real-captured 3D data in a data-driven manner, introducing *Stable-Sim2Real*. Given the depth maps of the 3D synthetic data, our approach applies a two-stage depth diffusion model based on Stable Diffusion [42] to generate realistic depth maps that are fused to produce real 3D point clouds. The initial stage generates stable but coarse depth maps, followed by a second-stage diffusion to enhance regional details.

This problem has catalyzed the development of 3D data simulation techniques, to minimize the gap between simulated and real-captured 3D data. While some studies [2, 30, 33, 39, 75] have explored grounding physical priors to simulate depth sensors, they struggle to capture the full complexity of real-world patterns and are confined to simulation environments, due to the reliance on predefined and *explicit* physical modeling. A better solution involves learning an *implicit* mapping from synthetic to realistic data in a data-driven manner, offering greater flexibility and adaptability to real-world diversities. However, only a handful of earlier works [4, 19, 47, 56] have ventured down this path. Hindered by the lack of synthetic-real paired datasets and the dependence on conventional networks like GANs [18], which are prone to training instability and mode collapse [29], these methods have yet to demonstrate the anticipated performance levels. As a result, developing data-driven 3D

*Corresponding author

Sim2Real methods has long been a challenge within the community, and has *shown little progress*.

In this paper, we aim to explore the data-driven 3D Sim2Real and refocus the community's attention on this critical problem. As a prerequisite, we choose a recent synthetic-real paired dataset, LASA [31], containing 10,412 high-quality 3D shape CAD annotations aligned with the real-world object scans. Driven by LASA, our efforts focus on designing an effective data-driven 3D Sim2Real algorithm. Given the inherent *indeterminacy* and *diversity* in real-captured data patterns, deep generative models emerge as a natural fit. Among these models, diffusion models [20, 53–55, 62] recently offer easier training dynamics and stronger generation ability. Yet, due to the lack of 3D data, it is hard to train a 3D diffusion model to get strong 3D priors for 3D simulation. Therefore, we opt to leverage the strong generalization priors of *2D* diffusion foundation models (*e.g.*, SD – Stable Diffusion [42]) to simulate real 2D depth maps, which are then fused to gain 3D data. This strategy is similar to real 3D data acquisition, where 2D depth is captured and fused into 3D.

To this end, an intuitive baseline is to sample CAD (*i.e.*, synthetic) depth maps and their paired real depth images from LASA, and then finetune SD to learn an implicit mapping between them. Although such a solution powered by large-scale paired data and the foundation diffusion model has shown effectiveness in conventional image translations, it faces ***special challenges*** in our task: while conventional image translations (*e.g.*, denoising/restoration) normally remove noise to produce *clean* images, our task aims to output *noisy* depth that is highly *un*certain, thereby making the distribution that needs to be learned *more complex*.

To address these challenges, we propose Stable-Sim2Real. The core is a novel two-stage depth diffusion model to simulate real-captured noise (a sample is shown in Fig. 1). At Stage-I diffusion, instead of directly generating the corresponding real-world depth map, our model generates the *residual* (*i.e.*, difference) between real and CAD depth maps. The simulated depth map is then yielded by adding the generated residual with the CAD depth. Empirically, this residual represents the noise added to clean (*i.e.*, synthetic) depth during real-world scanning. Moreover, compared to directly generating noisy real depth, adding noise to CAD depth, which is inherently clean and view-consistent, can produce more *stable* depth with smaller view variation and better preservation of the original geometry (see Sec. 3.2 for more discussions and analysis).

However, according to our observations and experiments, while some regions within the Stage-I generated depth map successfully fit the realistic pattern, some unsatisfactory areas still exhibit conspicuous 3D geometric discrepancies between the generation and real captures (shown in Fig. 6). To tackle this issue, our model then revolves around *discriminating* unsatisfactory areas and *enhancing* the Stage-I generation at local regions. We train a 3D-Aware Discriminator to distinguish between Stage-I generation and real captures at local geometries, which is expected to delineate the *distribution boundary* between them. If a generated patch can be successfully discriminated, it signifies an unsatisfactory area. Next, we introduce a Stage-II diffusion, conditioned not only on the CAD depth but also on Stage-I generation. More importantly, the diffusion loss is adjusted to *prioritize* on unsatisfactory regions. In this way, Stage-II diffusion is ***specialized*** on learning the distribution of unsatisfactory areas, thereby enhancing Stage-I generation. Finally, the generated depth maps are fused to derive the simulated 3D data. Notably, the binary classifier is only used during the training phase.

Last, we provide a comprehensive benchmark scheme for 3D data simulation. The key logic is: if a model's *real-world* performance can be improved after being trained *with simulated data*, it would validate the effectiveness of simulation methods. We focus on fundamental real-world 3D tasks: 3D shape reconstruction and 3D object/scene understanding. As for shape reconstruction, we pretrain a 3D reconstruction network that takes simulated 3D data as input and outputs clean 3D surfaces. For 3D object/scene understanding, the simulated 3D data is used to pretrain a self-supervised point cloud learning framework. To better assess the performance gain purely from simulated data *itself*, we conduct *few*-shot evaluation on pre-trained networks.

Our contributions are summarized as:

- We delve back into a substantial yet stagnant problem, data-driven 3D real data simulation, paving a new solution path of training generative foundation models on extensive synthetic-real paired data.
- We present *Stable-Sim2Real*, a novel 3D real data simulation method based on a two-stage depth diffusion model that can stably simulate real-captured noise.
- We provide a comprehensive scheme to benchmark 3D data simulation, based on few-shot evaluation on networks trained with simulated data for real-world 3D tasks.
- Extensive qualitative and quantitative results show that the data simulated by our method significantly boosts model performance on real-world 3D tasks.

2. Related Work

2.1. Simulation of Real-World Data

Simulation of 2D real-world data. In the realm of 2D real-world data simulation, two predominant approaches have emerged: domain randomization and domain adaptation. The first line of works [57, 58, 60] randomly perturbs textures, lighting, and material of synthetic data, thereby modeling the synthetic-real domain gap as one kind of variation to endow models with robustness to real-world data.

However, it brought the challenge of achieving a delicate balance between ensuring that the distribution of augmentations adequately covers that of real-world data and preventing the augmentation from being too extensive to hinder effective model training [24, 37, 72]. Domain adaptation, on the other hand, aims to minimize the domain gap by transferring knowledge from synthetic data to real-world scenarios, leveraging techniques such as adversarial training and self-supervised learning [5, 49, 71].

Simulation of 3D real-captured data. Acquiring real-world 3D data and their annotation is labor intensive and inefficient, deriving methods for the simulation of 3D data. Existing simulation methods primarily focus on enhancing the realism of *depth* information by grounding physical priors in simulators to close the domain gap with 3D data from the physical world [2, 30, 33, 39, 75]. However, these approaches are usually limited by their reliance on predefined and explicit physical modeling. Alternatively, data-driven methods turn to implicitly learn from the distribution in real-captured data, independent of physical modeling [4, 19, 47, 56], offering enhanced adaptability and generalization capabilities to various real-world scenarios. Nonetheless, due to the absence of extensive synthetic-real paired datasets and the reliance on traditional GANs [18], which suffer from training instability and mode collapse [29], these approaches have not achieved the expected performance. Consequently, the research progress of data-driven 3D Sim2Real techniques faces a recent stagnation.

2.2. Image Generation and Translation.

Conventional generative models. We formulate the 3D Sim2Real task as learning the mapping from synthetic to real-world captured noise, based on depth images. Existing approaches in the domain of image generation and translation include a variety of generative models such as Generative Adversarial Networks (GANs) [18], Variational Autoencoders (VAEs) [28], Cycle-Consistent Adversarial Networks (Cycle-GAN) [77], and Pix2Pix [22]. While GAN-based methods, which have been predominant in image generation before diffusion models, achieved significant milestones in Sim2Real tasks [6, 65, 71], they often struggle with maintaining local details and semantic consistency, leading to sub-optimal performance [13, 29, 32, 36, 38].

Diffusion models. Diffusion models, also known as denoising diffusion probabilistic models (DDPMs) [50], have emerged as a powerful alternative to GANs for image synthesis [13, 32, 34, 35, 43, 44, 74]. Unlike GANs, diffusion models offer a more stable training procedure and have been shown to generate images with higher fidelity and better preservation of fine details [21, 42]. Depth diffusion models recently track great attention. They mainly focus on the task of depth estimation in a monocular [14, 26, 45, 46, 59, 64] or multi-view [27, 67] manner. Other works also cover conventional image translation tasks, such as depth map super-resolution [48] and refinement [76], which produce *clean* and *certain* output images. Differently, our work leverages generative models to "generate *noise*", where the expected output is highly *un*certain. Thus, the distribution that needs to be learned is more complex, bringing fundamentally different challenges.

3. Method

We hope to exploit the strong prior of the diffusion-based foundation model, so we take *depth* map as our bridge towards the simulation of real-captured 3D data. The preliminaries on the diffusion model are provided in Sec. 3.1. Our Stable-Sim2Real is a two-stage depth diffusion framework, as illustrated in Fig. 2. The first-stage model (Sec. 3.2) learns to generate the depth residual, producing stable but coarse depth maps, where some areas are distinct from the realistic pattern. Therefore, a 3D binary classifier (Sec. 3.3) is introduced to discriminate these distinct areas. The second-stage model (Sec. 3.4) is specialized to enhance the generation of these areas. Finally, the enhanced depth maps are fused to obtain the 3D simulated data (Sec. 3.5).

3.1. Preliminaries on Diffusion Models

Our model is based on Denoising Diffusion Probabilistic Models (DDPM) [20, 51]. The key of DDPM is to model a data distribution $p(x)$ via building a Markov chain in the data space. It can be divided into two main processes, forward process and backward process.

Forward diffusion process. Given an initial data x_0, we gradually add random noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to x_0, yielding a series of noisy data, which can be simply formulated as $x_{t+1} = x_t + \boldsymbol{\epsilon}_t$. After adding noise for multiple steps, the initial distribution of $p(x)$ is expected to be transformed into a standard Gaussian distribution.

According to DDPM [20], this process can be directly achieved by the following forward diffusion process:

$$q(x_t) = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}, x_0 \sim p(x), t \in \{0, 1, ..., T\}, \tag{1}$$

where T denotes the number of the time step, t is the current time step, and α_t regulates the noise schedule deciding the pace at which the initial data distribution transforms into a standard Gaussian distribution.

Backward diffusion process. Given a noisy x_t from a standard Gaussian distribution as the initial state, x_t is denoised to a cleaner x_{t-1} at each time step, which can also be simply formulated as $x_{t-1} = x_t - \mu_\theta^{\boldsymbol{\epsilon}}(x_t, t) + \boldsymbol{\epsilon}_t$ in which $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \sigma_\mathbf{t} \mathbf{I})$ and $\mu_\theta^{\boldsymbol{\epsilon}}$ is a learnable network to predict

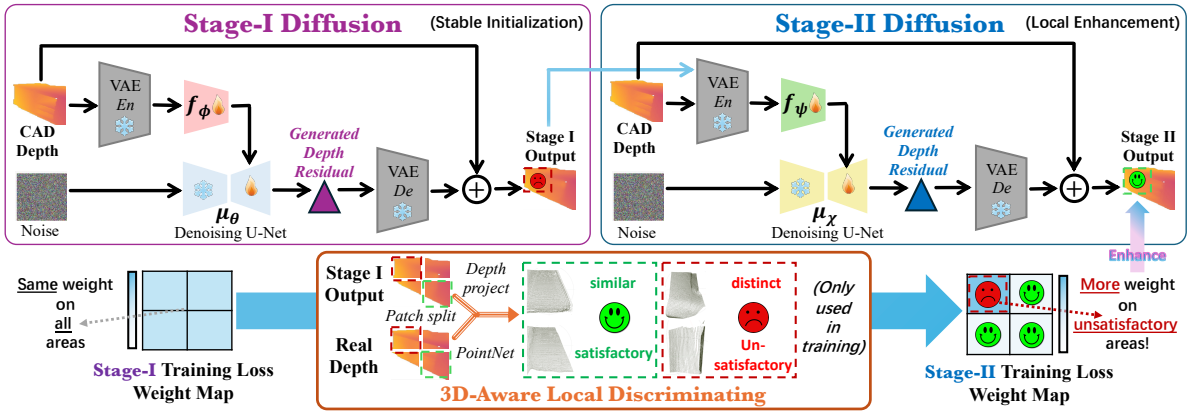

Figure 2. **An overview of our *Stable-Sim2Real***. Conditioned on CAD (*i.e.*, synthetic) depth maps, our model first generates the *residual* (*i.e.*, difference) between real-captured and CAD depth maps. By adding the generated residual to the CAD depth, Stage-I diffusion produces stable but coarse depth maps (Sec. 3.2). Next, the Stage-I output is projected into 3D and sent to a 3D discriminator, to identify the unsatisfactory local areas that are distinct from real-captured patterns (Sec. 3.3). Later, our Stage-II diffusion is conditioned on both CAD and Stage-I output, where the training loss of Stage-II is re-weighted to be specialized on enhancing unsatisfactory areas of Stage-I output (Sec. 3.4). *Notably*, the local discriminating and loss re-weighting are only used during training.

the injected noise. By repeating this reverse process until the maximum number of steps T, we can obtain the final generated data.

In DDPM [20], the backward diffusion process is proven to be:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1-\alpha_t}{\sqrt{\alpha_t(1-\Pi_{\tau=0}^{t}\alpha_\tau)}} \mu_\theta^{\epsilon}(x_t, t;) + \sigma_t \epsilon, \tag{2}$$

where σ_t denotes the standard deviation of $q(x_{t-1}|x_t, x_0)$ in the backward diffusion to sample random noise. The loss for training μ_θ^{ϵ} is:

$$L_\theta = \mathbb{E}_{\epsilon,t} \left\| \epsilon - \mu_\theta{}^{\epsilon}(x_t, t) \right\|^2 \tag{3}$$

3.2. Stage-I: Initial Depth Generation

Residual depth generation with SD. Different from unconditional diffusion models, text-to-image diffusion models focus on generating images with optional text prompts. A representative is Stable Diffusion (SD) [42]. SD trains a denoising U-Net architecture $\mu_\theta(x_t, t, c)$ on the latent space of a pre-trained Variational Auto-Encoder (VAE). Here, c represents the supplementary text prompt embedding, usually acquired through methods like CLIP [41].

We finetune SD model on synthetic-real paired depth data from LASA [31]. Specifically, given an initial synthetic depth map $\boldsymbol{S}$ and its paired real-captured depth map $\boldsymbol{R}$ from a pre-defined dataset, we first get the initial ground truth residual (*i.e.*, difference) depth map $\boldsymbol{D}_{\boldsymbol{res}}^{\boldsymbol{gt}} = \boldsymbol{S} - \boldsymbol{R}$. Then we follow ControlNet [73] to add an additional encoder En and decoder $\boldsymbol{\mu}_\theta{}^{\epsilon}$ for finetuning SD. Specifically, we run the forward process to add noise on $\boldsymbol{D}_{\boldsymbol{res}}^{\boldsymbol{gt}}$ following Eq. (1) and encode with En to obtain x_t. Next, we send $\boldsymbol{S}$ to a pretrained VAE encoder En, generating the latent condition signal. This latent signal $En(\boldsymbol{S})$ is further processed by an additional encoder $\boldsymbol{f}_\phi$ and sent to the decoder blocks of denoising U-Net. We train both the decoder blocks $\boldsymbol{\mu}_\theta{}^{\epsilon}$ and encoder $\boldsymbol{f}_\phi$ using the following loss (based on Eq. (3)):

$$L_{\theta,\phi} = \mathbb{E}_{\epsilon,\boldsymbol{S},t} \left\| \epsilon - \boldsymbol{\mu}_\theta{}^{\epsilon}(\boldsymbol{x}_t, t, \boldsymbol{f}_\phi(En(\boldsymbol{S}))) \right\|^2 \tag{4}$$

Learning depth residual helps to *produce stable depth*. Empirically, this residual represents the noise added to clean (*i.e.*, synthetic) depth during real-world scanning. Moreover, compared to directly generating noisy real depth, adding noise to CAD depth, which is inherently clean and view-consistent, can produce more *stable* depth with smaller view variation and better preservation of the original geometry. From experimental results, learning residual indeed performs better (see Tab. 5).

Learning depth residual is *theoretically identical* to learning real depth. First, assume that real depth is a sample from a Gaussian distribution, denoted by $\boldsymbol{R} \sim \mathcal{N}(\alpha_{\mathbf{R}}, \vartheta)$, where α denotes the mean and ϑ represents the variance. Notably, ϑ can naturally represent real-captured noise. Therefore, synthetic depth can be denoted by $\boldsymbol{S} \sim \mathcal{N}(\alpha_{\mathbf{S}}, 0)$, where the variance is zero as there is no real-captured noise.

In our design, as the diffusion model is inherently designed to learn the transformation from a random noise $\mathcal{N}(\mathbf{0}, \mathbf{I})$ to any other Gaussian distribution [53], we leverage it to model the distribution of the depth residual that can be denoted by $\mathcal{N}(\alpha_{\mathbf{R}} - \alpha_{\mathbf{S}}, \vartheta)$. Last, by adding the depth residual to synthetic depth, the output distribution can be formulated as: $\mathcal{N}(\alpha_{\mathbf{R}} - \alpha_{\mathbf{S}}, \vartheta) + \mathcal{N}(\alpha_{\mathbf{S}}, 0) = \mathcal{N}(\alpha_{\mathbf{R}}, \vartheta)$,

which is exactly identical to the previously assumed real-captured noise distribution.

3.3. 3D-Aware Local Discriminating

Unsatisfactory areas distinct from real patterns. Based on our observations, the noise captured in real data is spatially-*variant*, meaning different areas within a real depth map may display varying levels of perturbation. However, our Stage-I diffusion tends to introduce spatially-*uniform* noise across the entire input CAD depth. This discrepancy may stem from a bias within SD, as also discussed in [9]. As a result, although certain regions of the generated depth map align well with real perturbation patterns, there remain noticeable areas that deviate from reality. This leads to evident 3D geometric discrepancies between Stage-I outputs and real captures (as illustrated in Fig. 6).

Identifying unsatisfactory areas with 3D discriminator. Our design logic to discriminate unsatisfactory areas is that if the generated areas diverge significantly from real patterns, there should exist a *distribution boundary* of their *3D* geometries. To this end, we propose employing a 3D binary classifier to discriminate this boundary. Initially, we split all generated and real-captured depth maps into different patches representing different areas, which are then projected to form 3D point patches. Subsequently, we leverage PointNet [40] as a 3D binary classifier to differentiate whether the input point patches come from model generation or real captures. The CAD depth maps are also concatenated in the input as additional guidance. The *successful* classification of a generated patch indicates that its corresponding area is *distinct* from real captures. *Notably*, the binary classifier is solely used during training.

3.4. Stage-II: Unsatisfactory Area Enhancement

Improving unsatisfactory areas by re-weighting denoising loss. After identifying these local patches, we introduce a second-stage depth refinement model that differs from the Stage-I diffusion in two crucial aspects: 1) The diffusion model is now conditioned on the concatenation of synthetic depth $\boldsymbol{S}$ and the *previously generated* depth residual $\boldsymbol{D}^I$ from Stage I. Here, $\boldsymbol{D}^I = \boldsymbol{D}^I_{res} + \boldsymbol{S}$, where $\boldsymbol{D}^I_{res}$ is the output residual depth from Stage-I model. Thus, the training loss defined in Eq. (4) is updated to:

$$L_{\chi,\psi} = \mathbb{E}_{\boldsymbol{\epsilon},\boldsymbol{D}^I,\boldsymbol{S},t} \left\| \boldsymbol{\epsilon} - \boldsymbol{\mu}_\chi{}^{\boldsymbol{\epsilon}}(\boldsymbol{x}_t, t, \boldsymbol{f}_{\boldsymbol{\psi}}(En(\boldsymbol{D}^I_{res}, \boldsymbol{S}))) \right\|^2, \tag{5}$$

2) More importantly, the diffusion loss is re-weighted to prioritize the enhancement of the identified unsatisfactory regions, making the final Stage-II loss $L_{\chi,\psi}$ to be:

$$L_{\chi,\psi} = \omega L^{sim}_{\chi,\psi} + \lambda L^{dis}_{\chi,\psi}, \tag{6}$$

where $L^{sim}_{\chi,\psi}$ indicates the loss on similar regions, while $L^{dis}_{\chi,\psi}$ denotes the loss on previously identified distinct (*i.e.*, unsatisfactory) areas. ω and λ are weight coefficients, in which $\lambda > \omega$. In our experiment, setting $\omega = 0.5$ and $\lambda = 1.5$ produces the best result.

Our Stage-II diffusion model is ***specialized*** on learning the distribution of unsatisfactory areas in Stage-I generation. Meanwhile, by conditioning on Stage-I output and assigning less weight to satisfied areas, the generation on satisfied outputs is also *preserved* in Stage-II, as in Fig. 6.

3.5. Creating Real-World 3D Data

Two-stage real depth simulation. In the initial stage, we sample a random distribution $\boldsymbol{\epsilon}$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and feed it into the denoising U-Net $\boldsymbol{\mu}_\theta$ of Stage-I diffusion model. The model runs a backward diffusion process for T steps to generate the initial depth residual $\boldsymbol{D}^I_{res}$, following Eq. (2) with the synthetic depth $\boldsymbol{S}$ serving as the condition. Now we can obtain the initial depth map $\boldsymbol{D}^I = \boldsymbol{D}^I_{res} + \boldsymbol{S}$.

Moving to the second stage, we repeat a similar backward diffusion process to generate the residual depth $\boldsymbol{D}^{II}_{res}$, using the denoising U-Net $\boldsymbol{\mu}_\chi$ of our Stage-II diffusion model, conditioned on both synthetic depth $\boldsymbol{S}$ and Stage-I output $\boldsymbol{D}^I$. The final depth map $\boldsymbol{D}^{II} = \boldsymbol{D}^{II}_{res} + \boldsymbol{S}$. DDIM sampler [51] is utilized during the whole inference phase.

Depth fusion to 3D data. Given the 3D synthetic object, we follow LASA [31] to embed CAD objects into 3D scenes. This allows us to render synthetic depth images, simulating the occlusions in real-world captures. Then we conduct the aforementioned two-stage real depth simulation on rendered synthetic depth maps, followed by a depth fusion algorithm to create simulated 3D data.

4. Experiments

We provide a new benchmark scheme for 3D data simulation. To begin with, different simulation methods are performed on synthetic datasets. The 3D simulated data are then utilized to train networks for tackling fundamental real-world 3D visual tasks: 3D instance reconstruction, 3D object/scene understanding. The ***key logic*** of our evaluation is: if the model's real-world performance can be improved after being trained with simulated data, it would validate that the simulated data is *close to real distributions*, demonstrating the effectiveness of simulation methods.

4.1. Preliminaries

Training of Stable-Sim2Real. We train our model by finetuning the official StableDiffusion V2.1 [42]. The training dataset is a recent 3D synthetic-real paired dataset, LASA [31]. LASA is a large-scale dataset that contains 10,412 unique CAD models across 17 categories. Skilled artists are employed to meticulously craft aligned CAD models from 3D object real-world scans of [3]. LASA provides CAD-real depth map pairs, rendered based on camera

Method	Chair	Sofa	Table	Cabinet	Bed	Shelf
w/o sim. data	24.5 / 12.3 / 20.1	55.6 / 7.02 / 19.8	25.8 / 19.3 / 20.3	57.1 / 8.83 / 24.1	40.1 / 6.14 / 17.3	12.8 / 6.74 / 23.7
Cycle-GAN [77]	25.1 / 12.1 / 20.9	55.2 / 6.98 / 20.5	26.1 / 19.1 / 19.5	56.5 / 8.24 / 25.1	40.6 / 5.98 / 16.9	13.1 / 6.08 / 24.9
Pix2Pix [22]	24.9 / 11.0 / 20.5	56.4 / 6.74 / 21.3	26.6 / 18.5 / 22.1	57.4 / 7.83 / 25.5	41.3 / 6.01 / 17.8	14.2 / 6.15 / 26.2
Giga-GAN [25]	26.2 / 10.3 / 22.1	58.9 / 5.87 / 22.4	29.3 / 16.4 / 24.3	60.5 / 7.01 / 26.9	44.2 / 5.25 / 19.3	18.2 / 5.87 / 27.8
Stable-Diffusion [42]	27.3 / 9.51 / 25.5	59.4 / 5.93 / 23.6	29.7 / 15.8 / 24.7	61.6 / 6.92 / 27.5	45.1 / 5.31 / 20.1	**19.1** / 5.85 / 28.3
Ours	**29.2 / 7.83 / 27.5**	**62.9 / 4.09 / 25.1**	**31.2 / 14.1 / 26.8**	**63.2 / 5.58 / 28.6**	**46.3 / 4.23 / 21.8**	18.8 / **5.02 / 31.6**

Table 1. Quantitative comparison of few-shot evaluation on 3D instance reconstruction, where the model is pretrained using simulated data generated by different methods. "w/o sim. data" indicates pretraining using *no* simulated data but only synthetic data. Evaluation metrics are **mIoU / Chamfer L2 / F-score** respectively. Higher mIoU and F-score are better while lower Chamfer L2 is better. Chamfer L2 is scaled by 1,000. All results are reproduced using the official code of DisCo [31].

Method	Chair	Sofa	Table	Cabinet	Bed	Shelf
w/o sim. data	19.4 / 14.7 / 15.8	46.2 / 9.92 / 15.7	20.5 / 23.6 / 15.9	51.2 / 11.3 / 20.2	33.6 / 13.1 / 10.9	11.1 / 9.23 / 21.6
Cycle-GAN [77]	20.3 / 13.1 / 16.5	48.9 / 9.03 / 18.9	23.2 / 22.4 / 17.5	51.9 / 10.8 / 21.4	35.2 / 12.5 / 12.1	11.7 / 8.87 / 23.5
Pix2Pix [22]	19.3 / 12.3 / 17.4	49.7 / 9.51 / 18.8	22.6 / 21.5 / 18.8	52.7 / 11.2 / 21.1	37.7 / 10.0 / 13.8	12.5 / 8.91 / 24.3
Giga-GAN [25]	22.4 / 11.8 / 19.5	53.0 / 7.69 / 19.4	23.1 / 19.3 / 21.6	54.3 / 10.1 / 21.9	40.3 / 7.95 / 15.4	14.2 / 7.81 / 26.9
Stable-Diffusion [42]	23.3 / 11.7 / 20.4	54.1 / 7.98 / 19.2	24.2 / 17.9 / 22.6	55.1 / 9.07 / 22.4	40.4 / 7.89 / 16.2	15.8 / 6.72 / 28.3
Ours	**27.1 / 9.31 / 24.8**	**60.6 / 5.29 / 22.3**	**28.5 / 16.1 / 24.7**	**60.6 / 7.51 / 25.9**	**43.2 / 6.23 / 18.0**	**17.3 / 5.93 / 30.5**

Table 2. Quantitative comparison of zero-shot evaluation on 3D instance reconstruction, when the model is solely pretrained using simulated data generated by different methods.

information during the real-world collection. See the supplementary material for more implementation details.

Methods in comparison. Since our method operates on depth images, we select several image-to-image generation methods for comparison: 1) Cycle-Consistent Adversarial Networks (Cycle-GAN) [77], which is a representative generative network when paired training data are not available, so we train Cycle-GAN on the *pair-shuffled* LASA training data. 2) Pix2Pix [22], which is a classical image generation method. 3) Giga-GAN [25], a state-of-the art GAN-based method. 4) Stable Diffusion [42], our main baseline model. Both 2), 3), 4) are trained using the paired data from LASA.

4.2. 3D Instance Reconstruction

Setup. 3D instance reconstruction aims to recover the full geometry from real-captured partial point clouds or multi-view images. We utilize a recent state-of-the-art diffusion-based reconstruction network, DisCo, which is proposed in LASA [31]. It is originally pretrained on three synthetic datasets (*i.e.*, ShapeNet [8], ABO [10], and 3D-Future [17]) and finetuned on LASA real training data. During pretraining, DisCo simply constructs partial point clouds by adding random perturbation to synthetic data. Instead, in our experiment, to simulate partial point clouds for pretraining, we perform simulation using our method and other baseline models based on the original three synthetic datasets used to pretrain DisCo, and replace half of the original partial point clouds with the simulated 3D data. The other configurations all follow the original DisCo.

Subsequently, we employ few-shot and zero-shot learning configurations to evaluate: We randomly sample only a few LASA real training data to finetune the pretrained network. In this way, the real-world knowledge of the network will *mostly* come from the pretraining data (*i.e.*, the simulated data produced by our method and others) instead of the finetuning data, which can better reflect the performance gain from the simulated data ***itself***. The pretrained model is finetuned with 20∼30% of the original training samples for few-shot learning and is evaluated on the original LASA validation set. For zero-shot learning, we directly evaluate the pretrained network without fine-tuning.

Why not perform simulation on LASA synthetic data and compare the performance using the simulated data with that using LASA paired real GT? Since our StableSim2Real and other baseline techniques are trained on the LASA training set, it is unfeasible to conduct further inference on LASA, as this would not verify the actual model performance but only reflect the performance caused by overfitting. We can also not conduct simulation on a third-party paired dataset similar to LASA for 3D reconstruction, as there is currently *no* such paired dataset.

Quantitative results. Following [31], we utilize mean Intersection over Union (mIoU), L2 chamfer distance, and 1% F-score metrics. Initially, we normalize both the results and the ground-truth CAD objects to fall within the range of -0.5 to 0.5, according to which we calculate the aforementioned metrics between them. As shown in Tab. 1, under all few-shot settings, the model pretrained using the simulated data from our method achieves the best performance, and significantly surpasses the model using no simulated data (*i.e.*, only add random perturbation to synthetic data) for

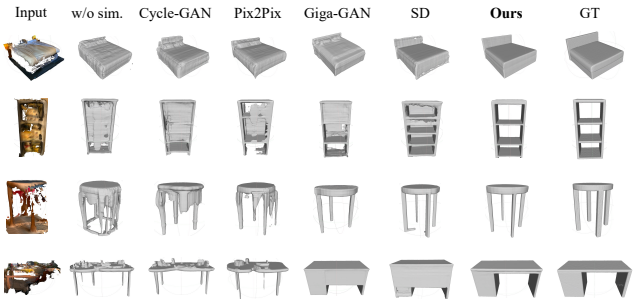

Figure 3. Qualitative comparison of few-shot evaluation on 3D instance reconstruction. "w/o sim." denotes using *no* simulated data for pretraining.

pretraining. Moreover, using the simulated data produced by Cycle-GAN or Pix2Pix does not improve the model performance compared to that using no simulated data, possibly attributed to their inferior simulation quality, introducing ambiguity and complexity during pretraining, making it hard for the model to learn useful real-world knowledge. Moreover, as listed in Tab. 2, under zero-shot setting, when solely using simulated data for pretraining without any finetuning, the model trained with our simulated data demonstrates more conspicuous effectiveness.

Qualitative results. Fig. 3 shows the qualitative comparison, where our simulated data can help the model to gain the best reconstruction *accuracy* and *diversity*. More qualitative results are provided in the supplementary material.

4.3. 3D Object/Scene Understanding

Setup. We select OcCo [63] as our backbone, which is a self-supervised point cloud pretraining framework. OcCo is originally pretrained on ModelNet [68], and finetuned on real-scanned datasets – ScanObjectNN [61] for object classification, and S3DIS [1] for scene semantic segmentation. ModelNet contains 12,311 synthetic CAD objects from 40 classes. ScanObjectNN includes $\sim$15,000 real-scanned objects across 15 categories. S3DIS encompasses 271 indoor rooms and 13 classes. The classification task outputs the class of the whole 3D object, and the semantic segmentation predicts the category of each point in the 3D scene.

In our experiment, we apply our method and other baselines on ModelNet to generate simulated data that are further used to replace half of the original ModelNet synthetic data to pretrain OcCo. Next, the pretrained model is again evaluated under the few-shot learning setting. A typical setting is "K-way N-shot", which indicates that K classes are randomly selected during finetuning, and each category contains N samples. The finetuned model is evaluated on the unseen samples picked from each of the K classes. We select DGCNN [66] as the backbone for OcCo. For few-shot learning, we employ the "5-way, 10-shot" setting. For S3DIS, we follow the original OcCo to report the results using 6-fold cross-validation. All the other configurations follow the original OcCo.

Method	w/o sim.	C-GAN [77]	P2P [22]	G-GAN [25]	SD [42]	**Ours**
Acc. (%)	72.4	72.5	72.1	74.2	74.6	**77.5**
mIoU (%)	47.9	48.0	47.7	49.1	48.9	**51.2**

Table 3. Quantitative comparison of few-shot evaluation on 3D object classification and 3D scene semantic segmentation. "w/o sim" means using *no* simulated data for pretraining.

Method	w/o sim.	C-GAN [77]	P2P [22]	G-GAN [25]	SD [42]	**Ours**
Metrics	59.2/3.54/25.3	60.4/3.21/25.2	61.5/3.02/26.4	62.1/2.86/26.7	62.3/2.83/27.5	**62.9/2.57/28.2**

Table 4. The **mIoU / Chamfer L2 / F-score** between simulated data and real-captures on LASA [31] validation set.

Quatitative results. We report the classification accuracy and segmentation mean Intersection over Union (mIoU) in Tab. 3. The model pretrained using our simulated data gets the best results across both 3D object classification and 3D scene understanding tasks. By pretraining with our simulated data, the model can acquire valuable real-world knowledge that helps downstream tasks. In contrast, pretraining with the low-quality simulated data from Cycle-GAN or Pix2Pix may even degrade the performance.

4.4. Is the simulated data close to real captures?

As previously mentioned, our Stable-Sim2Real is trained on LASA training set. In this section, we aim to evaluate the data simulation quality by directly performing inference and evaluation on LASA validation set. Tab. 4 compares the mIoU, Chamfer L2, and F-score metrics between the simulated data generated by different methods and the real captured Ground Truth (GT). Although not precise, this result can partially prove that our simulated data closely resembles real-world captures. Fig. 4 shows simulation results on depth maps and 3D point clouds. It demonstrates that while our simulated data closely aligns with real-world captures, *subtle differences* persist, highlighting the diversity.

We also provide the simulation results on ShapeNet [8] in Fig. 5. Despite the absence of real-captured Ground Truth (GT) in ShapeNet, the outcomes still show that the simulated results from our output closely resemble real captures, where the generated geometry is not largely broken, with certain realistic perturbations introduced.

4.5. Ablation Studies and Analysis

Module impacts. To evaluate the impact of each key design in our framework, we conduct ablation studies on the 3D instance reconstruction task, specifically on the table category. Similar to Tab. 2, we conduct experiments under zero-shot protocol. This allows for a focused evaluation of the module's impact without the influence of additional factors. Other setups remain consistent with Sec. 4.2.

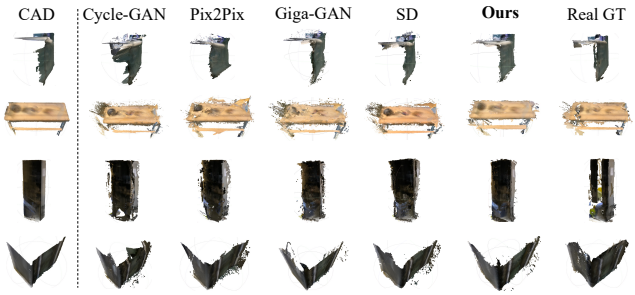

Figure 4. Qualitative simulation results on LASA [31].

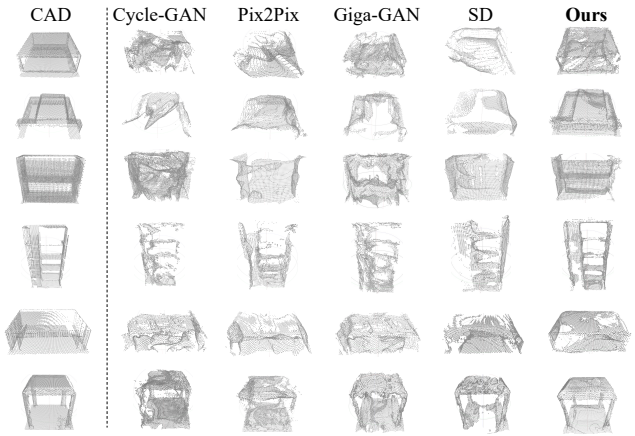

Figure 5. Qualitative simulation results on ShapeNet [8].

1) *Residual generation*. In Tab. 5 (w/o res.), directly generating the real-world depth map without using residual generation degrades the model performance, verifying the efficacy of our residual generation.

2) *Loss re-weighting*. In our Stage-II model, we re-weight the diffusion loss to focus more on unsatisfactory areas. In Tab. 5 (w/o re-w.), the result w/o loss re-weighting is provided, showing its effectiveness.

3) *3D binary discriminator*. Instead of training a 3D binary discriminator, another simple way is to directly set a threshold of the absolute difference between Stage-I generation and real-world GT to decide distinct areas. We tried various difference thresholds and reported the best result in Tab. 5 (w/o bin.), which hurts the model performance. To explain, our output depth is gained by adding *subtle* noise to synthetic depth. Thus, the *magnitude* of the difference between our output and its corresponding real GT is *similar* at 2D/pixel space, where it is impractical to find an appropriate threshold to identify distinct/unsatisfactory areas. Instead, their difference in distribution space is relatively more significant, so using a 3D binary classifier to delineate their ***distribution*** *boundary* is a more feasible way.

4) *Number of stages*. Theoretically, we can iteratively improve the generation quality by adding more stages. However, as shown in Tab. 5, adding Stage-III model does not improve the model performance, and Stage-IV model even slightly degrades the result. To analyze, Stage-II model has been sufficient to identify most of the unsatisfactory areas and improve their generation quality. In addition, adding redundant stages may also cause overfitting issues that hurt the performance.

Method	w/o res.	w/o re-w.	w/o bin.	Stage-III	Stage-IV	**Stage-II**
Metrics	26.4/18.2/22.1	27.1/17.9/23.3	27.3/17.5/23.6	**28.6/ 16.1**/24.5	28.0/16.9/24.1	28.5/**16.1/24.7**

Table 5. Quantitative ablations of zero-shot evaluation on 3D instance reconstruction of table category. Evaluation metrics are **mIoU / Chamfer L2 / F-score** respectively.

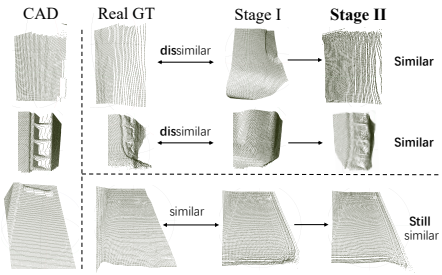

Figure 6. Qualitative simulation results at the patch level.

Analyzing Stage-II model at *3D patch*-level. 1) We compare simulation results with real-captured Ground Truth (GT) at the patch level. As illustrated in Fig. 6, our Stage-II model effectively refines previously unsatisfactory (*i.e.*, dissimilar to GT) patches from Stage-I output to be similar to GT, while keeping previously satisfactory (*i.e.*, similar to GT) patches almost unchanged.

2) When sending the newly generated output patches from Stage-II to our 3D binary discriminator, the classification accuracy notably *drops* from $\mathbf{67.8}\%$ to $\mathbf{32.5}\%$, suggesting that the generated patches successfully *approximate* the real-captured distributions, making it challenging for the classifier to discriminate. This again verifies the effectiveness of our Stage-II model. See the supplementary material for more ablations of the binary classifier.

5. Conclusion

We introduce Stable-Sim2Real, a two-stage depth diffusion model for 3D real data simulation. Initially, our method learns the residual between synthetic and real depth to generate a stable but coarse depth, where some areas may deviate from realistic patterns. Next, both synthetic and initial output depth are fed into the second-stage model, where diffusion loss prioritizes enhancing distinct regions identified by a binary classifier. Experiments show that the 3D simulated data generated from our method can improve the network performance in real-world 3D tasks, and our simulated data is highly similar to real captures. More limitations are discussed in the supplementary material.

Acknowledgement

The work was supported in part by Guangdong S&T Programme with Grant No. 2024B0101030002, the Basic Research Project No. HZQB-KCZYZ-2021067 of Hetao Shenzhen-HK S&T Cooperation Zone, Guangdong Provincial Outstanding Youth Fund (No. 2023B1515020055), No. NSFC 62172348, the Shenzhen Outstanding Talents Training Fund 202002, the Guangdong Research Projects No. 2017ZT07X152 and No. 2019CX01X104, the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001), and the Shenzhen Key Laboratory of Big Data and Artificial Intelligence (Grant No. ZDSYS201707251409055).

References

[1] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, 2016. 1, 7

[2] Kaixin Bai, Lei Zhang, Zhaopeng Chen, Fang Wan, and Jianwei Zhang. Close the sim2real gap via physically-based structured light synthetic data simulation. In *ICRA*, 2024. 1, 3

[3] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Yuri Feigin, Peter Fu, Thomas Gebauer, Daniel Kurz, Tal Dimry, Brandon Joffe, Arik Schwartz, and Elad Shulman. ARKitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile RGB-d data. In *NeurIPS*, 2021. 5

[4] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017. 1, 3

[5] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *ICRA*, 2018. 3

[6] Marc Bresson, Yang Xing, and Weisi Guo. Sim2real: Generative ai to enhance photorealism through domain transfer with gan and seven-chanel-360°-paired-images dataset. *Sensors*, 24(1):94, 2023. 3

[7] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *3DV*, 2017. 1

[8] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1, 6, 7, 8

[9] Hansheng Chen, Ruoxi Shi, Yulin Liu, Bokui Shen, Jiayuan Gu, Gordon Wetzstein, Hao Su, and Leonidas Guibas. Generic 3d diffusion adapter using controlled multi-view editing. *arXiv preprint arXiv:2403.12032*, 2024. 5

[10] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *CVPR*, 2022. 1, 6

[11] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 1

[12] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, 2023. 1

[13] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 3

[14] Yiquan Duan, Xianda Guo, and Zheng Zhu. Diffusiondepth: Diffusion denoising approach for monocular depth estimation. In *ECCV*, 2025. 3

[15] Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *T-RO*, 2023. 1

[16] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *ICCV*, 2021. 1

[17] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *IJCT*, 2021. 6

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 2020. 1, 3

[19] Xiao Gu, Yao Guo, Fani Deligianni, and Guang-Zhong Yang. Coupled real-synthetic domain adaptation for real-world deep depth enhancement. *TIP*, 2020. 1, 3

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2, 3, 4

[21] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022. 3

[22] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 3, 6, 7

[23] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, 2014. 1

[24] Cheongwoong Kang, Wonjoon Chang, and Jaesik Choi. Balanced domain randomization for safe reinforcement learning. *Applied Sciences*, 14(21):9710, 2024. 3

[25] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *CVPR*, 2023. 6, 7

[26] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *CVPR*, 2024. 3

[27] Numair Khan, Min H Kim, and James Tompkin. Differentiable diffusion for dense depth estimation from multi-view images. In *CVPR*, 2021. 3

[28] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *STAT*, 1050:1, 2014. 3

[29] Naveen Kodali, Jacob D. Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017. 1, 3

[30] Michael J Landau, Benjamin Y Choo, and Peter A Beling. Simulating kinect infrared and depth images. *IEEE transactions on cybernetics*, 2015. 1, 3

[31] Haolin Liu, Chongjie Ye, Yinyu Nie, Yingfan He, and Xiaoguang Han. Lasa: Instance reconstruction from real scans using a large-scale aligned shape annotation dataset. In *CVPR*, 2024. 2, 4, 5, 6, 7, 8

[32] Naoki Matsunaga, Masato Ishii, Akio Hayakawa, Kenji Suzuki, and Takuya Narihira. Fine-grained image editing by pixel-wise guidance using diffusion models. *arXiv preprint arXiv:2212.02024*, 2022. 3

[33] Stephan Meister, Rahul Nair, and Daniel Kondermann. Simulation of time-of-flight sensors using global illumination. In *VMV*, 2013. 1, 3

[34] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2021. 3

[35] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021. 3

[36] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021. 3

[37] Alexander Pashevich, Robin Strudel, Igor Kalevatykh, Ivan Laptev, and Cordelia Schmid. Learning to augment synthetic images for sim2real policy transfer. In *IROS*, 2019. 3

[38] Duo Peng, Ping Hu, Qiuhong Ke, and Jun Liu. Diffusion-based image translation with label guidance for domain adaptive semantic segmentation. In *ICCV*, 2023. 3

[39] Benjamin Planche, Ziyan Wu, Kai Ma, Shanhui Sun, Stefan Kluckner, Oliver Lehmann, Terrence Chen, Andreas Hutter, Sergey Zakharov, Harald Kosch, et al. Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition. In *3DV*, 2017. 1, 3

[40] 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. 5

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 4

[42] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1, 2, 3, 4, 5, 6, 7

[43] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, 2023. 3

[44] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *TPAMI*, 2022. 3

[45] Saurabh Saxena, Abhishek Kar, Mohammad Norouzi, and David J Fleet. Monocular depth estimation using diffusion models. *arXiv preprint arXiv:2302.14816*, 2023. 3

[46] Saurabh Saxena, Charles Herrmann, Junhwa Hur, Abhishek Kar, Mohammad Norouzi, Deqing Sun, and David J Fleet. The surprising effectiveness of diffusion models for optical flow and monocular depth estimation. *NeurIPS*, 2024. 3

[47] Yuefan Shen, Yanchao Yang, Youyi Zheng, C Karen Liu, and Leonidas J Guibas. Dcl: Differential contrastive learning for geometry-aware depth synthesis. *IEEE Robotics and Automation Letters*, 2022. 1, 3

[48] Yuan Shi, Huiyun Cao, Bin Xia, Rui Zhu, Qingmin Liao, and Wenming Yang. Dsr-diff: Depth map super-resolution with diffusion model. *Pattern Recognition Letters*, 184:225–231, 2024. 3

[49] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017. 3

[50] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 3

[51] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3, 5

[52] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015. 1

[53] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019. 2, 4

[54] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *NeurIPS*, 2020.

[55] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 2

[56] Chris Sweeney, Greg Izatt, and Russ Tedrake. A supervised approach to predicting noise in depth images. In *ICRA*, 2019. 1, 3

[57] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017. 2

[58] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, et al. Domain randomization and generative models for robotic grasping. In *IROS*, 2018. 2

[59] Fabio Tosi, Pierluigi Zama Ramirez, and Matteo Poggi. Diffusion models for monocular depth estimation: Overcoming challenging conditions. *arXiv preprint arXiv:2407.16698*, 2024. 3

[60] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *CVPRW*, 2018. 2

[61] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, 2019. 7

[62] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In *NeurIPS*, 2021. 2

[63] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matthew J. Kusner. Unsupervised point cloud pre-training via occlusion completion. In *ICCV*, 2021. 7

[64] Jiyuan Wang, Chunyu Lin, Lang Nie, Kang Liao, Shuwei Shao, and Yao Zhao. Digging into contrastive learning for robust depth estimation with diffusion models. In *ACM MM*, 2024. 3

[65] Leichen Wang, Bastian Goldluecke, and Carsten Anklam. L2r gan: Lidar-to-radar translation. In *ACCV*, 2020. 3

[66] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 2019. 7

[67] Zhen Wang, Qiangeng Xu, Feitong Tan, Menglei Chai, Shichen Liu, Rohit Pandey, Sean Fanello, Achuta Kadambi, and Yinda Zhang. Mvdd: Multi-view depth diffusion models. In *ECCV*, 2025. 3

[68] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 1, 7

[69] Mutian Xu, Pei Chen, Haolin Liu, and Xiaoguang Han. Toscene: A large-scale dataset for understanding 3d tabletop scenes. In *ECCV*, 2022. 1

[70] Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Tianyou Liang, Guanying Chen, Shuguang Cui, and Xiaoguang Han. Mvimgnet: A large-scale dataset of multi-view images. In *CVPR*, 2023. 1

[71] Chengjie Yuan, Yunlei Shi, Qian Feng, Chunyang Chang, Michael Liu, Zhaopeng Chen, Alois Christian Knoll, and Jianwei Zhang. Sim-to-real transfer of robotic assembly with visual inputs using cyclegan and force control. In *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1426–1432. IEEE, 2022. 3

[72] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2100–2110, 2019. 3

[73] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 4

[74] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 3

[75] Xiaoshuai Zhang, Rui Chen, Ang Li, Fanbo Xiang, Yuzhe Qin, Jiayuan Gu, Zhan Ling, Minghua Liu, Peiyu Zeng, Songfang Han, Zhiao Huang, Tongzhou Mu, Jing Xu, and Hao Su. Close the optical sensing domain gap by physics-grounded active stereo sensor simulation. *IEEE Transactions on Robotics*, 2023. 1, 3

[76] Xiang Zhang, Bingxin Ke, Hayko Riemenschneider, Nando Metzger, Anton Obukhov, Markus Gross, Konrad Schindler, and Christopher Schroers. Betterdepth: Plug-and-play diffusion refiner for zero-shot monocular depth estimation. *arXiv preprint arXiv:2407.17952*, 2024. 3

[77] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 3, 6, 7