Realiz3D: 3D Generation Made Photorealistic via Domain-Aware Learning

Ido Sobol[1,2] Kihyuk Sohn[2] Yoav Blum[2] Egor Zakharov[2] Max Bluvstein[2] Andrea Vedaldi[2] Or Litany[1]

[1] Technion [2] Meta AI

https://idosobol.github.io/realiz3d/

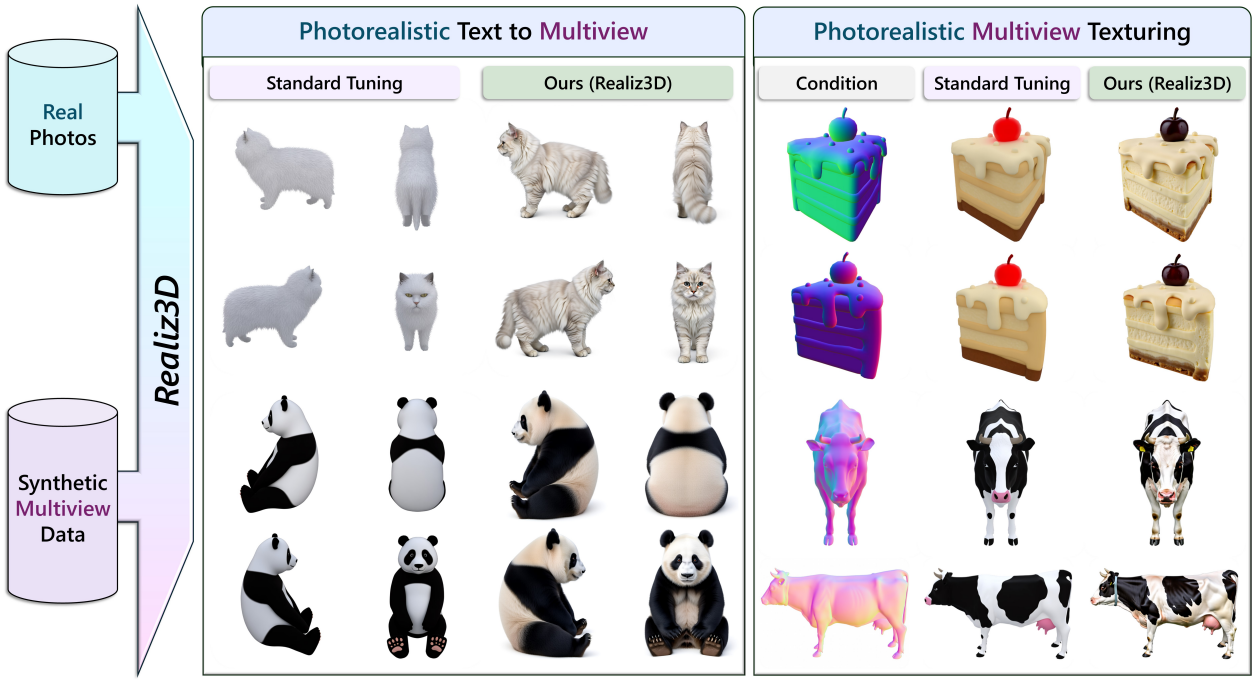

Figure 1. *Realiz3D* is a framework that leverages both real and synthetic data to train diffusion models that generate photorealistic images while faithfully adhering to input conditions and maintaining 3D consistency. Shown are two representative applications: text-to-multiview generation (left) and multiview texturing (right). Compared to standard fine-tuning on mixed real and synthetic data, Realiz3D produces noticeably more realistic results while preserving geometric fidelity across views. The prompts shown are (left) "a fluffy cat", "a panda bear, sitting"; (right) "a slice of creamy cheesecake with a cherry on top", "a cow with black and white patterns". Best viewed zoomed in.

Abstract

We often aim to generate images that are both photorealistic and 3D-consistent*, adhering to precise geometry, material, and viewpoint controls. Typically, this is achieved by fine-tuning an image generator, pre-trained on billions of real images, using renders of synthetic 3D assets, where annotations for control signals are available. While this approach can learn the desired controls, it often compromises the realism of the images due to domain gap between photographs and renders. We observe that this issue largely arises from the model learning an unintended association between the presence of control signals and the synthetic appearance of the images. To address this, we introduce* Realiz3D*, a lightweight framework for training diffusion models, that decouples controls and visual domain. The key idea is to explicitly learn visual domain, real or synthetic, separately from other control signals by introducing a covariate that, fed into small residual adapters, shifts the domain. Then, the generator can be trained to gain controllability, without fitting to specific visual domain. In this way, the model can be guided to produce realistic images even when controls are applied. We enhance control transferability to the real domain by leveraging insights about roles of different layers and denoising steps in diffusion-based generators, informing new training and inference strategies*

that further mitigate the gap. We demonstrate the advantages of Realiz3D in tasks as text-to-multiview generation and texturing from 3D inputs, producing outputs that are 3D-consistent and photorealistic.

1. Introduction

While diffusion-based image generators have made significant progress in recent years, it remains challenging to equip them with *precise 3D controls* to ensure that images conform to prescribed geometry, material, and viewpoint specifications. The latter is important, or even crucial, in many applications. For example, image generators are often used to guide 3D content generation [2, 21, 32, 37, 44] in an attempt to sidestep the lack of large-scale 3D datasets to train such models directly. There is no lack of data to train image generators; however, their application to 3D generation requires the ability to control geometry and viewpoint, which is not natively supported by most image generators.

The main challenge in training image generators with 3D controls is that real images lack 3D-like annotations such as geometry, materials, and cameras. Thus, the usual strategy is to pre-train the model on billions of real images and then fine-tune it on a relatively small number of renders of synthetic 3D assets, for which 3D annotations can be easily obtained. However, such renders are far from photorealistic, resulting in a severe domain gap compared to real images. Ultimately, this leads to an undesirable trade-off between *realism* learned from real images and *controllability* learned from synthetic 3D data. We study this problem and identify a key cause for the degradation in realism: when fine-tuning on synthetic data, the model tends to associate the presence of 3D controls with the synthetic look of the corresponding images. In other words, the control signals *leak domain identity*, so that, when controls are given at inference, the model *also* makes the image look synthetic.

To address this problem, we introduce Realiz3D, a lightweight framework for fine-tuning image generators with 3D control, while preserving photorealism. We do so by decoupling domain identity from control signals. In the first stage, before learning the desired 3D controls, we learn a separate binary control for visual domains, indicating the model to operate in real or synthetic mode. Domain signal is integrated through Domain Shifters, lightweight residuals that shift the generation toward the desired visual domain.

After the first stage, we introduce the desired 3D control signals (e.g., one or multiple viewpoints, normal maps, or other cues), using the synthetic samples for supervision. Although 3D annotations are available only for synthetic data, the Domain Shifter have already learned to disentangle visual domains as a co-variate introduced before, reducing domain leakage. At inference, we can operate in the "real mode", while providing the model with 3D control signals, yielding results that are both realistic and controllable.

To achieve effective transfer of control to the real domain, while maintaining realism, we leverage insights into the roles of different network layers and denoising steps in diffusion-based image generators. As observed before [16, 23, 38], early network layers and denoising steps predominantly determine the structure of the generated image, whereas later layers and steps determine its detailed appearance. Realiz3D exploits this by allowing synthetic data to influence early layers and denoising steps more strongly, while real data has a stronger effect on later ones.

Together, these stages encourage domain-agnostic behavior, enabling effective transfer of control to real domain.

To summarize, our contributions are threefold:

1. A flexible and general recipe for tuning diffusion models on controllable yet domain-shifted datasets, while maintaining the realistic prior of the base model.
2. A new domain-shifting adapter design that separates domain identity from control signals and prevents domain leakage during fine-tuning.
3. A layer-aware training and sampling strategy that progressively unifies feature spaces, enabling realistic and controllable generation for tasks such as text-to-multiview and texturing from 3D inputs.

2. Related Work

Control in Image and 3D Generation. While the quality of image generators has improved dramatically in recent years, control is equally important in applications. Many have thus sought to augment image generators with control signals like depth or normal maps, semantic masks, camera viewpoints, or human poses, to enable conditional generation. These methods typically inject control into pre-trained models and fine-tune to achieve controllability [15, 17, 25, 45]. Learning 3D controls (e.g., depth maps, normals, or multiple viewpoints) [19, 22, 31, 32, 41] requires data annotated with this information, which is difficult to obtain in the real world. Hence, authors often use synthetic data, for example by rendering assets in large-scale 3D model collections like Objaverse [6, 7]. However, fine-tuning a model on synthetic data can affect realism, 'forgetting' the look of real images. Various approaches were proposed to mitigate forgetting, including LoRA layers [14], adapters and ControlNet modules [45], or simply by 'replaying' real data while fine-tuning on synthetic data [32].

Training Adapters To Mitigate domain Gaps. *Wonder3D* [22] jointly generates multiview RGB images and corresponding normal maps by introducing a domain switcher that modifies the model's existing conditioning mechanism. A 1D domain vector is concatenated with the timestep embedding and learned jointly with the model. However, this method does not explicitly enforce consistency between the generated image and its normal map, re-

lying instead on synthetic paired data and cross-domain attention. Similarly, we learn domain embeddings to guide the model towards one of multiple domains. Unlike [22], we neither modify the existing conditioning mechanism nor rely on paired data. Moreover, since both realistic and synthetic domains are well represented in T2I models, jointly training the adapters with the model may cause it to collapse into two modes (controllable and synthetic, vs realistic and uncontrollable). *AnimateDiff* [12] adapts T2I models for video generation using video data, which is often lower quality than image datasets. They train domain adapters, implemented as LoRA layers, to first fit the noisy domain using video frames, then freeze the adapters while training the model on videos. The adapters are removed at inference. We also use a multi-stage training procedure, but fitting an adapter to the synthetic domain is ineffective, as the "undesired" domain is already encoded in the base model's weights. *Still-Moving* [4] trains temporal attention blocks to adapt a T2I model for video generation, and then reuses them in a customized T2I model. To align the temporal blocks' outputs with the model's distribution, they introduce Spatial Adapters implemented as linear projections.

3. Diffusion Models and Domain Gaps

We consider image generators based on denoising diffusion [34] and summarize key findings from the literature.

Timesteps and Domain Gaps. In denoising diffusion models, sampling evolves through timesteps $t = T, T-1, \ldots, 0$. Starting from $X_T \sim \mathcal{N}(0, I)$, a neural network Φ iteratively denoises X_t at each timestep to generate a clean data sample X_0. Previous studies [24, 27, 33, 42, 43] show that early timesteps ($t \approx T$) primarily establish low-frequency structure of generated samples, while later timesteps ($t \approx 0$) determine high-frequency details. Formally, consider two marginal distributions $q(X_t^{\text{real}})$ and $q(X_t^{\text{syn}})$ obtained by noising real and synthetic distributions $p(X^{\text{real}})$ and $p(X^{\text{syn}})$. In the limit of $t = T$, both distributions converge to the same Gaussian distribution, thus are equal. SDEdit [24] adds noise to a non-realistic image, and denoises with a pre-trained diffusion model, to produce a realistic image that preserves the structure. [5] shows that noisy data can be used for training at early timesteps.

Layers and Domain Gaps. The level of details across generation in diffusion models is not only linked to the timestep t, but also to the layers of the underlying denoising neural network. [16, 23, 36, 38] have studied the feature maps computed by different layers of UNet-based diffusion models. UNet [30] is a hierarchical encoder-decoder architecture with skip connections. The encoder extracts progressively coarser structures, while the decoder upsamples and combines features to integrate both coarse and fine-grained information in the output. [38] shows that low-resolution UNet features capture rough 2D shapes and low-frequency patterns, while high-resolution features encode textures and fine details. [23] demonstrates that feature maps capture progressively finer details as denoising advances. Others have noted similar patterns in denoising diffusion transformers [16], and in vision transformers in general [1, 11].

In this work, we leverage these insights by enforcing 3D controls in earlier layers of a *diffusion transformer* [26], while allowing deeper layers to maintain realism.

4. Method

Problem Formulation. We aim to train a controllable diffusion model capable of generating $V \geq 1$ photorealistic and 3D-consistent views $\{x^v_{\text{real}}\}_{v=1}^V$, conditioned on one or more spatial control signals c, such as per-view normal or depth maps. Formally, the model learns the conditional distribution $q_\theta\left(\{x^v_{\text{real}}\}_{v=1}^V \mid c\right)$ that generates realistic and controlled samples, geometrically consistent across views.

To achieve this, we assume access to a text conditioned image generator $q_\psi\left(x_{\text{real}}\right)$ pre-trained on real images and two complementary data sources: a *synthetic* dataset $\{\{x^v_{\text{syn}}\}_{v=1}^V, c\}$, rendered from 3D assets that provide accurate supervision for the control signal c (e.g., camera pose, normals); and a *real* dataset $\{x_{\text{real}}, \varnothing\}$, composed of diverse natural images with null control signal $c = \varnothing$.

Method Overview. We train an image generator for multi-view synthesis by extending its single-view output to a grid representation [2, 31], where multiple views are spatially tiled, and self-attention operates between all views. For real data, we form grids of arbitrary images and restrict attention to operate within each view (*single-image mode*) [32].

The naïve approach of fine-tuning q_θ on synthetic data $\{\{x^v_{\text{syn}}\}_{v=1}^V, c\}$ alone can successfully learn the control signal c, but may *catastrophically forget* the appearance of realistic images due to overfitting to synthetic images. Mixed-domain training, which mixes the real data without control signal with the synthetic data, mitigates, but does not fully resolve the forgetting issue, as shown in Tab. 1. We hypothesize that, since only the synthetic samples carry non-null control, the model implicitly associates the very presence of $c \neq \varnothing$ with the synthetic domain, causing leakage of synthetic appearance whenever control is applied.

To address this issue, we explicitly separate domain identity from the control signal by introducing a co-variate $e_{\text{domain}} \in \{e_{\text{real}}, e_{\text{syn}}\}$, injected into the model via our *Domain Shifters* (Fig. 2). In *stage 1*, we freeze the diffusion backbone and train only the Domain Shifters to distinguish between real and synthetic data under null control, learning $q_\theta(x \mid e_{\text{domain}}, \varnothing)$, so that the model internalizes the notion of domain *independently of control*. In *stage 2*, we introduce control conditioning (available only for synthetic data) and fine-tune the shared backbone to follow it, modeling $q_\theta(\{x^v\}_{v=1}^V \mid e_{\text{domain}}, c)$. Utilizing the Domain Shifters,

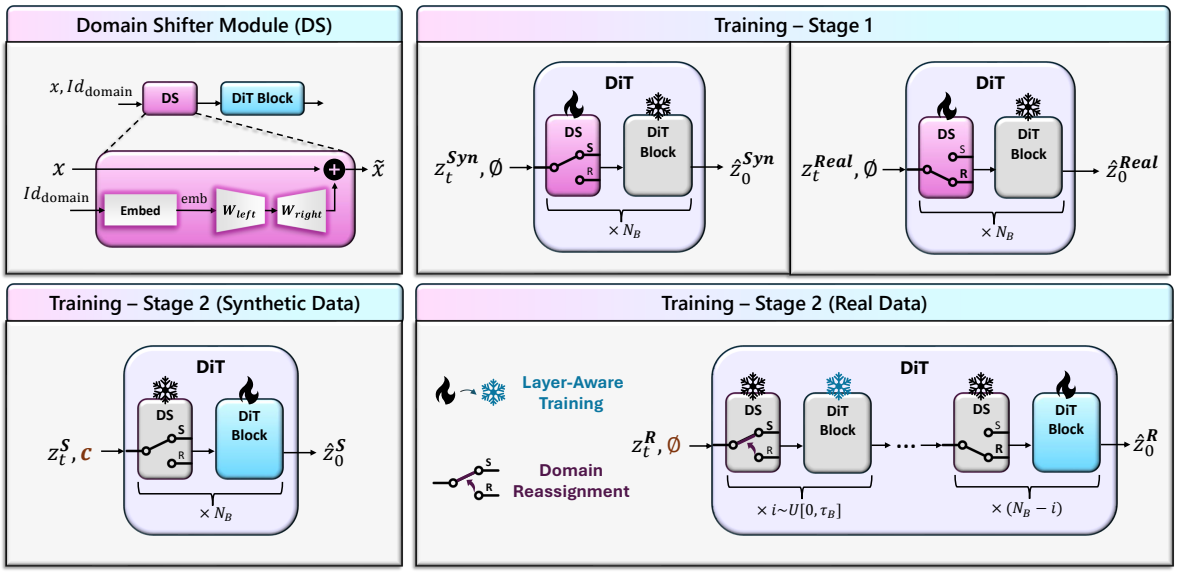

Figure 2. **Method overview.** Realiz3D introduces Domain Shifters, lightweight residual adapters that learn visual domain identity (real vs. synthetic) independently of control signals, enabling the model to learn controllability without compromising realism. **(top left)** A Domain Shifter encodes domain identity as a low-rank residual added to latent features (Sec. 4.1). **(top right)** Stage 1: Domain Shifters are trained with mixed real and synthetic data under null control while the base model is frozen, learning domain separation (Sec. 4.1). **(bottom)** Stage 2: The diffusion model is fine-tuned for controllable generation using both domains (Sec. 4.2). **(bottom left)** Synthetic samples teach controllability under the synthetic mode. **(bottom right)** Real samples (without control) are used for *Representation Binding*—combining (1) *Layer-Aware Training*, which freezes some early layers to preserve structure while updating later, appearance-related ones, and (2) *Domain Reassignment*, which occasionally reuses the synthetic mode in early layers to transfer control to the real domain. Throughout all stages of training, we rely solely on the standard diffusion loss

we propose a *Representation Binding* strategy that ensures that controllability learned on synthetic data transfers effectively to the realistic domain.

Throughout the training, *we rely solely on the standard diffusion loss*. The diffusion objective is used to train our Domain Shifters in Stage 1 and the DiT backbone in Stage 2. *At inference time*, setting $e_{\text{domain}} = e_{\text{real}}$ with $c \neq \varnothing$ enables the model to generate realistic yet controllable images.

4.1. Decoupling Domain from Control with Domain Shifters (Stage 1)

Given an image generator based on denoising diffusion, we denote by $X \in \mathbb{R}^d$ the latent representation entering a diffusion block. A *Domain Shifter* module $\mathcal{D}$ consists of two learnable domain embeddings, $e_{\text{syn}}, e_{\text{real}} \in \mathbb{R}^d$, and a shared low-rank transformation that maps these embeddings into the model's latent space by applying a domain-specific residual adapter (See top row in Fig. 2):

$$\tilde{X} = X + \mathcal{D}(\text{domain}) \;=\; X + W_{\text{left}} W_{\text{right}}\, e_{\text{domain}},$$

where $W_{\text{left}} \in \mathbb{R}^{d\times r}$ and $W_{\text{right}} \in \mathbb{R}^{r\times d}$ define a rank-r mapping with $r \ll d$.

The embedding is added to all tokens within the block, acting as a low-rank bias that modulates activations according to domain identity. Analogous to LoRA-style adapters, this low-rank residual provides sufficient capacity to traverse nearby modes in latent space [29] while maintaining stability and efficiency. In stage 1 (Fig. 2, top right) we freeze the diffusion backbone and optimize only Domain Shifters using both real and synthetic images with null control $c = \varnothing$. We operate in single-image mode, restricting attention to operate within each view only. Since both domains already reside within the pre-trained model's feature space, these lightweight low-rank residuals suffice to capture domain identity explicitly (further discussion is in the Appendix). Then, the model cleanly separates visual domains from control signals, laying the foundation for controllable cross-domain generation in stage 2.

4.2. Fine-tuning with Representation Binding (Stage 2)

Backbone Fine-Tuning. To gain controllability without hindering realism, in stage 2 we propose a strategy for fine-tuning the diffusion backbone while keeping the Domain Shifters frozen. A straightforward approach is to fine-tune the backbone with synthetic data only, while switching the Domain Shifters to synthetic mode, relying on the shared backbone to transfer controllability to real images.

At generation, however, we observe that when switching Domain Shifters to real mode, control signal is not always

respected, and generated samples may still appear synthetic. We attribute this to: (1) *Forgetting of realism*: the backbone drifts toward synthetic statistics since fine-tuning updates are applied only to synthetic data (see ablation 2 in Tab. 2); and (2) *Partial control transfer*: control transferability is merely emergent. Without access to samples with both $e_{\text{domain}} = e_{\text{real}}$ and $c \neq \varnothing$, the shared model lacks experience applying control under real-domain conditions again fitting to the synthetic distribution.

Both factors highlight the need for domain-agnostic behavior in the shared model. To that end, we reintroduce real data for training during Stage 2, and propose a strategy that leverages the model's internal feature hierarchy to enable robust transfer of control to the real domain.

Bridging Unpaired Domains through Feature Space. To address both challenges, we observe that early diffusion layers tend to be domain-agnostic: capturing coarse structure and low-frequency content, shared across real and synthetic images (Sec. 3). Later layers, in contrast, refine high-frequency appearance, where domain gap is more pronounced. By leveraging early layers as a bridge, we can explicitly bind both domains in feature space, promoting transfer of controllability from synthetic to real data while preserving visual fidelity. Building on this, we introduce two complementary strategies, shown in Fig. 2 (bottom), that operationalize this principle: one preserves realism, and another enhances control transferability to real domain.
(1) Preserving Realism with Layer-Aware Training. To prevent forgetting of realism, we incorporate real samples into training, inspired by [32]. However, since real images lack explicit control supervision, naïvely training on them could interfere with the model's ability to respect the control signal learned from synthetic data. Guided by our observation that early layers are largely domain-agnostic and structure related, we update the model with real samples only in the later diffusion blocks, those primarily responsible for appearance refinement, while keeping early blocks frozen. This ensures that training on real data does not disturb the control-related representations formed in early layers.

When processing real samples, Domain Shifters operate in real mode, allowing the model to maintain realistic appearance statistics without altering the shared structural pathway. Concretely, during each real-data training iteration, we freeze DiT blocks $B \in [0, B_i]$, where i is an integer block index randomly drawn from $[0, \tau_B]$ (see Fig. 2, bottom right). This stochastic layer-freezing regularizes early representations, without requiring a fixed cutoff.
(2) Enhancing Control Transferability via Domain Reassignment. To further promote control transfer, we introduce *Domain Reassignment*. With probability p_B, we reassign early DiT blocks ($B \in [0, B_j]$, j is an integer, sampled from $[0, \tau_B]$) to operate in synthetic mode even when processing real samples; that is, we substitute $e_{\text{domain}} \leftarrow e_{\text{syn}}$ in the corresponding Domain Shifters (Fig. 2, bottom right). This asymmetric design integrates real samples into the synthetic feature space, rather than the other way around, since the synthetic domain is the one endowed with explicit control supervision. Consequently, early layers learn shared structural representations that carry controllability, while later layers remain anchored to real-domain appearance.
These components, shown in Fig. 2, form our *Representation Binding* strategy: a soft feature-space alignment, preserving realism while encouraging control transfer.

4.3. Inference-time Domain Shifting

At inference time, thanks to the control transfer established during fine-tuning, one can simply switch the domain adapter to the *real* mode ($e_{\text{domain}} = e_{\text{real}}$) and provide a control condition $c \neq \varnothing$. The model then generates outputs that are both realistic in appearance and faithful to the specified control. *Yet, we can do even better.* While this setup already enables controllable generation in real domain, we find that the control signal can be further strengthened without sacrificing realism. As noted in Sec. 4.2, samples generated with $e_{\text{domain}} = e_{\text{syn}}$ tend to follow the control c more faithfully, as the synthetic domain is *directly* supervised for control.

At inference, we adopt a partial, non-stochastic *domain reassignment*: pre-defined selected early layers and timesteps are set to synthetic mode ($e_{\text{domain}} = e_{\text{syn}}$), while later layers and timesteps remain in real mode. The configuration is tuned once, and not per example. As early layers primarily capture coarse, domain-agnostic structure, this hybrid configuration allows users to rebalance realism and controllability at test time without additional training. Further details and tuning strategy appear in the Appendix.

5. Experiments

We demonstrate the effectiveness of Realiz3D on Multiview Texturing and Text-to-Multiview Generation.

Datasets. *Synthetic Data:* We use an internal dataset of 120K synthetic 3D assets, with their textual descriptions. Each asset is rendered from $V = 4$ viewpoints, with normal and position maps. *Real Data:* While we could use the training data of the base model, we simply use images generated by the base model itself. We use the textual descriptions from the synthetic dataset as prompts, ensuring fairness, and generate V photorealistic images per prompt with white background. The synthetic and real datasets are matched in size. *Evaluation Data:* To evaluate our method, we use 40 3D objects, from Sketchfab, used and reported in [2], along with prompts describing the original object and texture. We create synthetic data (via rendering) and realistic data (by generating realistic images with the T2I model and text prompts) for the evaluation objects.

Implementation Details. We leverage an internal pretrained text-to-image diffusion transformer, whose architec-

ture follows [26] and performance is comparable to other diffusion transformers such as Stable Diffusion 3 [9]. For all tasks, we fine-tune the model to generate a 2×2 image grid of $V = 4$ orthogonal views, each at 512×512 resolution. Full implementation details are in the Appendix.

Baselines. We compare Realiz3D to three categories of adaptation techniques, trained under identical settings using the same base model and data. Then, we evaluate Realiz3D against pretrained models, producing multiview images from text. While we do not have access to their training data, they serve as reference points for assessing Realiz3D relative to state-of-the-art performance.

Full Fine-Tuning: All base model's parameters are trained. (1) Using synthetic samples only (Syn only). (2) Using synthetic and realistic samples [32], evenly (Syn + Real).

Lightweight Fine-Tuning: Only a small set of parameters is trained to retain the base model's prior, using synthetic data only. (3) Training LoRA layers [14], added to each transform in all attention layers. (4) Training linear adapter layers, added before each DiT block, implemented as the product of two low-rank matrices, inspired by [25].

Adapter-based Methods: We evaluate methods, discussed in Sec. 2, that propose adapters to bridge domain gaps, and adjust them as necessary to tackle synthetic-to-real adaptation. (5) Domain Adapters [12]. (6) Spatial Adapter [4], applied similarly to [12] using linear adapter layers. For (3)-(6) we test two common ranks: 32 and 128, and show results with rank 32. (7) Two variants of Domain Switcher [22]: one with the switcher trained jointly with the model [22], and another trained separately in a two-stage manner.

Training-Free Methods: We evaluate SDEdit [24] for multiview applications, using $t = 500$ (See Appendix).

Pretrained (Text-to-Multiview). We evaluate the realism of TRELLIS [40], a recent 3D-native model.

5.1. Multiview Texturing

Task Definition. Given a set of $V \geq 1$ aligned normal maps $\{c_{normal}\}_0^V$ and position maps $\{c_{position}\}_0^V$, our goal is to generate 3D-consistent images $\{x_{RGB}\}_0^V$ matching the given geometry, thus learning the conditional distribution $q(\{x_{RGB}\}_0^V | \{c_{normal}\}_0^V, \{c_{position}\}_0^V)$.

Implementation Details: Following previous works, normal and position maps are encoded using the model's VAE, and are channel-concatenated to the noisy latent.

Metrics. *3D Consistency:* Following prior works, we back-project the generated images onto their mesh and project the same views, reporting PSNR, SSIM [39] and LPIPS [46] between generated and re-projected views.

Prior Preservation: We compare our generated views to a set of realistic images generated by the base T2I model using the same prompts, and report FID_B and KID_B.

Real-World Realism: Prior preservation metrics assess realism but rely on synthesized data. We compare our generated views to real-world images from ImageNet [8], selected from categories best fitting the evaluation objects (further details in the Appendix). We report FID_I and KID_I.

Text-Image Alignment: We report CLIP score. The prompts describe the original texture, and do not require realism.

Quantitative Evaluation. Main results in Tab. 1 demonstrate that control and realism typically trade off. Realiz3D uniquely achieves strong performance in both. Still, while Realiz3D achieves strong results with notably improved realism, its adherence to control remains slightly lower than fully synthetic baselines. We attribute this to: (1) 3D consistency metrics are sensitive to fine-grained details—typically absent in synthetic data—that require precise pixel-level consistency; (2) Lack of realism can occasionally manifest in unrealistic geometry, causing Realiz3D to deviate from the signal in favor of photorealism; (3) Inconsistent appearance may result from the base model's lighting bias.

Qualitative Evaluation. We present visual examples of generated textures in Fig. 1 and Fig. 3. Only the best-performing baselines are shown in the main paper. Full and additional results appear in the Appendix. To showcase 3D consistency, meshes textured with our generated images appear on our project page.

Ablation Study. We firstly verify the importance of our two-stage training, and of using real data at stage 2. Additionally, to assess each component's contribution, we evaluate Realiz3D by gradually adding them, showing that each enhances controllability with minimal compromise to realism, and that all together yield the best overall performance. Full results of the ablation study are reported in Tab. 2.

5.2. Text-to-Multiview Generation

Task Definition. Given a set of $V \geq 1$ camera viewpoints $\{c_{view}\}_0^V$, our goal is to generate 3D-consistent images $\{x_{RGB}\}_0^V$ matching the given views, thus learning the conditional distribution $q(\{x_{RGB}\}_0^V | \{c_{view}\}_0^V)$.

Implementation Details: We generate images in a grid of four fixed viewpoints, where camera viewpoint information is injected via positional encoding into the latent samples.

Metrics. *3D Consistency:* We lift our generated multiview to 3D using a pre-trained LGM model [37]. We measure reprojection error and report PSNR, SSIM and LPIPS. *Prior Preservation:* FID_B and KID_B. *Real-World Realism:* FID_I and KID_I. *Text-Image Alignment:* CLIP score.

Quantitative Evaluation. Main results appear in Tab. 3, using the prompts from the *Evaluation Data* described earlier. While Realiz3D significantly improves realism, its 3D consistency remains slightly lower than fully synthetic baselines. As with texturing, 3D consistency is sensitive to fine-grained details, and lighting bias in the base model may cause inconsistencies. For pretrained models, we report realism-related metrics in Tab. 3. Their performance is comparable or worse than our fully synthetic baseline, vali-

Table 1. **Multiview Texturing: Quantitative Results.** Realiz3D achieves significantly improved realism while maintaining 3D consistency comparable to the synthetic-only baseline (top row).

	Method	3D Consistency			Prior Preservation		Real-World Realism		Text Align
		PSNR ↑	SSIM ↑	LPIPS ↓	FID_B ↓	KID_B ↓	FID_I ↓	KID_I ↓	CLIP ↑
Full Tuning	Syn Only	**25.76**	**0.9269**	**0.0831**	168.21	0.0240	218.29	0.0431	0.2628
	Syn + Real	25.63	0.9260	0.0833	164.37	0.0226	214.84	0.0411	0.2629
Light Tuning	LoRA (rank 32 / 128)	21.88 / 22.12	0.9012 / 0.9033	0.1027 / 0.1014	151.97 / 155.65	0.0157 / 0.0168	204.80 / 207.69	0.0324 / 0.0339	**0.2682 / 0.2680**
	Linear Adapters (rank 32 / 128)	22.49 / 22.87	0.9069 / 0.9088	0.0959 / 0.0967	154.16 / 156.07	0.0186 / 0.0189	210.44 / 213.50	0.0372 / 0.0379	0.2678 / 0.2671
Train Free	SDEdit	22.21	0.9013	0.1169	147.06	0.0149	204.35	0.0327	0.2672
Adapters	Domain Adapter (rank 32 / 128)	25.61 / 25.60	0.9266 / 0.9264	0.0843 / 0.0848	164.17 / 163.64	0.0223 / 0.0220	215.80 / 215.28	0.0408 / 0.0398	0.2610 / 0.2608
	Spatial Adapter (rank 32 / 128)	24.91 / 24.86	0.9200 / 0.9198	0.0858 / 0.0863	155.71 / 153.35	0.0197 / 0.0181	210.62 / 208.64	0.0393 / 0.0343	0.2634 / 0.2636
	Domain Switcher (2-Stage)	24.94	0.9193	0.0888	157.89	0.0185	210.18	0.0350	0.2644
	Domain Switcher (Joint)	23.16	0.9069	0.0991	145.48	0.0156	205.08	0.0335	0.2663
	Ours	24.78	0.9153	0.0865	**141.90**	**0.0121**	**200.24**	**0.0291**	0.2674

Table 2. **Ablation Study.** (top rows) The impact of our two-stage training and using real data at Stage 2. (bottom rows) The contribution of Realiz3D components: Layer-Aware Training (LA Train), Domain Reassignment (Reassign), and inference shifting (Sampling).

#	Method	Training	3D Consistency			Prior Preservation		Real-World Realism		Text Align
			PSNR ↑	SSIM ↑	LPIPS ↓	FID_B ↓	KID_B ↓	FID_I ↓	KID_I ↓	CLIP ↑
(1)	DiT + DS	*Joint*	25.53	0.9267	0.0850	166.93	0.0234	216.63	0.0424	0.2624
(2)	DiT + DS (w/o real data at Stage 2)	2-Stage	25.11	0.9232	0.0849	162.02	0.0220	212.86	0.0395	0.2656
(3)	DiT + DS	2-Stage	23.97	0.9123	0.0897	137.23	0.0098	198.44	0.0258	0.2695
(4)	DiT + DS + Sampling	2-Stage	24.25	0.9128	0.0883	141.99	0.0123	202.07	0.0293	0.2679
(5)	DiT + DS + Reassign	2-Stage	24.18	0.9125	0.0877	139.21	0.0102	198.71	0.0265	0.2678
(6)	DiT + DS + Reassig + Sampling	2-Stage	24.37	0.9130	0.0871	141.08	0.0117	199.33	0.0289	0.2670
(7)	DiT + DS + LA Train + Reassign	2-Stage	24.52	0.9141	0.0870	141.20	0.0118	199.39	0.0290	0.2675
(8)	Ours (DS + LA Train + Reassign + Sampling)	2-Stage	24.78	0.9153	0.0865	141.90	0.0121	200.24	0.0291	0.2674

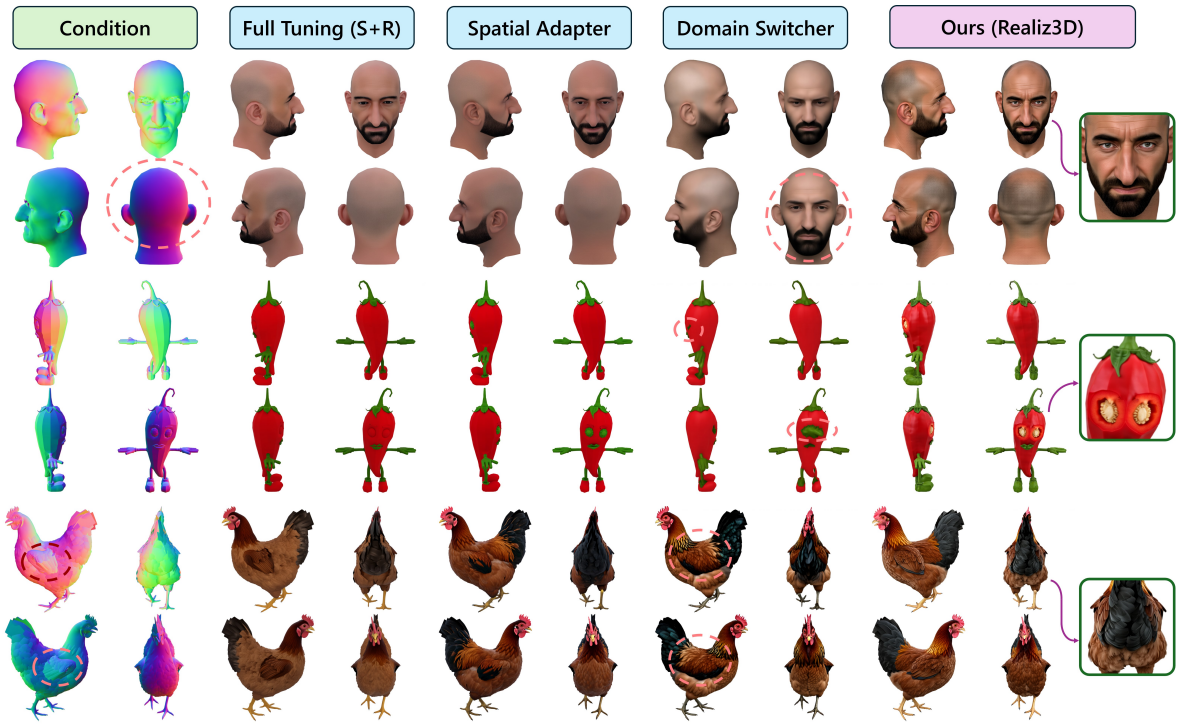

Figure 3. **Multiview Texturing: Qualitative Results.** Prompts: "A man with dark, short beard", "A standing red pepper", "A brown barn chicken", respectively. All prompts are appended with "highly photorealistic and detailed". Red, dashed circles highlight inconsistent regions (either with the geometry or with other views). Realiz3D achieves significant improvements in photorealism while remaining 3D-consistent and faithful to the geometric conditions. Only normal maps are shown due to limited space. Best viewed zoomed in.

Table 3. **Text-to-Multiview Generation: Quantitative Results.** Realiz3D achieves significantly improved realism while maintaining 3D consistency comparable to the synthetic-only baseline (top row).

	Method	3D Consistency			Prior Preservation		Real-World Realism		Text Align
		PSNR ↑	SSIM ↑	LPIPS ↓	FID_B ↓	KID_B ↓	FID_I ↓	KID_I ↓	CLIP ↑
Full Tuning	Syn Only	**19.66**	**0.8779**	**0.0964**	168.60	0.0204	215.57	0.0363	0.2541
	Syn + Real	19.37	0.8714	0.1017	164.44	0.0192	214.72	0.0361	0.2586
Light Tuning	LoRA (rank 32 / 128)	17.54 / 17.91	0.8548 / 0.8568	0.1222 / 0.1186	153.89 / 154.11	0.0125 / 0.0131	204.14 / 206.68	0.0264 / 0.0269	0.2563 / 0.2579
	Linear Adapters (rank 32 / 128)	18.07 / 18.38	0.8603 / 0.8690	0.1223 / 0.1191	152.21 / 155.40	0.0127 / 0.0150	204.02 / 208.78	0.0256 / 0.03133	0.2583 / 0.2526
Train Free	SDEdit	18.46	0.8431	0.1232	139.18	0.0112	199.67	0.0263	0.2609
Pretrained	TRELLIS (large)	-	-	-	181.92	0.0275	224.22	0.0441	0.2495
Adapters	Domain Adapter (rank 32 / 128)	19.27 / 19.02	0.8655 / 0.8621	0.1076 / 0.1082	158.52 / 157.45	0.0164 / 0.0163	212.00 / 210.67	0.0308 / 0.0313	0.2607 / 0.2601
	Spatial Adapter (Rank 32 / 128)	18.48 / 18.36	0.8422 / 0.8350	0.1216 / 0.1234	132.48 / 130.25	0.0072 / 0.0071	200.38 / 199.46	0.0212 / 0.0205	0.2633 / 0.2638
	Domain Switcher (2-Stage)	18.56	0.8481	0.1134	153.36	0.0146	208.14	0.0303	0.2587
	Domain Switcher (Joint)	17.67	0.8010	0.1290	129.56	0.0061	197.11	0.0180	0.2629
	Ours	19.02	0.8631	0.1075	**122.01**	**0.0056**	**196.01**	**0.0171**	**0.2643**

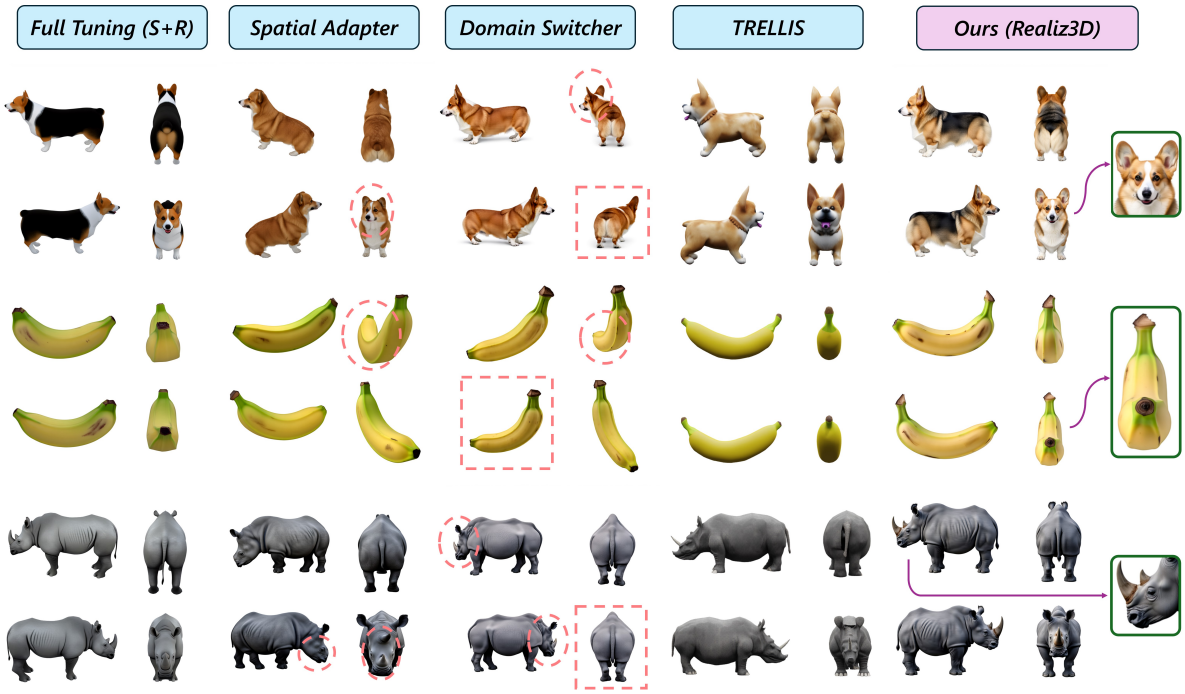

Figure 4. **Text-to-Multiview: Qualitative Results.** "A cute corgi dog", "A delicious ripe banana", "A rhino with thick gray skin", each prompt was appended with "highly photorealistic and detailed". Red circles/squares highlight inconsistent regions/incorrect viewpoints, respectively. Realiz3D achieves notable improvements in photorealism while maintaining strong 3D consistency. Best viewed zoomed in.

dating our conclusions, and the effectiveness of Realiz3D.

Qualitative Evaluation. We present visual examples in Fig. 1 and Fig. 4. Only best-performing and external baselines are shown in the main paper. Full comparisons and additional results appear in the Appendix.

6. Conclusions, Limitations and Future Work

We introduced *Realiz3D*, a fine-tuning strategy for diffusion models that enables controllability from synthetic data while preserving photorealism of the base model. Realiz3D is built on three key innovations: *Domain Shifters*, designed to learn separable visual domains; *layer-aware training* process, which maintains realism without sacrificing controllability; and *domain reassignment*, which improves control transfer to real domain. A domain-aware sampling process boosts performance at test time. Our work marks a significant step toward photorealistic 3D generation. **Limitations.** While Realiz3D significantly improves realism, a small gap in control adherence remains (Sec. 5). Key factors are: (1) 3D consistency is sensitive to fine-grained details; (2) Domain gaps can occasionally manifest in geometry, not just appearance; (3) The base model's lighting bias can cause inconsistent appearance (failure cases are in the Appendix). Advances in relighting [3, 18, 20] open promising avenues to address this. In addition, Realiz3D is currently designed to support control signals that are largely domain-agnostic, as text or geometric signals. This may limit direct applicability to other types of conditions, as images.

Acknowledgements

We sincerely thank Timur Bagautdinov, Oran Gafni, Ita Lifshitz and Thu Nguyen-Phuoc for invaluable discussions and feedback. Or Litany acknowledges support from the Israel Science Foundation (grant 624/25) and the Azrieli Foundation Early Career Faculty Fellowship. This research was also supported in part by an academic gift from Meta. The authors gratefully acknowledge this support.

References

[1] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2(3):4, 2021. 3

[2] Raphael Bensadoun, Yanir Kleiman, Idan Azuri, Omri Harosh, Andrea Vedaldi, Natalia Neverova, and Oran Gafni. Meta 3d texturegen: Fast and consistent texture generation for 3d objects. *arXiv preprint arXiv:2407.02430*, 2024. 2, 3, 5

[3] Sumit Chaturvedi, Mengwei Ren, Yannick Hold-Geoffroy, Jingyuan Liu, Julie Dorsey, and Zhixin Shu. Synthlight: Portrait relighting with diffusion model by learning to re-render synthetic faces. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 369–379, 2025. 8, 15

[4] Hila Chefer, Shiran Zada, Roni Paiss, Ariel Ephrat, Omer Tov, Michael Rubinstein, Lior Wolf, Tali Dekel, Tomer Michaeli, and Inbar Mosseri. Still-moving: Customized video generation without customized video data. *ACM Transactions on Graphics (TOG)*, 43(6):1–11, 2024. 3, 6

[5] Giannis Daras, Kulin Shah, Yuval Dagan, Aravind Gollakota, Alex Dimakis, and Adam Klivans. Ambient diffusion: Learning clean distributions from corrupted data. In *NeurIPS*, 2023. 3

[6] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 2

[7] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009. 6

[9] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024. 6

[10] Daniel Gatis. Rembg: A tool to remove image backgrounds. https://github.com/danielgatis/rembg, 2025. Accessed: 2025-10-15. 14

[11] Amin Ghiasi, Hamid Kazemi, Eitan Borgnia, Steven Reich, Manli Shu, Micah Goldblum, Andrew Gordon Wilson, and Tom Goldstein. What do vision transformers learn? a visual exploration. *arXiv preprint arXiv:2212.06727*, 2022. 3

[12] Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023. 3, 6, 13

[13] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 14

[14] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 2, 6

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 2

[16] Dengyang Jiang, Mengmeng Wang, Liuzhuozheng Li, Lei Zhang, Haoyu Wang, Wei Wei, Guang Dai, Yanning Zhang, and Jingdong Wang. No other representation component is needed: Diffusion transformers can provide representation guidance by themselves. *arXiv preprint arXiv:2505.02831*, 2025. 2, 3, 11

[17] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22511–22521, 2023. 2

[18] Ruofan Liang, Zan Gojcic, Huan Ling, Jacob Munkberg, Jon Hasselgren, Chih-Hao Lin, Jun Gao, Alexander Keller, Nandita Vijaykumar, Sanja Fidler, et al. Diffusion renderer: Neural inverse and forward rendering with video diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26069–26080, 2025. 8, 15

[19] Jiantao Lin, Xin Yang, Meixi Chen, Yingjie Xu, Dongyu Yan, Leyi Wu, Xinli Xu, Lie Xu, Shunsi Zhang, and Ying-Cong Chen. Kiss3dgen: Repurposing image diffusion models for 3d asset generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5870–5880, 2025. 2

[20] Yehonathan Litman, Fernando De la Torre, and Shubham Tulsiani. Lightswitch: Multi-view relighting with material-guided diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27750–27759, 2025. 8, 15

[21] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9298–9309, 2023. 2

[22] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang,

Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9970–9980, 2024. 2, 3, 6, 13

[23] Grace Luo, Lisa Dunlap, Dong Huk Park, Aleksander Holynski, and Trevor Darrell. Diffusion hyperfeatures: Searching through time and space for semantic correspondence. *Advances in Neural Information Processing Systems*, 36: 47500–47510, 2023. 2, 3

[24] 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*, 2022. 3, 6, 14

[25] 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 *Proceedings of the AAAI conference on artificial intelligence*, pages 4296–4304, 2024. 2, 6

[26] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 3, 6

[27] Chensheng Peng, Ido Sobol, Masayoshi Tomizuka, Kurt Keutzer, Chenfeng Xu, and Or Litany. A lesson in splats: Teacher-guided diffusion for 3d gaussian splats generation with 2d supervision. *arXiv preprint arXiv:2412.00623*, 2024. 3

[28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 14

[29] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *NeurIPS*, 2017. 4, 13

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 3

[31] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023. 2, 3

[32] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation, 2024. 2, 3, 5, 6

[33] Ido Sobol, Chenfeng Xu, and Or Litany. Zero-to-hero: Enhancing zero-shot novel view synthesis via attention map filtering. *Advances in Neural Information Processing Systems*, 37:30522–30553, 2024. 3

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

[35] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 14

[36] Saar Stern, Ido Sobol, and Or Litany. Appreciate the view: A task-aware evaluation framework for novel view synthesis. *arXiv preprint arXiv:2511.12675*, 2025. 3

[37] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024. 2, 6

[38] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1921–1930, 2023. 2, 3, 11

[39] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 6

[40] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21469–21480, 2024. 6, 15

[41] Xinli Xu, Wenhang Ge, Jiantao Lin, Jiawei Feng, Lie Xu, HanFeng Zhao, Shunsi Zhang, and Ying-Cong Chen. Flexgen: Flexible multi-view generation from text and image inputs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18714–18724, 2025. 2

[42] Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made slim. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 22552–22562, 2023. 3

[43] Mingyang Yi, Aoxue Li, Yi Xin, and Zhenguo Li. Towards understanding the working mechanism of text-to-image diffusion model. *arXiv preprint arXiv:2405.15330*, 2024. 3

[44] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. *European Conference on Computer Vision*, 2024. 2

[45] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 2

[46] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. 6

Appendix

Contents

A. Feature Maps in Diffusion Models

A.1. Motivation

As thoroughly discussed in the *Diffusion Models and Domain Gaps* section (Section 3) of the main paper, prior works has shown that visual details evolve throughout the denoising process in diffusion models, both across timesteps and across layers of the neural network. Building on studies that analyze diffusion features [16, 38], we examine and visualize the internal representations of our diffusion transformer backbone over the course of denoising.

To do so, we first generate diverse prompts describing random objects in random styles (e.g., photorealistic, cartoon, watercolor, anime, comic, etc.) using an LLM. We then synthesize images from these prompts using our base DiT text-to-image model, running generation with 20 DDIM steps.

During generation, we extract intermediate feature maps at multiple timesteps and layers of the network. We apply PCA to these features and visualize the resulting components as RGB images, presented in Fig. 5. Importantly, PCA is computed independently for each timestep–layer pair, so the colors are not consistent across timesteps or layers. Concretely, we extract features at four timesteps (800, 700, 500, and 200, with $T = 1000$) and at three layers corresponding to the beginning, middle, and end of the model. Let N_B denote the total number of DiT blocks.

We present the PCA-reduced features in Fig. 5. The figure comprises four vertically stacked subfigures, where each subfigure shows features from different layers at a **fixed** timestep. Within each subfigure, rows follow the order of the network depth. The top row shows features from an early layer in the architecture, and the bottom row shows features from a later layer. Similarly, the ordering of the subfigures follows the denoising schedule: the top subfigure corresponds to an early timestep at sampling (where the noise level is highest), and the bottom subfigure corresponds to a late timestep (where the noise level is lowest).

As shown in the figure, when the noise level is high, the features primarily capture coarse and structural information, which is largely domain-agnostic. As the noise level decreases, the features increasingly reflect high-frequency patterns and fine-grained details.

The denoising layers exhibit a similar trend: early layers encode coarse structural patterns, shared between synthetic and real data, whereas later layers capture fine details.

A.2. Case Study: Layer-Selective Training in 2D Image Generation

As discussed above and on the main paper, early layers in the diffusion transformer network usually correspond to structural and coarse patterns, while later laters capture fine-details and high-frequency patterns. We leverage this observation in our proposed Layer-Aware Training, where real data affect later layer more strongly and synthetic data has a stronger influence of the early layers.

Before developing our Layer-Aware Training strategy, we conducted a proof-of-concept experiment for image generation, to test whether this insight could be used during training to gain controllability from synthetic data while preserving realism from real data. The results are shown in Fig. 6.

We conducted the proof-of-concept experiment using two equal-sized datasets: one synthetic and one realistic. The synthetic images were rendered from 3D assets, while the realistic images were generated by a T2I model using prompts describing similar objects, augmented with the phrases “highly realistic” and “white background” appended to each prompt.

We fine-tuned the base DiT T2I model in two ways. First, we fine-tuned it solely on synthetic data. Second, we applied a layer-selective strategy: the first 50% of DiT blocks were fine-tuned on synthetic data (while the later blocks were frozen), and the remaining 50% were fine-tuned on realistic data (while the early blocks were frozen). Although both approaches caused the model to shift toward object-centric generation with white backgrounds, the layer-Selective strategy consistently preserves the base model’s realistic prior more effectively than the full-synthetic baseline, as shown in Fig. 6. While a sim-

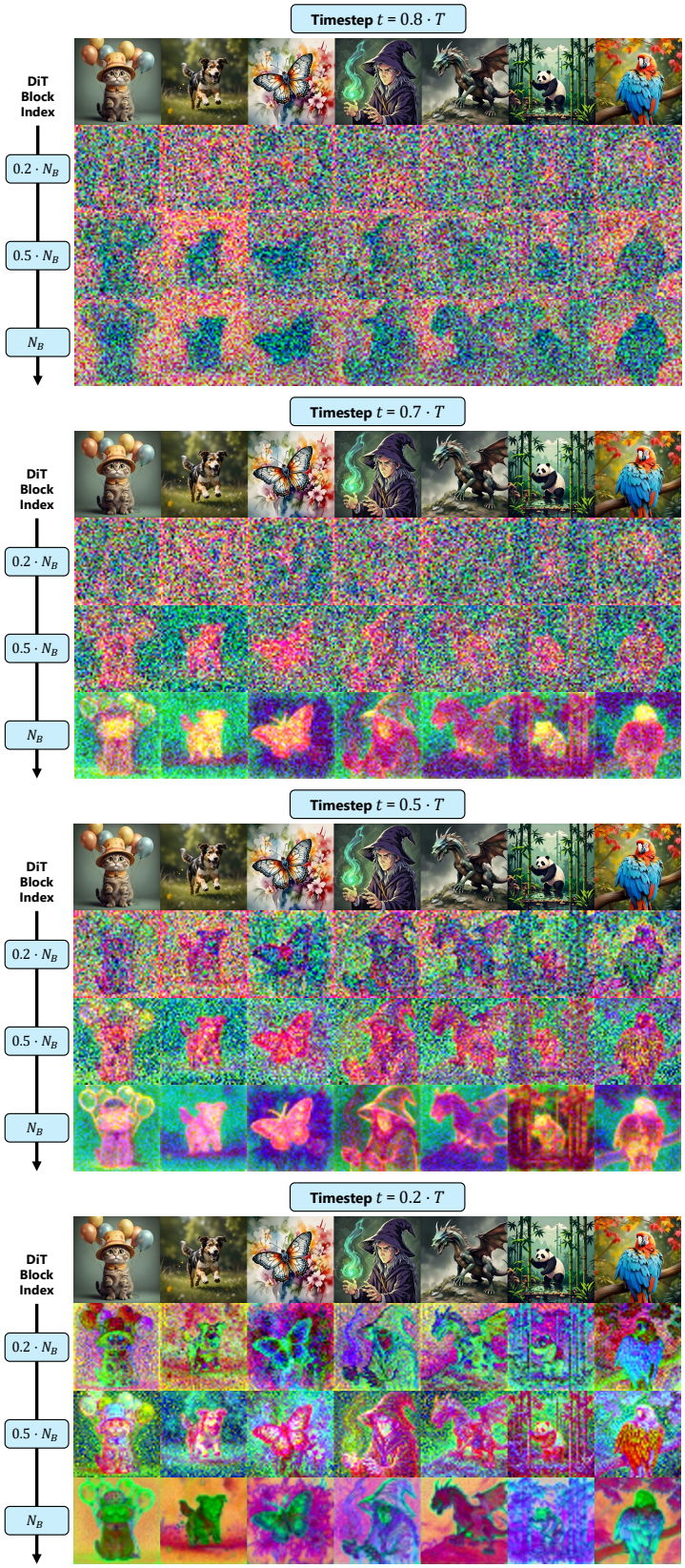

Figure 5. Diffusion features, extracted from our base T2I model at different timesteps and layers during the generation process. We visualize the first three PCA components as RGB images. See Sec. A.1 for additional details and analysis.

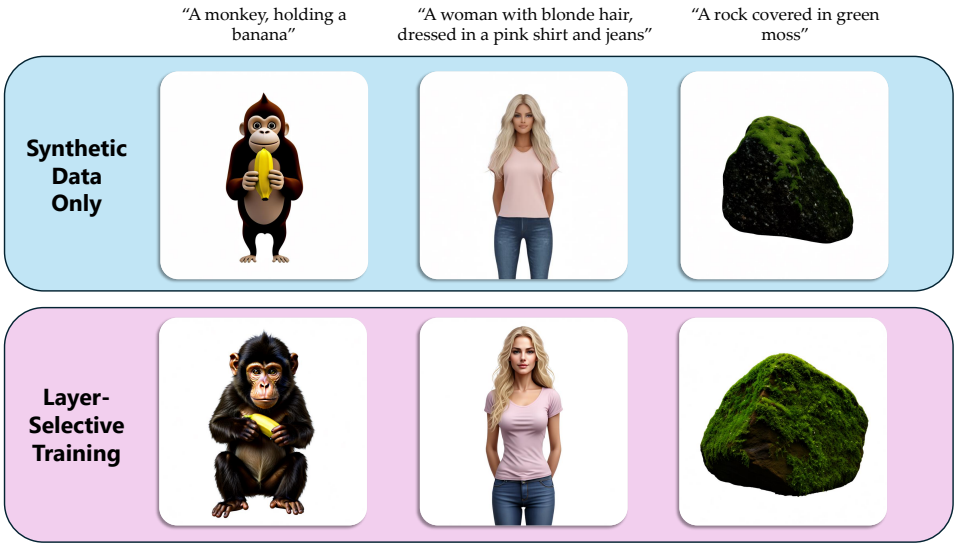

Figure 6. **Layer-Selective Training.** Training early blocks on synthetic data and later blocks on real data enables the model to learn controllable properties from synthetic data while maintaining photorealism. See further details in Sec. A.2.

ilar experiment was done with timestep-selective training, we found the layer-selective approach to be more stable and robust.

This proof-of-concept experiment provides the core motivation and inspiration for the layer-aware training approach described in the main paper.

B. Method

In this section, we provide additional information and implementation details about the different components of Realiz3D.

B.1. Domain Shifters (Stage 1)

Discussion: Domain Shifter Design Our design builds on the intuition that our goal is to rebalance existing visual modes within the pretrained model rather than introduce new modalities. Prior adapter-based methods [12, 22] handle much larger modality shifts: AnimateDiff [12] suppresses the low-resolution and noisy video latent pathway to inject temporal conditioning, while Wonder3D adds conditioning streams for normal maps that differ substantially from natural images. In our case, both real and synthetic imagery already occupy the model's learned feature space, making a lightweight low-rank residual sufficient to adjust their balance without disrupting pretrained representations [29].

B.2. Fine-tuning with Representation Binding (Stage 2)

As described in the main paper, we incorporate real samples during training to prevent forgetting realism. Since real images lack explicit control supervision, naïvely training on them could disrupt control learned from synthetic data. To avoid this, we use real data to mostly update the later diffusion blocks, responsible for appearance refinement, while freezing the early blocks.

Concretely, During each real-data training iteration, we freeze DiT blocks $B \in [0, B_i]$, where i is an integer block index randomly drawn from $[0, \tau_B]$. This stochastic layer-freezing regularizes more strongly early representations, without requiring a fixed cutoff. We find the stochastic approach to be robust to small variations and selection of configuration. Empirically, setting $\tau_B \in [0.4, 0.5]$ of the total number of blocks provides stable and robust performance.

Throughout the entire training process, we rely solely on the traditional diffusion loss, which is well established and empirically stable. At stage 1, the diffusion objective is used to train our Domain Shifters, and at stage 2 the same objective is used to train the DiT backbone. When training on synthetic samples, the training objective is $\mathcal{L} = \mathbb{E}_{z,\epsilon\sim\mathcal{N}(0,1),\,t}\left[\|\epsilon - \epsilon_\theta(z_t, t, c, e_{syn})\|_2^2\right]$ (at stage 1, $c = \varnothing$).

When training on realistic samples, the training objective is $\mathcal{L} = \mathbb{E}_{z,\epsilon\sim\mathcal{N}(0,1),\,t}\left[\|\epsilon - \epsilon_\theta(z_t, t, c = \varnothing, e_{real})\|_2^2\right]$.

B.3. Inference-time Domain Shifting

As discussed in the main paper, to further promote control transfer, we introduce *Domain Reassignment*. With probability p_B, we reassign early DiT blocks ($B \in [0, B_j]$, j is an integer, sampled from $[0, \tau_B]$) to operate in synthetic mode even when processing real samples; that is, we substitute $e_{\text{domain}} \leftarrow e_{\text{syn}}$ in the corresponding Domain Shifters.

In practice, we set $e_{\text{domain}} = e_{\text{syn}}$ when either of the following conditions hold:

- *Layer-based shifting:* For all timesteps t and DiT blocks $B \leq B_{\text{max}}$, corresponding to coarse sample structure. We typically set B_{max} to 20%-30% of the total number of blocks N_B.
- *Time-based shifting:* For all DiT blocks and timesteps $t \geq t_{\text{max}}$, corresponding to high noise levels. We find $t_{\text{max}} \in [800, 1000]$ to be most effective.

By tuning B_{max} and t_{max} on a small validation set, users can trade off between stronger control adherence and higher photorealism without any retraining. In our experiments, we prioritize realism, and perform inference-time domain shifting to improve controllability while keeping the realism loss minimal.

We observe that time-based shifting tends to have a stronger effect on realism while layer-based shifting tend to have a mild and stable effect. Therefore, to select the mentioned hyperparameters, we first fix B_{max} by evaluating a small validation set across a few candidate values, ranging from 0% to 40% of the total blocks in 10% increments. Then, we select t_{max} to achieve the desired balance, testing values between 800 and 1000 in steps of 50. For the selection process, we evaluate realism-based metrics and define a threshold that represents the maximum reduction in realism we are willing to tolerate. Since we do not perform an exhaustive search, the entire tuning process takes less than an hour and is performed only once.

C. Implementation Details

C.1. Data

Realistic Training Data. As mentioned in the main paper, to create the realistic dataset, we use the same prompts from the synthetic dataset and generate $V = 4$ photorealistic images using the base T2I model. This ensure fair evaluation, as both datasets contain the same diversity of objects.

As the base model is biased towards the frontal view, we generate each prompt using four viewpoint descriptions ("front", "back", "right side", "left side"), thus collecting $V = 4$ photorealistic images per prompt. The model often fails to follow the desired viewpoint, yet this strategy helps to enrich the data and increase pose variation.

These view descriptions are **not** used during training. During training, the prompts associated to the images do not contain this information, and the random placement of realistic images in the 2×2 grid is not affected by this information.

ImageNet Curation Process for Real-World Realism Metrics While the prior preservation metrics indicate whether the generated images seem realistic, these metrics are based on synthesized images. We aim to ground our evaluation in real-world image data which, to the best of our knowledge, was not used for training our base model nor the pretrained models.

To that end, we processed the prompts describing the evaluation objects through LLM, to find the most fitting categories in the ImageNet dataset, resulting in 42 classes. Then, we use CLIP [28] score to select 4 images per class from the validation set, and segment them using an internal tool similar to Rembg [10].

C.2. Training and Sampling

Training. All evaluated models are trained for 10 epochs on 64 NVIDIA H100 GPUs, using batch size of 8 and learning rate of $5e^{-}5$. Adapters that are trained separately are trained for 3 epochs. We verified that all models and adapters converged and did not reach overfitting. Importantly, throughout the training process we rely only on the standard reconstruction diffusion loss, which is well-established and empirically stable. To prevent adapter-based methods from overfitting to white-background, object-centric layouts, we perform a very short warmup on realistic samples in all experiments that incorporate real data during training.

Sampling. For sampling we use DDIM [35] with 50 steps. Following the base model, we use CFG [13] only for the text condition.

Hyperparameters. For all experiments we set $r = 8$ for Domain Shifter modules.

For Representation Binding, we set τ_B to be 40% of the total number of blocks and $p_b = 0.1$.

For Inference-time Domain Shifting we set B_{max} to be 30% of the total number of blocks and $t_{\text{max}} = 950$. The selection process of B_{max} and t_{max} happens at test-time using a small validation set of 20 held-out objects.

D. Evaluation

D.1. Implementation Details

SDEdit Baseline. We evaluate the effectiveness of the SDEdit [24] mechanism for multiview applications. Specifically, we take the outputs of the synthetic full fine-tuning baseline, add noise to the images, and then denoise them independently using the base pretrained T2I model. The same noise realization is applied to all images of the same grid, while each image is denoised independently. Following the original paper, we set the noise injection timestep to 500. At higher timesteps, images are expected to exhibit reduced 3D consistency while potentially improving realism. How-

ever, even at $t = 500$, SDEdit yields inferior performance to Realiz3D in both 3D consistency and realism.

Pretrained Models. For Text-to-Multiview Generation, we also evaluate TRELLIS [40], a 3D-native model. We use the official text-to-3D "TRELLIS-text-large" model with the original hyperparameters. TRELLIS directly generates a 3D asset and not multiview images, therefore we generate a 3D asset from text and render the 4 orthogonal views (front, back and sides views) that we show across the paper.

Qualitative Results. For all baselines reported with two rank settings (32 and 128), all results shown in the main paper and the supplementary materials were obtained using rank 32.

D.2. Ablation Study: Qualitative Results

To further support the ablation study from the main paper, we present visual examples in Fig. 7 that illustrate the effects of our Representation Binding and Inference-Time Domain Shifting on the model's final outputs. Both techniques improve control adherence while preserving strong realism.

D.3. Additional Qualitative Results

We present qualitative results using the evaluation data described in the main paper, as well as additional test objects from our internal dataset that were held out for evaluation and not used during training.

Multiview Texturing. We present additional qualitative results in Fig. 8, Fig. 9 and Fig. 10, showing all baselines described in the main paper. The corresponding prompt appears in the caption. Although our method uses both normal and position maps as geometric conditions, only the normal maps are shown for readability. Realiz3D achieves significant improvements in photorealism while remaining 3D-consistent and faithful to the geometric conditions.

Text-to-Multiview Generation. We present additional qualitative results in Fig. 11, Fig. 12 and Fig. 13, showing all baselines described in the main paper. The corresponding prompt appears in the caption. Realiz3D achieves notable improvements in photorealism while maintaining strong 3D consistency.

D.4. Text-to-3D Results

We perform text-to-3D generation by backprojecting generated textures onto their corresponding original meshes. Note that we generate only four orthogonal views, which may not fully cover the entire surface.

The results are provided on our project page, where we present side-by-side 3D assets produced by the full-tuning baseline (trained on both real and synthetic data for fairness) and by Realiz3D.

The results highlight that Realiz3D achieves comparable 3D-consistency to the fully synthetic and full fine-tuning baselines, producing coherent and realistic 3D assets.

E. Limitations and Future Work

As mentioned in the main paper, although Realiz3D significantly improves realism, a small gap in control adherence remains. We attribute this to several key factors: (1) 3D consistency is sensitive to fine-grained details. Because the synthetic data primarily contains smooth textures, the task becomes easier for the model. As a result, synthetic baselines tends to produce relatively smooth outputs, reducing the need to learn perfect pixel-level 3D consistency. In other words, these baselines often appear consistent even without having learned accurate pixel-level 3D-consistency. In contrast, Realiz3D produces fine-grained details (e.g., complex textures and materials, hair and fur) that are much more sensitive to even slight misalignment. (2) Domain gaps can occasionally manifest in unrealistic geometry, not just appearance, causing Realiz3D to slightly deviate from the original geometry. (3) The base model's lighting bias can lead to inconsistent appearance. Our synthetic data is uniformly lit, and the generated views largely preserve this property, appearing evenly illuminated. However, the base T2I model shows a bias in lighting for certain objects and materials. An example failure case appears in Fig. 14, where we generate the texture of a hamburger. The hamburger is consistently lit more strongly from the front right, while the back left remains noticeably darker. The same trend is consistent across different seeds. After examining hamburger images online and inspecting those generated by our base T2I model, we find that this lighting condition is extremely common in real-world images, creating a strong lighting bias in the base model. Recent advances in relighting [3, 18, 20] and specifically the use of two synthetic domains, one uniformly lit and one randomly lit, offer promising avenues for addressing this gap.

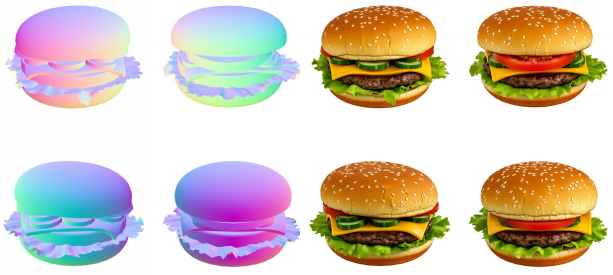

Figure 14. **Multiview Texturing.** Inconsistent lighting caused by the base T2I model's lighting bias.

In addition, a natural extension of Realiz3D is to apply our techniques to video diffusion models, which have recently demonstrated remarkable capabilities. When these

Figure 7. **Ablation Study.** We demonstrate the importance of our Representation Binding and Inference-time Domain Shifting on Multi-view Texturing. Red circles highlight inconsistent regions with the geometry. Both techniques enhance control adherence, while maintaining realism. The presented prompts: ”A baby stroller with a leather seat and a black plastic container on the bottom”, ”A structure with a flat top, made of natural stone”.

models are trained to incorporate a 3D condition, they are often fine-tuned on synthetic data, introducing a similar domain gap.

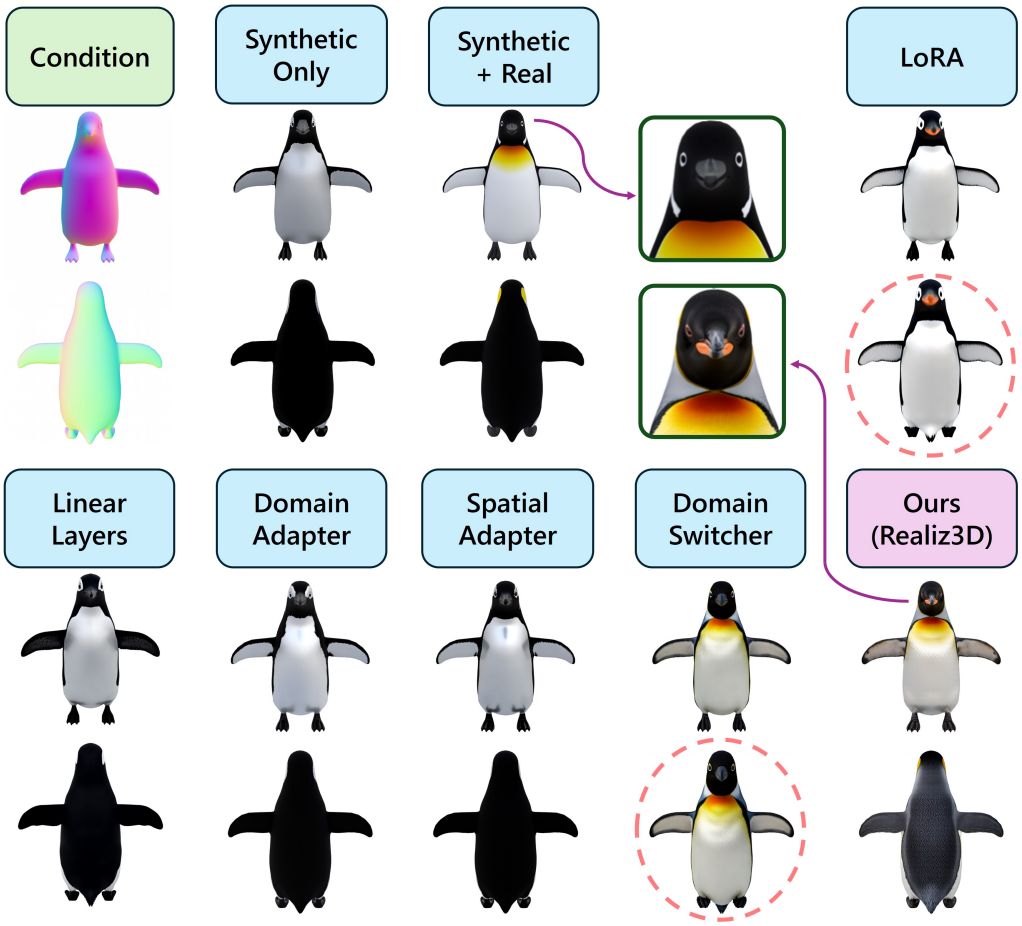

Figure 8. **Multiview Texturing.** "A king penguin, highly realistic and detailed". Red circles highlight inconsistent regions (either with the geometry or with other views). Best viewed zoomed in.

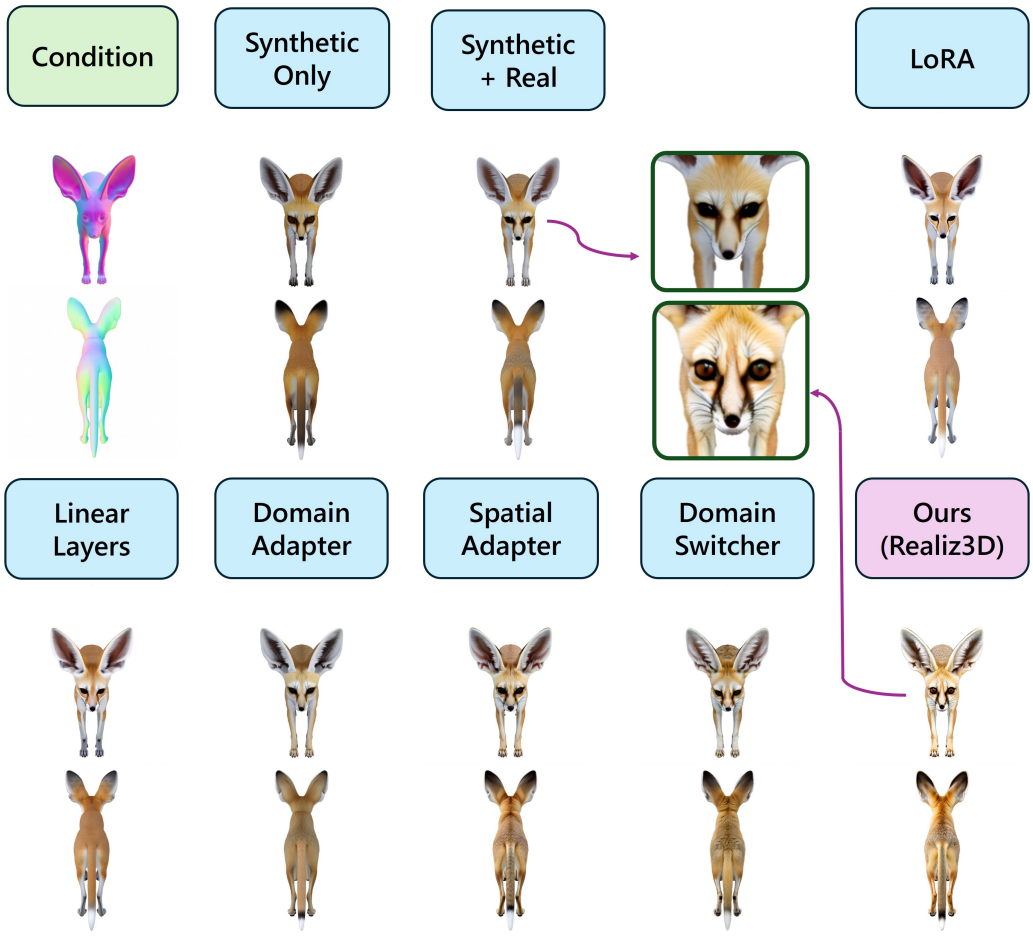

Figure 9. **Multiview Texturing.** ”A fennec fox, highly realistic and detailed”. Best viewed zoomed in.

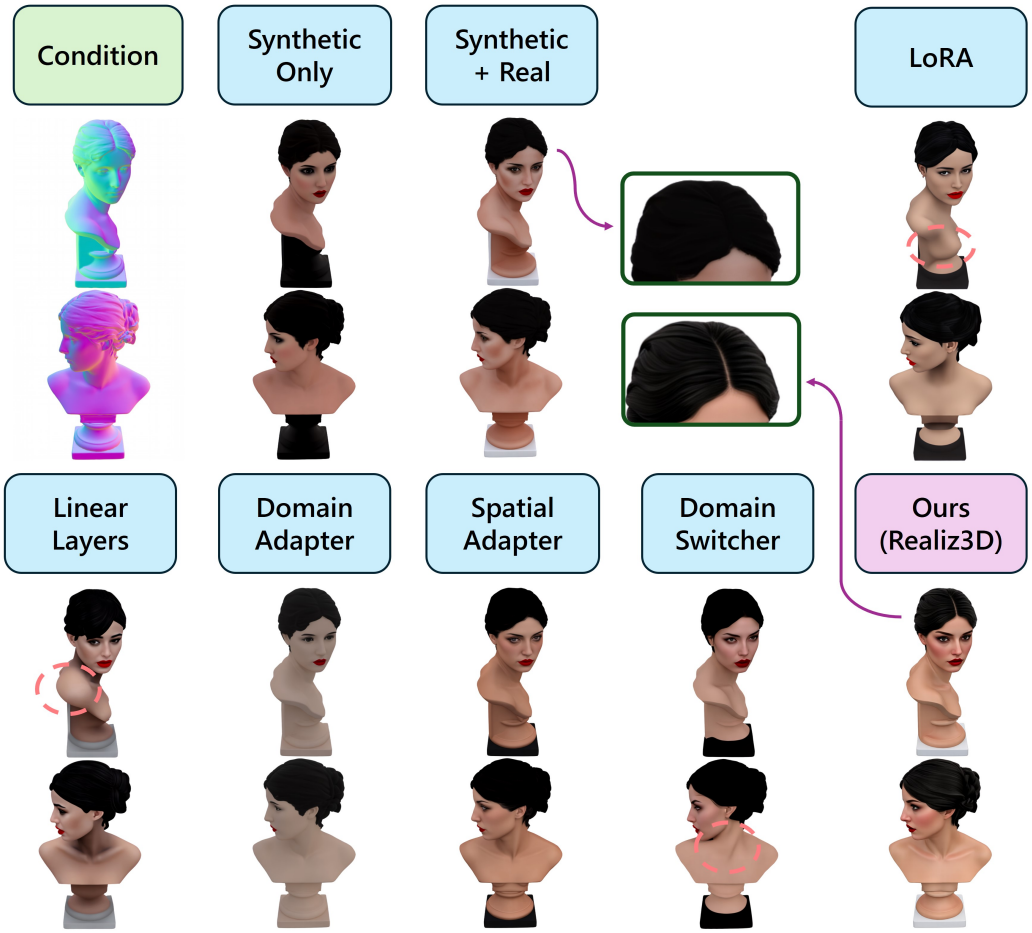

Figure 10. **Multiview Texturing.** ”A woman with a dark hair and red lips, highly realistic and detailed”. Red circles highlight inconsistent regions (either with the geometry or with other views). Best viewed zoomed in.

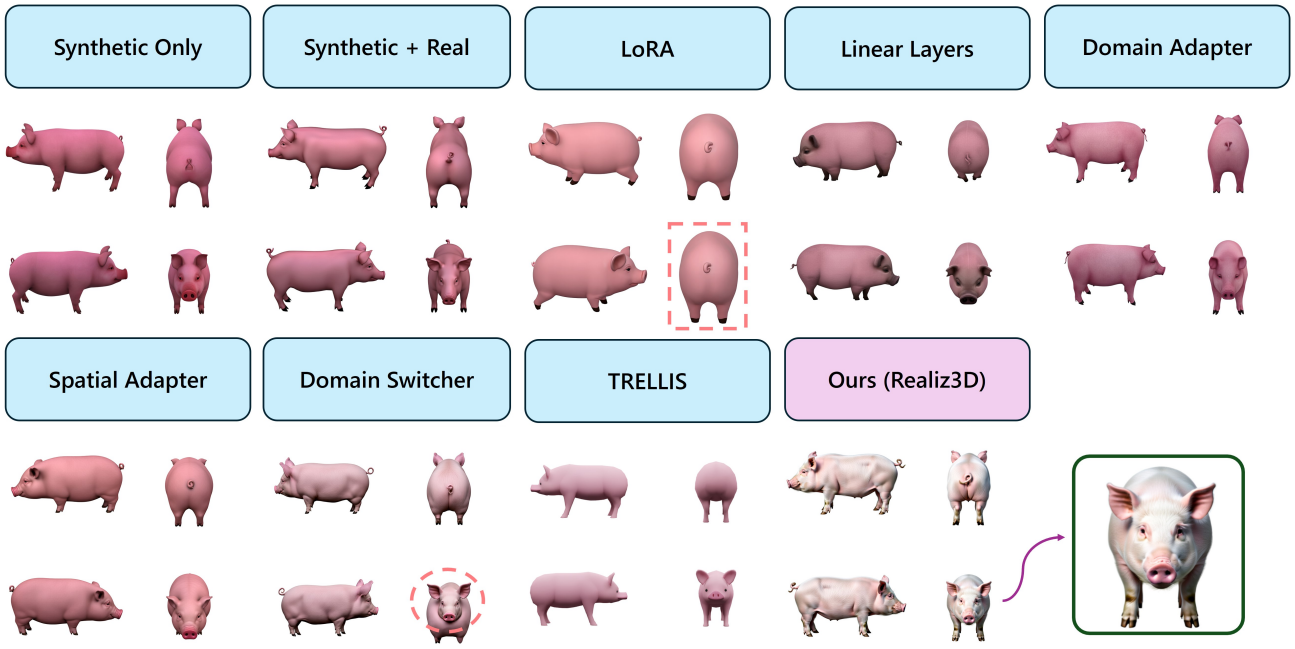

Figure 11. **Text-to-Multiview Generation.** "A pink farm pig, highly realistic and detailed". Red circles/squares highlight inconsistent regions/incorrect viewpoints, respectively. Best viewed zoomed in.

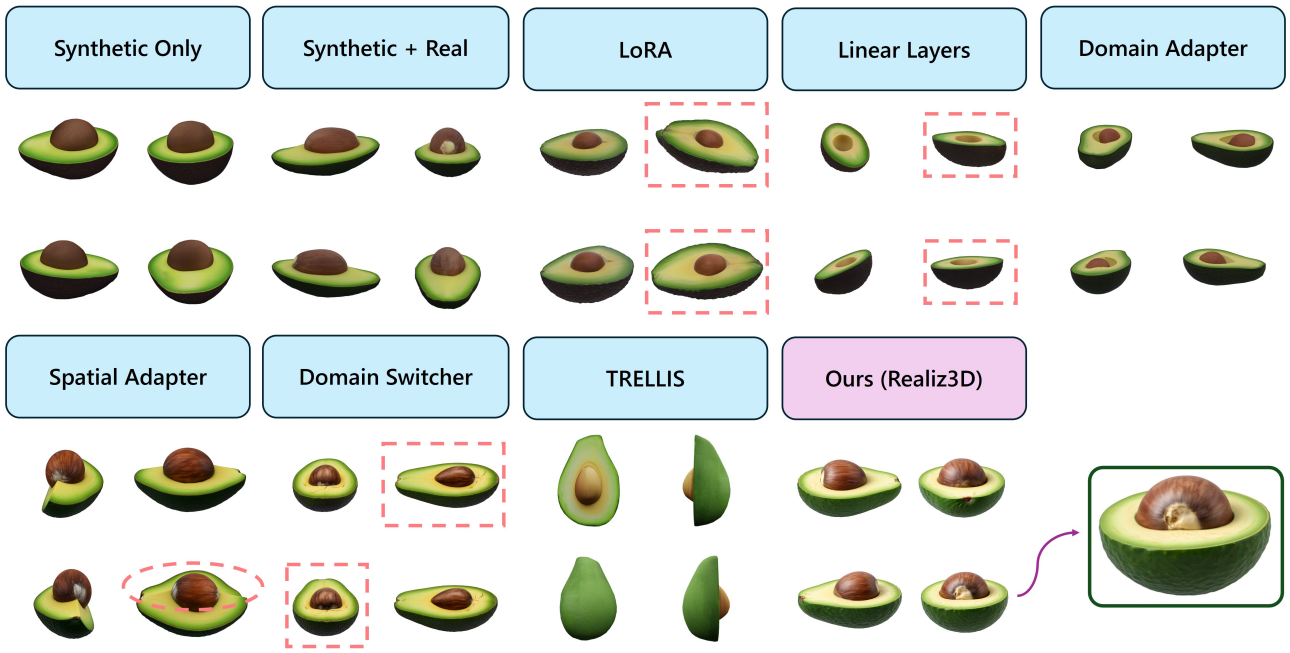

Figure 12. **Text-to-Multiview Generation.** "A half slice of avocado, highly realistic and detailed". Red circles/squares highlight inconsistent regions/incorrect viewpoints, respectively. Best viewed zoomed in.

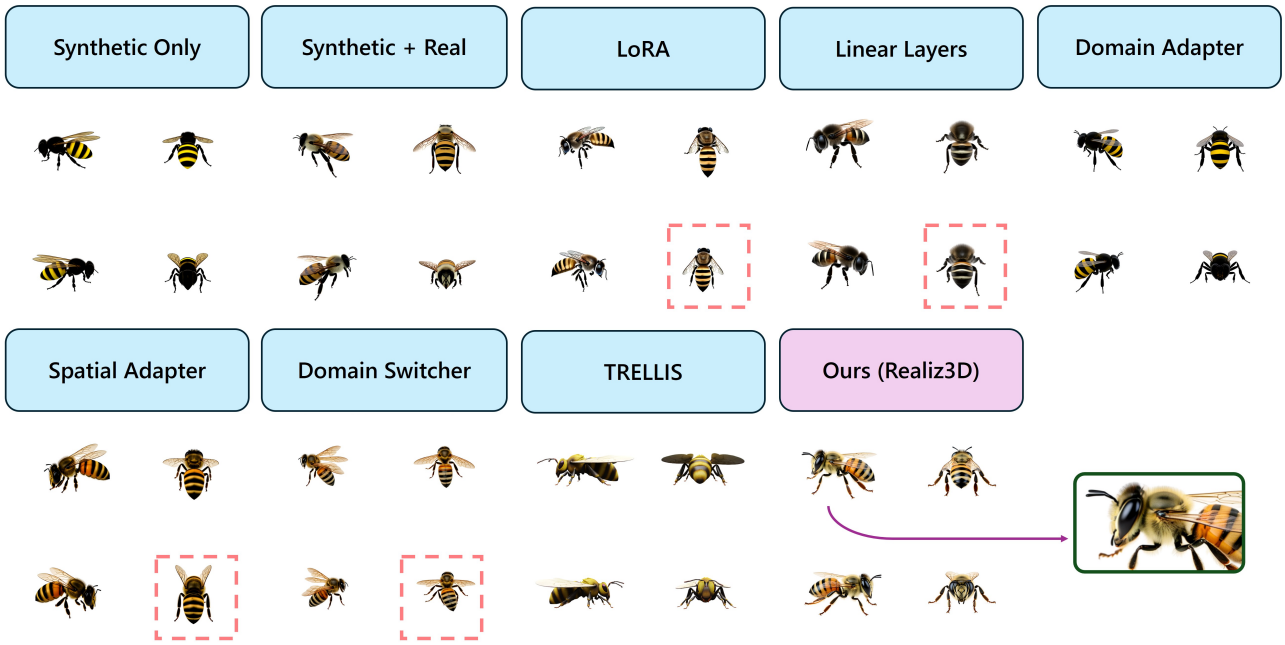

Figure 13. **Text-to-Multiview Generation.** "A bee, highly realistic and detailed". Red circles/squares highlight inconsistent regions/incorrect viewpoints, respectively. Best viewed zoomed in.