FLiTo: Fast High-Fidelity 3D Generation via Image Supervision

Anonymous authors
Paper under double-blind review

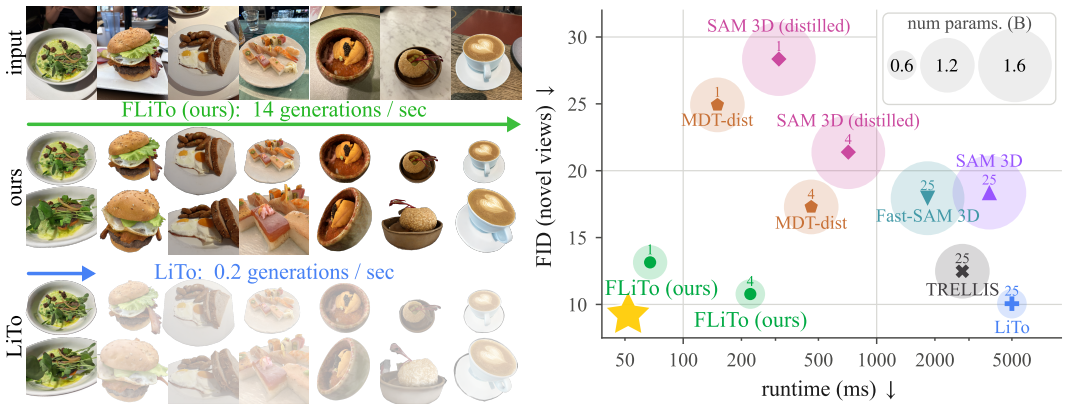

Figure 1: **FLiTo generates 70 3D latents in the time LiTo generates one.** Left: Our single-step model achieves 70× speedup while closely matching the quality of the original multi-step LiTo model (Chang et al., 2026). Right: Across recent image-to-3D methods, FLiTo achieves a substantially better speed–quality trade-off. Marker size denotes model parameter count, and numbers indicate generation steps. Input photos are from authors' photo library. Please see the supplemental website for videos of generated objects.

Abstract

Recent advances in latent 3D generative modeling have enabled high-quality 3D content generation from images. However, state-of-the-art models typically require many generation steps, resulting in substantial inference cost and limiting their use in interactive applications. While distillation can reduce the number of sampling steps, generation quality degrades significantly in the few-step regime, particularly for single-step generation. In this work, we study how 2D supervision can be effectively incorporated to improve single- and few-step latent 3D generation. We derive a variational objective that naturally unifies conditioning-image reconstruction with distribution matching against a teacher 3D generator. Building on this formulation, we incorporate complementary forms of 2D supervision, including feature-distribution matching, semantic consistency, and human preference. Through extensive ablations, we analyze the contribution of each supervision signal and show that 2D supervision is particularly effective in the challenging single-step regime. Our final single-step generator achieves state-of-the-art speed and generation quality among single-step 3D generators, reducing FID by 47% over the strongest baseline while running in 67.5 ms on an NVIDIA B200 GPU and 2.4 seconds on an MacBook Pro with M5 Max.

1 Introduction

Recent advances in image-to-3D generation have led to remarkable improvements in the quality and generality of generated 3D assets, as demonstrated by methods such as TRELLIS (Xiang et al., 2025b), TRELLIS.2 (Xiang et al., 2026), SAM3D (Chen et al., 2026), and LiTo (Chang et al., 2026). However, high-quality generation typically relies on iterative sampling over many steps,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

resulting in substantial inference latency. Reducing this latency is important for bringing image-to-3D generation into interactive creative workflows, where users often need to quickly generate, inspect, and iterate on 3D assets. Although recent works have explored distilling these multi-step models to fewer sampling steps, aggressive acceleration often comes at the cost of generation quality, and producing high-quality 3D assets in just a single step remains particularly challenging.

This quality–efficiency trade-off becomes particularly pronounced as the number of sampling steps is reduced. Existing acceleration like Fast-SAM3D (Feng et al., 2026) and distillation methods like SAM3D-distilled (Chen et al., 2026) and MDT-dist (Zhou et al., 2026) can retain much of the quality of the original models (SAM3D and TRELLIS (Xiang et al., 2025b)) with a moderate number of steps, but their performance degrades rapidly under more aggressive reduction. As shown in Fig. 1, the gap is especially large in the single-step setting, where the model must map an input image to a high-quality 3D representation in a single forward pass. This raises a natural question: how can we provide sufficiently strong supervision to a single-step model to recover the quality of iterative generation?

In this work, we show that meaningful 2D supervision can substantially narrow the quality gap between single-step and iterative image-to-3D generation. We begin by deriving a variational objective for maximizing the likelihood of 2D renderings of generated objects. This derivation naturally yields two complementary terms: a rendering objective that encourages consistency with the input image, and a Kullback–Leibler divergence (KL) regularizer that keeps the student distribution close to the teacher, which we instantiate using DMD (Yin et al., 2024b). Our ablation shows that both terms are essential in practice. Rendering supervision alone substantially improves the generation quality, but it leads to degenerate solutions that prioritize reconstructing the visible surface of the input view. The KL regularization prevents this collapse and restores complete, plausible 3D geometry and view-dependent appearance, validating the behavior predicted by our derivation. Building on this perspective, we further interpret our framework as reinforcement learning with image-space rewards and a KL prior, allowing us to incorporate richer forms of 2D supervision, including semantic similarity (Radford et al., 2021; Tschannen et al., 2025), feature distribution matching (Yang et al., 2026), and human preference (Ma et al., 2025). Together, these ingredients produce our highest-quality single-step model, achieving the best FID among the compared single- and few-step methods while requiring only 67.5 ms on a B200 GPU, 92.2 ms on an H100 GPU, and 2.4 seconds on an MacBook Pro with M5 Max, from image to 3D latents. The same framework also generalizes to few-step generation, where it further improves the quality of a four-step model.

In summary, we make the following contributions:

- **Principled objective.** We derive a variational objective for maximizing the likelihood of 2D renderings and show that it naturally decomposes into a rendering objective and a KL regularizer.
- **General 2D-supervision framework and detailed ablations.** We formulate our approach as reward optimization with a KL prior, enabling the incorporation of additional 2D supervision, including CLIP, SigLIP, feature-distribution matching, and human-preference objectives. Through detailed ablations, we demonstrate the complementary roles of the rendering and KL terms and quantify the contribution of each form of supervision.
- **Strong empirical results.** Our method reduces FID compared with previous state-of-the-art single-step image-to-3D models, MDT-dist by 47% and SAM3D-distilled by 53%, while enabling highly efficient generation in only 67.5 ms on a B200 GPU and 2.4 seconds on an MacBook Pro.

2 Related Work

3D generative modeling. Generative modeling of 3D objects has been explored across a wide range of representations, including implicit fields, point-based representations, and learned latent spaces. Today, state-of-the-art approaches rely on latent representations that can be broadly classified into two categories. First, latent set approaches (Zhang et al., 2023; Chang et al., 2026; Zhang et al., 2024; Zhao et al., 2025) effectively compress 3D objects into a latent point cloud, enabling a highly detailed latent representation that can pose difficulty for learning (Lai et al., 2025a). Second, sparse voxel representations (Xiang et al., 2025b;a; Li et al., 2025; Chen et al., 2026; Wu et al., 2024; 2025) express objects as a combination of coarse voxel geometry and sparse latent appearance; however, such a decomposition necessitates two-stage generation. Regardless the underlying

representation, the vast majority of such approaches currently rely on flow-based multi-step models for generation, which are currently too costly for on-device usage. Our work explores how a LiTo generative model (Chang et al., 2026) can be accelerated to a few or even just a single step, while preserving much of the original performance via 2D image supervision.

Few-step generative modeling. Reducing the sampling cost of flow-based models has received considerable attention in image generation. Consistency Models (Song et al., 2023) learn mappings from noisy diffusion trajectory points to their clean trajectory endpoints, enabling few-step sampling. MeanFlow (Geng et al., 2026a) directly learns an average velocity field for few-step generation. Distribution Matching Distillation (DMD) (Yin et al., 2024b;a) distills a diffusion model into a few-step generator by matching the generator and teacher distributions via an auxiliary flow model.

Few-step generation has only recently been explored for 3D. SAM3D (Chen et al., 2026) trained a shortcut model (Frans et al., 2025) for their sparse-structure generator, that enables higher quality with fewer sampling steps. FlashVDM (Lai et al., 2025b) introduced progressive flow distillation to reduce VecSet diffusion models to a few sampling steps, while MDT-dist (Zhou et al., 2026) distilled TRELLIS (Xiang et al., 2025b) to one or two steps via a combined MeanFlow and DMD-type formulation. Rather than distilling the model, Fast-SAM3D (Feng et al., 2026) accelerated the pretrained SAM3D model (Chen et al., 2026) without retraining by reusing computation across sampling steps and tokens. Our work studies how supervision on the rendered 2D images can provide complementary training signals for few-step 3D generation.

2D supervision for 3D generation. We are not the first to explore how to train 3D generative models with 2D supervision. A significant body of work has explored how GAN-type architectures can be applied to this context, where a generator outputs a 3D representation while a discriminator provides feedback by comparing 2D renderings with ground truth images (Chan et al., 2022; Skorokhodov et al., 2023; Chen et al., 2019; Gao et al., 2022). However, such architectures are historically difficult to train and prone to mode collapse. Beyond adversarial supervision, pretrained 2D diffusion models have also been used to guide 3D generation through score-distillation optimization (Poole et al., 2022; Wang et al., 2023), but these methods usually require per-sample optimization during inference. In the context of 3D flow matching models, existing methods have proposed additionally integrating rendering losses on the denoised clean predictions during standard flow matching training (Shabanov et al., 2026; Peng et al., 2025); however, such techniques are not particularly theoretically grounded, and may break the diffusion trajectory. Instead of flow matching or diffusion directly in 3D, a number of approaches have explored whether a 3D representation can instead be used as a bottleneck for 2D flow-based models (Anciukevičius et al., 2023; Cai et al., 2025), but such techniques can be expensive as object coverage scales with the number of generated views. In the space of 3D reconstruction, Reddy et al. (2024) and Jiang et al. (2024) have also adapted semantic consistency losses that are similar to ours to supervise novel views, but these approaches usually produce blurry novel views, as they perform reconstruction instead of generation.

3 METHOD

As discussed in Section 1, the performance of 3D generative models degrades as the number of sampling steps decreases, particularly for the single-step setting. We attribute this to the reduced capacity of the generative process — as step count decreases, the model has fewer opportunities to refine its own output, and must instead commit to a final 3D asset in one pass. To address this problem, we note that generated objects are ultimately perceived through their rendered 2D views. We therefore hypothesize that, when the capacity of the generative process is severely constrained, as in one-step generation, directly supervising these renderings provides a stronger and more effective learning signal than relying on 3D supervision alone. This motivates our use of 2D image supervision to improve the quality of aggressively distilled 3D generators. See Fig. 2 for illustration.

3.1 MAXIMIZING 2D IMAGE LIKELIHOOD VIA VARIATIONAL LOWER BOUND

Given a dataset $\mathcal{D} = \{(I_1, I'_1, C'_1), \ldots, (I_n, I'_n, C'_n)\}$, where I_j is the input image that defines the world coordinate, I'_j is the image when viewing the scene from camera C'_j, we maximize the log-

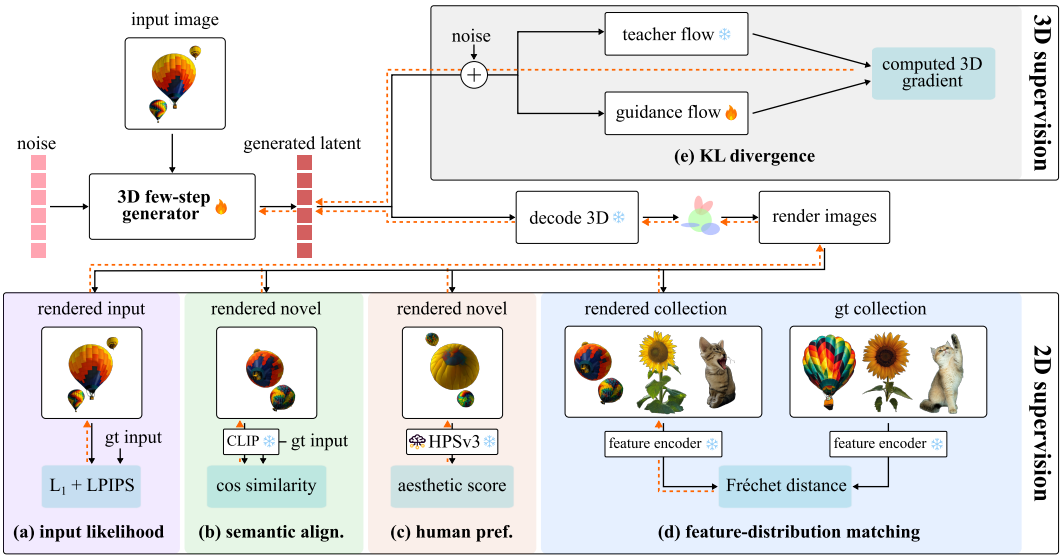

Figure 2: **Overview of our framework.** We propose 2D image supervision for improving the quality of a few-step image-to-3D generator. We increase the likelihood of the input image via a reconstruction loss **(a)**, align the semantics of novel views and the input view via CLIP (Radford et al., 2021) similarity **(b)**, maximize human preference score (Ma et al., 2025) **(c)**, and align the overall rendered image distributions via Fréchet distance (Yang et al., 2026) **(d)**. Simultaneously, to avoid significantly diverging from 3D priors, we maintain a KL-divergence term via DMD (Yin et al., 2024b) with a multi-step 3D teacher flow model **(e)**. Image credits: (Gardner, 2019; Gill, 2023; Green, 2016; Burden, 2020; Loan, 2019; Nee, 2020).

likelihood of the rendered I' from camera C':

$$\log p(I' \mid I, C') = \log \int p(I', \mathbf{z} \mid I, C')\, \mathrm{d}\mathbf{z} = \log \int p(I' \mid \mathbf{z}, C')\, p(\mathbf{z} \mid I)\, \mathrm{d}\mathbf{z}$$
$$\geq \int q_\theta(\mathbf{z} \mid I)\, \log p(I' \mid \mathbf{z}, C')\, \mathrm{d}\mathbf{z} - D_{\mathrm{KL}}\left(q_\theta(\mathbf{z} \mid I) \parallel p(\mathbf{z} \mid I)\right), \tag{1}$$

where $p(\mathbf{z} \mid I)$ is the prior distribution of the 3D latent given the input image, implemented via a teacher image-to-3D generator, $q_\theta(\mathbf{z} \mid I)$ is the image-to-3D generator we are training, $p(I \mid \mathbf{z}, C) = \mathcal{N}(I; f(\mathbf{z}, C), \sigma^2)$ models the distribution of the rendered image given 3D and camera, and $f(\mathbf{z}, C)$ is the deterministic process of decoding the 3D latent $\mathbf{z}$ into 3DGS and render from the camera C. Intuitively, the first term in Eq. (1) enables us to optimize the image-to-3D model q_θ by comparing rendered images while the KL divergence term makes sure q_θ generate meaningful 3D latents instead of creating 2D billboards.

We intentionally choose $q_\theta(\mathbf{z} \mid I)$ (instead of $q_\theta(\mathbf{z} \mid I, I', C')$), which enables us to optimize the single image-to-3D model q_θ. However, as q_θ is unaware of I' and C', there can be multiple possibilities for I' when C' sees unobserved regions in the input image I. This makes $p(I' \mid \mathbf{z}, C')$ low and the lower-bound loose. An ideal generated 3D object only guarantees the rendered input view should look like the input image (*i.e.*, the input image has high likelihood when evaluating $p(I' \mid \mathbf{z}, C')$). Thus, we use the input camera as C'. This ensures the generated object $\mathbf{z}$ look like the input image I when rendered from the input viewpoint. In practice, instead of using the ℓ_2 loss (which assumes $p(I \mid \mathbf{z}, C)$ is a Normal distribution), which tends to cause blurry images, we use a combination of ℓ_1 and LPIPS losses (Zhang et al., 2018; Johnson et al., 2016). Specifically, let I_0 and A_0 be the RGB image and alpha mask of the input conditioning view, and $\hat{I}_0$ and $\hat{A}_0$ be the corresponding renderings of the generated object. We have

$$\mathcal{L}_{\mathrm{rec}} = \frac{1}{hw} \|I_0 - \hat{I}_0\|_1 + \lambda_{\mathrm{lpips}}\, \mathrm{LPIPS}(I_0, \hat{I}_0) + \frac{1}{hw} \|A_0 - \hat{A}_0\|_1 + \lambda_{\mathrm{lpips}}\, \mathrm{LPIPS}(A_0, \hat{A}_0), \tag{2}$$

where we set $\lambda_{\mathrm{lpips}} = 0.2$.

The KL divergence $D_{\mathrm{KL}}\left(q_\theta(\mathbf{z} \mid I) \parallel p(\mathbf{z} \mid I)\right)$ is difficult to compute in general; but in our case $q_\theta(\mathbf{z} \mid I)$ is a deterministic function, $G_\theta(\boldsymbol{\epsilon})$, that push-forwards an initial noise $\boldsymbol{\epsilon}$, we can compute its gradient following Yin et al. (2024b) and Wang et al. (2023):

$$\nabla_\theta D_{\mathrm{KL}}\left(q_\theta(\mathbf{z} \mid I) \parallel p(\mathbf{z} \mid I)\right) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\\ \mathbf{z}=G_\theta(\boldsymbol{\epsilon})}} \left(\nabla_{\mathbf{z}} \log p(\mathbf{z} \mid I) - \nabla_{\mathbf{z}} \log q_\theta(\mathbf{z} \mid I)\right) \frac{\partial G_\theta(\boldsymbol{\epsilon})}{\partial \theta}, \tag{3}$$

where the expectation is estimated by 1-sample Monte Carlo estimation. In practice, we follow Yin et al. (2024b) and compute the KL divergence between the noised distributions, under the noising schedule of the teacher flow matching model, $p(\mathbf{z} \mid I)$, which allows us to compute the score function $\nabla_{\mathbf{z}} \log p^t(\mathbf{z} \mid I)$. We estimate the score function $\nabla_{\mathbf{z}} \log q_\theta^t(\mathbf{z} \mid I)$ by learning an additional auxiliary flow matching model $q_\phi^t(\mathbf{z} \mid I)$ on the generated $\mathbf{z}$ sampled from $q_\theta(\mathbf{z} \mid I)$. In Section 4 (Table 2), we ablate the effect of the conditioning image reconstruction loss and the KL divergence terms in Eq. (1), and we provide the derivation of Eq. (3) in Section A.

3.2 Additional 2D supervisions via reinforcement learning framework

In addition to the variational bound perspective, Eq. (1) can also be interpreted as a KL-regularized reward maximization, where the reward is computed as the negative reconstruction loss of the input image. This perspective allows us to use additional 2D supervision beyond log-likelihood of 2D images. Any such supervision must respect the ambiguity of image-to-3D generation: the back side of an object is not determined by its front view, and the ground-truth sample in the training set is just one plausible completion among many. This requirement precludes pixel-wise reconstruction losses on hidden views, that would penalize valid generations for differing from an arbitrary reference. We instead seek rewards that assess plausibility, and propose three possibilities below.

Feature distribution matching. The *distribution* of the rendered novel views from generated objects and real objects should match. This observation motivates us to use Fréchet distance (Yang et al., 2026) on the image features as a loss. Specifically, given a feature extractor, ϕ_e, we generate n 3D objects and render k views randomly for each object; we then compute the mean and covariance of the nk extracted image features. We do the same for real objects in the dataset while using the same input image and views as the generated ones. Let the means and covariance of features from the generated and the real objects be $(\hat{\boldsymbol{\mu}}_e, \hat{\boldsymbol{\Sigma}}_e)$ and $(\boldsymbol{\mu}_e, \boldsymbol{\Sigma}_e)$, respectively. We minimize the Fréchet distance, which is computed as

$$\mathcal{D}_e = \|\hat{\boldsymbol{\mu}}_e - \boldsymbol{\mu}_e\|_2^2 + \mathrm{Tr}\left(\hat{\boldsymbol{\Sigma}}_e + \boldsymbol{\Sigma}_e - 2\left(\hat{\boldsymbol{\Sigma}}_e \boldsymbol{\Sigma}_e\right)^{1/2}\right). \tag{4}$$

In practice, we use MAE (He et al., 2022), SigLIP (Zhai et al., 2023), and CLIP (Radford et al., 2021) as feature extractors, for which we compute separate μ and Σ via EMA-based estimators for computational efficiency (Yang et al., 2026). We apply the distribution matching separately to RGB and alpha maps. The resulting loss is

$$\mathcal{L}_{\mathrm{FD}} = \sum_{e\in\mathcal{E}} \left(\mathcal{D}_e^{\mathrm{rgb}} + \lambda_\alpha \mathcal{D}_e^{\alpha}\right), \tag{5}$$

where $\mathcal{E} = \{\mathrm{MAE}, \mathrm{SigLIP2}, \mathrm{CLIP}\}$, and we use $\lambda_\alpha = 0.1$ in our experiments. Note that we intentionally do not use Inception as a feature, as FID is used commonly as an evaluation metric. This ensures we do not optimize FID and the evaluation is fair.

Human-preference supervision. Fréchet distance provides macroscopic, distribution-level guidance, but it does not directly provide sample-level supervision. To this end, while we cannot directly use reconstruction losses to supervise novel views, we can encourage novel views to become generically higher quality according to some reference-free metric. For this purpose, we adopt the output of a human preference score model, HPSv3 (Ma et al., 2025), as another reward. Rather than maximizing the HPSv3 score, we use a paired preference model that compares the score of the same novel view between the generated and the reference object in the training set. Let $h(\cdot)$ be the HPSv3 output logit under a shared prompt. We maximize the log-likelihood of a human preferring the generated view over the reference view averaged across $\mathcal{V}_{\mathrm{pref}}$ views:

$$\mathcal{L}_{\mathrm{pref}} = \frac{1}{|\mathcal{V}_{\mathrm{pref}}|} \sum_{j\in\mathcal{V}_{\mathrm{pref}}} \mathrm{softplus}\left(h(I_j^{\mathrm{ref}}) - h(\hat{I}_j)\right), \tag{6}$$

where $\mathrm{softplus}(x) = \log(1 + \exp(x))$. Intuitively, the loss encourages the generated rendering to approach the perceptual quality of the corresponding reference rendering, rather than maximizing the absolute preference score, which is unbounded and prone to reward hacking.

Semantic consistency. Fréchet distance and human-preference score improve the overall quality of novel views, but they are computed agnostically to the actual input view. To fill this gap, we can encourage the novel views to be semantically similar to the input view, following recent work in 3D large reconstruction models (Jiang et al., 2024; Reddy et al., 2024). Let ϕ_{CLIP} be the CLIP image encoder and $\mathcal{V}_{\mathrm{novel}}$ be the rendered novel views. We maximize a threshold CLIP cosine similarity between the novel view and the input view:

$$\mathcal{L}_{\mathrm{sem}} = \frac{1}{|\mathcal{V}_{\mathrm{novel}}|} \sum_{j \in \mathcal{V}_{\mathrm{novel}}} [m_{\mathrm{sem}} - s_j]_+, \qquad s_j = \frac{\phi_{\mathrm{CLIP}}(I_0)^\top \phi_{\mathrm{CLIP}}(\hat{I}_j)}{\|\phi_{\mathrm{CLIP}}(I_0)\|_2 \|\phi_{\mathrm{CLIP}}(\hat{I}_j)\|_2}, \tag{7}$$

where $[x]_+ = \max(x, 0)$, and we use $m_{\mathrm{sem}} = 0.9$, which is set by computing CLIP similarities between real novel and input views. The margin help stop penalizing a novel view once its semantic agreement with the input image reaches the desired level without requiring novel-views to look exactly like the input view, avoiding any potential Janus problems.

3.3 Overall objective

We minimize the total loss:

$$\mathcal{L} = \lambda_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{FD}}\mathcal{L}_{\mathrm{FD}} + \lambda_{\mathrm{pref}}\mathcal{L}_{\mathrm{pref}} + \lambda_{\mathrm{sem}}\mathcal{L}_{\mathrm{sem}} + \lambda_{\mathrm{kl}}D_{\mathrm{KL}}, \tag{8}$$

where λ_* are the loss weights controlling the contribution of each objective. As all rewards / losses are differentiable, the gradients are propagated through reward functions, Gaussian splatting, and the frozen 3DGS decoder to the trainable generator. In Section 4, we ablate the effect of each loss.

4 Experiments

In the section, we first discuss our training data and evaluation protocol. We then conduct series of ablation studies to analyze the effect of reconstructing the input view, the KL regularization, and individual 2D rewards. Finally, we compare our final model with existing baselines.

4.1 Training setting and evaluation protocol

We base our implementation on LiTo (Chang et al., 2026) — we use their pretrained 3D latent encoder and decoder and follow their training settings, including data, object normalization, and camera placements. To fairly compare with existing baselines, we use a consistent input image resolution (518×518).

Dataset. We train our models on Objaverse-XL (Deitke et al., 2023), using the same training, validation, and test splits as LiTo (Chang et al., 2026). The data is based on the 500 thousand samples filtered by Xiang et al. (2025b), but unlike them, who used all of them as training data, we split them into training, validation, and test sets with 8:1:1 ratios. For numerical evaluation, we utilize 1000 objects from the test set. For each object, one view is provided to the model as the conditioning image, while the remaining views are used to evaluate the generated 3D object from unseen viewpoints.

Training details. We first train a base MeanFlow model (Geng et al., 2026a; Zhang et al., 2026) that will be used as initialization of our single- and 4-step models. For the four-step setting, we additionally experiment with a DMD-distilled LiTo model. We adopted a few improvements from recent literature (Singh et al., 2026) to the original LiTo model. Please refer to Section B for details about the LiTo and base models. Note that we compare against the improved LiTo model.

After the base models are trained, we finetune them with our proposed loss in Section 3. 2D losses are calculated by rendering Gaussians decoded from generated latents. When training with the KL divergence, we train the auxiliary flow matching model, $q_\phi^t(\mathbf{z} \mid I)$, every iteration by sampling from

$q_\theta(\mathbf{z} \mid I)$, and we train $q_\theta(\mathbf{z} \mid I)$ every 10 iterations. We use the improved LiTo model for the flow matching teacher $\nabla_{\mathbf{z}} \log p^t(\mathbf{z} \mid I)$ with a CFG scale of 3.

We use ADAMW (Loshchilov & Hutter, 2019) for optimization. All models use a 2000-step linear warmup from 1×10^{-8}. The auxiliary flow matching model warms up to a peak learning rate of 1×10^{-5} in both settings. For the generator, the peak learning rate is 1×10^{-5} for the MeanFlow-initialized model and 1×10^{-6} for the DMD-initialized model. Unless otherwise noted, all training was performed with a batch size of 64. In the four-step setting, to reduce compute requirements and avoid gradient explosion, we rely on DRaFT-1 (Clark et al., 2024) for stable and efficient training.

Metrics. We report Fréchet Inception Distance (FID) and Kernel Inception Distance (KID) for conditioning and novel views separately (Parmar et al., 2022). Note that Inception is not used in our training objective to avoid corrupting the evaluation metrics. We also report LPIPS (Zhang et al., 2018; Johnson et al., 2016) between the rendered conditioning view and the input view. We do not use LPIPS for novel views because a single conditioning image does not uniquely determine the appearance of unobserved regions. For novel views, we also report PickScore (Kirstain et al., 2023), which predicts human preferences, and SigLIP2 cosine similarity, which measures semantic consistency to the input view. As our models share the same latent space as LiTo, we additionally report the log-likelihood of the generated 3D latents computed by the multi-step teacher LiTo model using the instantaneous change-of-variable method (Chen et al., 2018). The log-likelihood of LiTo latents measures the quality of 3D geometry and view-dependent appearance. For readability, we scale LPIPS and KID values by 1e-2 and 1e-3, respectively.

4.2 Effect of 2D image supervision

Loss ablation. To start, we first study how each component of our image-level supervision contributes to performance. We run a small ablation in the 1-step setting at 48k training iterations, with results shown in Table 2.

First, we show the impact of using a conditioning image reconstruction loss alone and when combined with KL-divergence minimization. Conditioning loss alone can be seen to significantly improve input view performance, but overall structure deteriorates as illustrated by the drop in novel-view performance and the log likelihood of real 3D latents. Incorporating the KL term with the conditioning image reconstruction loss mitigates the issues, while preserving most performance gain in input image consistency. It also outperforms training with the KL term alone.

Next, we evaluate the impact of feature distribution matching, human-preference, and semantic consistency. Both feature distribution matching and semantic consistency significantly improve the performance of novel views, while human-preference supervision primarily seems to improve preference metrics. None of these additional rewards significantly impact input view fidelity, but slightly push the generated 3D latents out of the original LiTo distribution.

Finally, we show the performance of our final recipe (see Section B). It can be seen that combining all of our proposed rewards yields a model with the best performance.

Distillation method comparison. Next, we compare our recipe against alternative approaches for obtaining few-step models. In the 1-step setting (Table 3(a)), we benchmark a 48k iteration checkpoint against MeanFlow (Geng et al., 2026a) and DMD (Yin et al., 2024b) trained for a matching number of iterations. Applying our supervision on top of a MeanFlow initialization consistently improves over both baselines despite similar training time. We observe the same trend in the 4-step setting (Table 3(b)), where we additionally apply our 2D supervision to a DMD base model. At 128k training iterations, this variant yields slightly worse novel-view but better input-view metrics compared to a 4-step MeanFlow base model trained under the same losses, and both surpass MeanFlow-only and DMD-only training. Our supervision can be seen to be complementary to the choice of base distillation method rather than tied to a particular one, and applicable to models that use more than a single step.

Table 1: **Quantitative comparison of few-step 3D generation methods**. The runtime measures everything from image to 3D latents on an B200 GPU.

	input view		novel view		preference		alignment			runtime
method	fid↓	lpips↓	fid↓	kid↓	pick↑	hps↑	siglip↑	clip↑	uni3d↑	(ms) ↓
(a) 1-step										
ours	27.11	11.06	13.14	1.64	19.67	3.872	0.887	0.906	0.381	67.5
SAM3D-Distill (Chen et al., 2026)	56.93	17.26	28.35	7.13	19.43	2.616	0.844	0.858	0.360	312.3
MDT-dist (Zhou et al., 2026)	56.79	18.30	24.92	6.36	19.53	3.184	0.855	0.866	0.355	150.3
(b) 4-step										
ours (DMD init)	26.48	11.42	10.77	1.18	19.61	3.912	0.878	0.896	0.381	222.3
ours (MF init)	28.58	12.07	11.40	1.00	19.68	3.946	0.883	0.901	0.381	249.0
SAM3D-Distill (Chen et al., 2026)	49.35	14.78	21.39	4.82	19.52	3.780	0.868	0.878	0.385	713.6
MDT-dist (Zhou et al., 2026)	48.66	17.13	17.29	3.61	19.54	3.663	0.871	0.883	0.385	458.9
(c) 25-step										
LiTo (Chang et al., 2026)	25.90	10.53	10.06	1.05	19.64	4.123	0.884	0.900	0.395	4999.8
SAM3D (Chen et al., 2026)	46.09	13.79	18.36	3.46	19.59	4.047	0.874	0.887	0.392	3824.8
TRELLIS (Xiang et al., 2025b)	43.26	16.40	12.48	1.31	19.59	4.348	0.878	0.890	0.390	2772.6
Fast-SAM3D (Feng et al., 2026)	48.28	14.87	17.94	2.98	19.46	2.898	0.869	0.877	0.385	1833.1

Table 2: **Small-scale ablation study of different losses in a 1-step setting.** "cr" is short for "conditioning reconstruction loss". "clip", "fd" and "hps" refer to our proposed semantic consistency, feature-distribution matching and human-preference supervision rewards, respectively.

	input view		novel view		preference		alignment		latent
method	fid↓	lpips↓	fid↓	kid↓	pick↑	hps↑	siglip↑	clip↑	nll↓
base	73.50	15.20	55.14	27.26	19.38	1.068	0.843	0.851	-0.294
+ cr	27.93	10.80	37.99	16.00	19.59	2.248	0.861	0.879	0.123
+ kl	38.98	13.17	25.82	8.27	19.47	2.370	0.868	0.882	-0.796
+ cr + kl	33.71	12.00	26.10	8.13	19.53	2.634	0.866	0.884	-0.796
+ cr + kl + clip	32.47	11.70	18.81	4.70	19.60	3.076	0.875	0.896	-0.735
+ cr + kl + fd	31.89	11.82	19.03	4.78	19.58	3.109	0.872	0.891	-0.746
+ cr + kl + hps	37.82	12.15	25.97	8.09	19.55	2.934	0.867	0.882	-0.681
all	31.71	11.83	16.38	3.25	19.60	3.336	0.878	0.896	-0.704

4.3 Comparison with state-of-the-art

Models and baselines. For our 1-step model, we run our proposed supervision on a MeanFlow base for 170k iterations with a batch size of 256. Our 4-step models use a batch size of 64, with 128k and 196k iterations on the MeanFlow and DMD base models, respectively. For few-step baselines, we compare with MDT-dist (Zhou et al., 2026) and SAM3D-Distill (Chen et al., 2026). To get a sense of upper-bound performance, we also evaluate the multi-step generative models of the improved LiTo model (Chang et al., 2026), the full SAM3D (Chen et al., 2026), Fast-SAM3D (Feng et al., 2026), and TRELLIS (Xiang et al., 2025b). For the two-stage SAM3D/Fast-SAM3D/MDT-dist/TRELLIS pipelines, the reported number of steps is applied to both generative stages. For these methods, we additionally perform post-generation optimization to mitigate potential object pose misalignment. The operation improves evaluation metrics significantly, and we do not include the optimization runtime. See Section C for details.

Quantitative results. Table 1(a) shows quantitative evaluation results in the 1-step setting. Our 1-step model can be seen to clearly outperform the baselines across all metrics, even on hold-out metrics not seen during training, while running more than twice as fast. A similar trend holds in the 4-step setting in Table 1(b) — both our DMD and MeanFlow-initialized models significantly outperform the baselines across a vast majority of metrics in a fraction of the runtime. Interestingly, our 1-step model boasts better input-view LPIPS and alignment scores compared to our 4-step variants; we attribute this to the computational and numerical difficulties of multi-step training, which we currently tackle by training only the last step (Clark et al., 2024).

Table 3: **Small-scale comparison of different methods for few-step compression.**

	input view		novel view		preference		alignment		latent
method	fid↓	lpips↓	fid↓	kid↓	pick↑	hps↑	siglip↑	clip↑	nll↓
(a) 1-step									
meanflow (Geng et al., 2026a)	73.50	15.20	55.14	27.26	19.38	1.068	0.843	0.851	-0.294
DMD (Yin et al., 2024b)	253.4	20.41	210.6	155.7	19.01	-6.632	0.655	0.731	-0.300
ours (meanflow init)	31.71	11.83	16.38	3.25	19.60	3.336	0.878	0.896	-0.704
(b) 4-step									
meanflow (Geng et al., 2026a)	34.31	11.95	16.52	3.48	19.59	3.182	0.873	0.892	-1.052
DMD (Yin et al., 2024b)	27.85	11.40	15.39	3.86	19.59	3.488	0.872	0.891	-1.245
ours (meanflow init)	28.58	12.07	11.40	1.00	19.68	3.946	0.883	0.901	-1.167
ours (DMD init)	25.51	10.62	11.97	1.87	19.61	3.740	0.879	0.896	-1.285



Figure 3: **Qualitative comparison of 1-step 3D generation methods on real image input**. We compare with MDT-dist (Zhou et al., 2026) and SAM3D-Distill (Chen et al., 2026). Our method captures more structure and detail than the baselines. Image credits: (Sorge, 2016; Vakhidov, 2026).

Qualitative results. Figure 3 shows qualitative comparisons in the single-step setting on real photos. MDT-dist and SAM3D-Distill exhibit significant structural changes, noticeable color drift and substantial loss of fine texture detail, particularly for objects with complex geometry and appearance. In contrast, our approach is faithful to the conditioning image while maintaining consistency in unseen views. We show more results in Figure 4 and Figure 5 in the appendix, where we also show 4-step outputs from the few-step baselines and 20-step generations from the improved LiTo model, SAM3D and TRELLIS models. Our method can be seen to maintain a better input fidelity compared to the baselines. **Please see the supplemental website for video visualizations.**

5 Limitations

While effective, our proposed image supervision induces additional computational training cost over standard few-step training. It requires decoding a 3D asset, rendering the output, and computing image-space rewards. However, the memory costs can be mitigated via gradient checkpointing, and this overhead is confined to training: the resulting model has identical architecture and inference cost as the base model. While the 4-step FLiTo achieves close FID to the 25-step teacher LiTo model, there is still a quality gap between the 1-step FLiTo model and the teacher. Fully recovering the quality of a multi-step 3D generator under highly compressed sampling remains an open problem. Closing this gap could require more sophisticated forms of 2D supervision, such as adversarial supervision (Chan et al., 2022; Yin et al., 2024a) or powerful 2D model distillation (Poole et al., 2022; Wang et al., 2023). We leave these directions as interesting future work.

AI USE STATEMENT

In this work, we used generative AI tools to improve writing, search for related works, edit software code, and suggest experimental parameters. We have not used generative AI tools for anything else (*e.g.*, data generation and evaluation), and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. For example, LLM-generated code was verified and tested for correctness, and writing improvements are verified by all authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Titas Anciukevičius, Zexiang Xu, Matthew Fisher, Paul Henderson, Hakan Bilen, Niloy J Mitra, and Paul Guerrero. Renderdiffusion: Image diffusion for 3d reconstruction, inpainting and generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12608–12618. IEEE, 2023.

Aaron Burden. Sunflower field under blue sky during daytime, 2020. URL https://unsplash.com/photos/sunflower-field-under-blue-sky-during-daytime-2IzoIHBgYAo. Licensed under the Unsplash License.

Yuanhao Cai, He Zhang, Kai Zhang, Yixun Liang, Mengwei Ren, Fujun Luan, Qing Liu, Soo Ye Kim, Jianming Zhang, Zhifei Zhang, et al. Baking gaussian splatting into diffusion denoiser for fast and scalable single-stage image-to-3d generation and reconstruction. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 25062–25072. IEEE, 2025.

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16123–16133, 2022.

Jen-Hao Chang, Xiaoming Zhao, Dorian Chan, and Oncel Tuzel. Lito: Surface light field tokenization. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 100644–100671, 2026.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. Sam 3d: 3dfy anything in images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7220–7232, 2026.

Kevin Clark, Paul Vicol, Kevin Swersky, and David Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *International Conference on Learning Representations*, volume 2024, pp. 4793–4822, 2024.

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 (NeurIPS)*, 36:35799–35813, 2023.

Weilun Feng, Mingqiang Wu, Zhiliang Chen, Chuanguang Yang, Haotong Qin, Yuqi Li, Xiaokun Liu, Guoxin Fan, Libo Huang, Yulun Zhang, et al. Fast-sam3d: 3dfy anything in images but faster. *arXiv preprint arXiv:2602.05293*, 2026.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, volume 2025, pp. 34668–34684, 2025.

Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:31841–31854, 2022.

Jesse Gardner. Red and multicolored hot air balloon, 2019. URL https://unsplash.com/photos/red-and-multicolored-hot-air-balloon-dSEE5q7vCgA. Licensed under the Unsplash License.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems (NeurIPS)*, 38:75460–75482, 2026a.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30467–30476, 2026b.

Triyansh Gill. A group of hot air balloons flying through a blue sky, 2023. URL https://unsplash.com/photos/a-group-of-hot-air-balloons-flying-through-a-blue-sky-DI9qmqje6Yo. Licensed under the Unsplash License.

Paul Green. Close-up photo of common sunflower, 2016. URL https://unsplash.com/photos/close-up-photo-of-common-sunflower-5lRxNLHfZOY. Licensed under the Unsplash License.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15979–15988. IEEE, 2022.

Hanwen Jiang, Qixing Huang, and Georgios Pavlakos. Real3d: Scaling up large reconstruction models with real-world images. *arXiv preprint arXiv:2406.08479*, 2024.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, pp. 694–711. Springer, 2016.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:36652–36663, 2023.

Zeqiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Qingxiang Lin, Jingwei Huang, Chunchao Guo, and Xiangyu Yue. LATTICE: Democratize high-fidelity 3d generation at scale. *arXiv preprint arXiv:2512.03052*, 2025a.

Zeqiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Fuyun Wang, Huiwen Shi, Xianghui Yang, Qingxiang Lin, Jingwei Huang, Yuhong Liu, et al. Unleashing vecset diffusion model for fast shape generation. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 2523–2533. IEEE, 2025b.

Zhihao Li, Yufei Wang, Heliang Zheng, Yihao Luo, and Bihan Wen. Sparc3d: Sparse representation and construction for high-resolution 3d shapes modeling. *arXiv preprint arXiv:2505.14521*, 2025.

Loan. Silver tabby kitten on floor, 2019. URL https://unsplash.com/photos/silver-tabby-kitten-on-floor-7AIDE8PrvA0. Licensed under the Unsplash License.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yuhang Ma, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. Hpsv3: Towards wide-spectrum human preference score. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15086–15095. IEEE, 2025.

Matvei. Elegant still life with tangerines and water jug, 2026. URL https://www.pexels.com/photo/elegant-still-life-with-tangerines-and-water-jug-37511454/. Licensed under the Pexels License.

Alvan Nee. White and brown long fur cat, 2020. URL https://unsplash.com/photos/white-and-brown-long-fur-cat-ZCHj_2lJP00. Licensed under the Unsplash License.

White Noiise. A white and black eagle sitting on a bench, 2024. URL https://www.pexels.com/photo/a-white-and-black-eagle-sitting-on-a-bench-28267590/. Licensed under the Pexels License.

Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

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. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 28707–28717. IEEE, 2025.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

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 (ICML)*, pp. 8748–8763. PmLR, 2021.

Pradyumna Reddy, Ismail Elezi, and Jiankang Deng. G3dr: Generative 3d reconstruction in imagenet. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9655–9665. IEEE, 2024.

Ahan Shabanov, Peter Hedman, Ethan Weber, Zhengqin Li, Denis Rozumny, Gael Le Lan, Naina Dhingra, Lei Luo, Andrea Vedaldi, Christian Richardt, et al. Free-range gaussians: Non-grid-aligned generative 3d gaussian reconstruction. *arXiv preprint arXiv:2604.04874*, 2026.

Mayank Singh, Michele Stoppa, Alvise Memo, Rui Yu, Harsha Kalli, Srimanth Gunturi, Muhammad Ahmed Riaz, Behrooz Shahsavari, Waleed Abdulla, and David E Jacobs. Luce: Relightable gaussians for 3d asset generation. *arXiv preprint arXiv:2608.23943*, 2026.

Ivan Skorokhodov, Aliaksandr Siarohin, Yinghao Xu, Jian Ren, Hsin-Ying Lee, Peter Wonka, and Sergey Tulyakov. 3d generation on imagenet. *arXiv preprint arXiv:2303.01416*, 2023.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning (ICML)*, 2023.

Roberta Sorge. Clear glass bottle, 2016. URL https://unsplash.com/photos/clear-glass-bottle-kp9UVn-PUac. Licensed under the Unsplash License.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Nurkamol Vakhidov. Mulberry tree branch against clear blue sky, 2026. URL https://www.pexels.com/photo/mulberry-tree-branch-against-clear-blue-sky-37443702/. Licensed under the Pexels License.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in neural information processing systems*, 36:8406–8441, 2023.

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Jingxi Xu, Philip Torr, Xun Cao, and Yao Yao. Direct3d: Scalable image-to-3d generation via 3d latent diffusion transformer. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:121859–121881, 2024.

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Yikang Yang, Yajie Bao, Jiachen Qian, Siyu Zhu, Xun Cao, Philip Torr, et al. Direct3d-s2: Gigascale 3d generation made easy with spatial sparse attention. *arXiv preprint arXiv:2505.17412*, 2025.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation. *Tech report*, 2025a.

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. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480. IEEE, 2025b.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, et al. Native and compact structured latents for 3d generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14419–14429, 2026.

Jiawei Yang, Zhengyang Geng, Xuan Ju, Yonglong Tian, and Yue Wang. Representation fr\'echet loss for visual generation. *arXiv preprint arXiv:2604.28190*, 2026.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:47455–47487, 2024a.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. *URL https://arxiv.org/abs/2311.18828*, 3, 2024b.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 11941–11952. IEEE, 2023.

Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions On Graphics (TOG)*, 42(4):1–16, 2023.

Huijie Zhang, Aliaksandr Siarohin, Willi Menapace, Michael Vasilkovsky, Sergey Tulyakov, Qing Qu, and Ivan Skorokhodov. Alphaflow: Understanding and improving meanflow models. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 144056–144094, 2026.

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. Clay: A controllable large-scale generative model for creating high-quality 3d assets. *ACM Transactions On Graphics (TOG)*, 43(4):1–20, 2024.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595. IEEE, 2018.

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3D 2.0: Scaling Diffusion Models for High Resolution Textured 3D Assets Generation. *arXiv*, 2025.

Zanwei Zhou, Taoran Yi, Jiemin Fang, Chen Yang, Lingxi Xie, Xinggang Wang, Wei Shen, and Qi Tian. Few-step flow for 3d generation via marginal-data transport distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 13853–13861, 2026.

APPENDIX

A DERIVATION OF GRADIENT OF KL DIVERGENCE

Here, we derive Eq. (3).

$$\mathrm{KL}\left(q_\theta(\mathbf{z} \mid I) \parallel p(\mathbf{z} \mid I)\right) = \mathop{\mathbb{E}}_{\mathbf{z}\sim q_\theta(\mathbf{z}|I)} \log q_\theta(\mathbf{z} \mid I) - \log p(\mathbf{z} \mid I) \tag{9}$$

$$= \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \log q_\theta(G_\theta(\boldsymbol{\epsilon}) \mid I) - \log p(G_\theta(\boldsymbol{\epsilon}) \mid I). \tag{10}$$

Taking the gradient with respect to θ gives

$$\boldsymbol{\nabla}_\theta D_{\mathrm{KL}}\left(q_\theta \parallel p\right) = \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}} \boldsymbol{\nabla}_\theta \log q_\theta(G_\theta(\boldsymbol{\epsilon}) \mid I) - \boldsymbol{\nabla}_\theta \log p(G_\theta(\boldsymbol{\epsilon}) \mid I). \tag{11}$$

Using the chain rule,

$$\boldsymbol{\nabla}_\theta \log q_\theta(G_\theta(\boldsymbol{\epsilon}) \mid I) = \boldsymbol{\nabla}_\theta \log q_\theta(\mathbf{z} \mid I)\,|_{\mathbf{z}=G_\theta(\boldsymbol{\epsilon})} + \boldsymbol{\nabla}_\mathbf{z} \log q_\theta(\mathbf{z} \mid I)\, \frac{\partial G_\theta(\boldsymbol{\epsilon})}{\partial \theta}, \tag{12}$$

$$\boldsymbol{\nabla}_\theta \log p(G_\theta(\boldsymbol{\epsilon}) \mid I) = \boldsymbol{\nabla}_\mathbf{z} \log p(\mathbf{z} \mid I) \frac{\partial G_\theta(\boldsymbol{\epsilon})}{\partial \theta}, \tag{13}$$

since $p(\mathbf{z} \mid I)$ is independent of θ.

Therefore,

$$\boldsymbol{\nabla}_\theta D_{\mathrm{KL}}\left(q_\theta \mid p\right) = \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}}\left[\boldsymbol{\nabla}_\theta \log q_\theta(\mathbf{z} \mid I)|_{\mathbf{z}=G_\theta(\boldsymbol{\epsilon})} + \left(\boldsymbol{\nabla}_\mathbf{z} \log q_\theta(\mathbf{z} \mid I) - \boldsymbol{\nabla}_\mathbf{z} \log p(\mathbf{z} \mid I)\right) \frac{\partial G_\theta(\boldsymbol{\epsilon})}{\partial \theta} \right]. \tag{14}$$

The first term vanishes in expectation due to the score-function identity:

$$\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}} \boldsymbol{\nabla}_\theta \log q_\theta(\mathbf{z} \mid I)|_{\mathbf{z}=G_\theta(\boldsymbol{\epsilon})} = \mathop{\mathbb{E}}_{\mathbf{z}\sim q_\theta} \left[\boldsymbol{\nabla}_\theta \log q_\theta(\mathbf{z} \mid I)\right] \tag{15}$$

$$= \int q_\theta(\mathbf{z} \mid I) \boldsymbol{\nabla}_\theta \log q_\theta(\mathbf{z} \mid I)\, \mathrm{d}\mathbf{z} = \int \frac{q_\theta(\mathbf{z} \mid I)}{q_\theta(\mathbf{z} \mid I)} \boldsymbol{\nabla}_\theta q_\theta(\mathbf{z} \mid I)\, \mathrm{d}\mathbf{z} \tag{16}$$

$$= \boldsymbol{\nabla}_\theta \int q_\theta(\mathbf{z} \mid I)\, \mathrm{d}\mathbf{z} = \boldsymbol{\nabla}_\theta 1 = 0. \tag{17}$$

Hence,

$$\boxed{\boldsymbol{\nabla}_\theta D_{\mathrm{KL}}\left(q_\theta(\mathbf{z} \mid I) \parallel p(\mathbf{z} \mid I)\right) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\\ \mathbf{z}=G_\theta(\boldsymbol{\epsilon})}} \left[\left(\boldsymbol{\nabla}_\mathbf{z} \log q_\theta(\mathbf{z} \mid I) - \boldsymbol{\nabla}_\mathbf{z} \log p(\mathbf{z} \mid I)\right) \frac{\partial G_\theta(\boldsymbol{\epsilon})}{\partial \theta}\right].} \tag{18}$$

B MORE TRAINING DETAILS

Improved LiTo flow matching. We finetune the public LiTo generative model checkpoint (Chang et al., 2026), which has 28 blocks, 1152 feature dimension, 16 heads, and 4608-dimensional MLPs. We make two improvements, following recent advancement in the literature. First, instead of using just the final embeddings of DINOv2-large, we follow Luce (Singh et al., 2026) and incorporate intermediate feature as well — we concatenate the outputs of layer 4, 11, 17, and 23 along the channel dimension as input into the flow matching model. Second, inspired by SAM3D (Chen et al., 2026), we incorporate alpha as an additional input via the learnable linear patchification. We finetune for 945k iterations with a batch size of 256 with these adjustments. The two improvements provide additional information about the input image, and in our experiments, significantly improve upon the original LiTo model.

MeanFlow training. In practice, to stabilize training, we utilize AlphaFlow (Zhang et al., 2026), which uses a predetermined training curriculum from typical flow matching training, consistency, to MeanFlow training. We adopt LiTo's model architecture but replace all LayerNorm with RMSNorm, following Geng et al. (2026b), which we found stabilizes training. Like Geng et al. (2026b), we also input the start and end flow-matching times as input tokens. We slightly increase model size relative to the original LiTo flow matching architecture — we use 30 blocks, 1536 feature dimension, 24 heads, and 4608-dimensional MLPs. This raises the total parameter count from 626.7 to 727.0 million parameters (Table 5).

DMD. We distill the above improved LiTo flow-matching model via DMD (Yin et al., 2024b;a), with the absolute time-agnostic weighting used in the official DMD implementation. Following DMD2, we use an two time-scale update rule instead of a regression loss, with 10 auxiliary flow matching iterations for every generator iteration. We do not use an additional GAN discriminator on the generated latents; The teacher velocity is computed with a CFG scale of 3.

Loss weight details. For our final recipe, we used $\lambda_{\text{rec}} = 1$, $\lambda_{\text{FD}} = 1$, $\lambda_{\text{pref}} = 1 \times 10^{-3}$, $\lambda_{\text{sem}} = 0.255$, and $\lambda_{\text{kl}} = 1$. λ_{pref} and λ_{sem} were selected by calculating the gradient norm produced by each loss at iteration 0 of 2D supervision, and normalizing them relative to the gradient norm of feature-distribution matching.

C Evaluation details

We use the same evaluation set as LiTo (Chang et al., 2026), which contains 1000 unseen objects from ObjaverseXL. For each object (normalized from [-1, 1]), we randomly select a camera pose looking at the origin from 1.75 to 5.0 units away as the input conditioning view, with FOV set such that the object fills most of it. For evaluation, we render 8 views per object with cameras at radius 4 from the origin. Azimuths are uniformly spaced every $45°$, with a random jitter in $[-22.5°, 22.5°]$. The elevation is selected randomly from $[0°, 30°]$.

Inference settings. To fairly evaluate all baselines (LiTo (Chang et al., 2026), SAM3D (Chen et al., 2026), TRELLIS (Xiang et al., 2025b), MDT-dist (Zhou et al., 2026), Fast-SAM3D (Feng et al., 2026)), we use their released checkpoint (besides LiTo, which we use the checkpoint with the improvements mentioned in Section B) and their released inference settings (ODE sampler and classifier-free guidance (CFG) scales). Specifically, LiTo uses Heun with CFG scale = 3; SAM3D uses Euler with CFG scale = 8 for sparse structure generation and CFG scale = 2 for texture generation; SAM3D (distilled) uses the distillation mode without CFG and typical texture generation with Euler and CFG scale = 2; SAM3D (both distilled) uses Euler and without CFG; Fast-SAM3D follows SAM3D's setting with 25 steps; TRELLIS uses CFG scale = 6 for both sparse structure and slat generation; MDT-dist uses CFG scale = 6 for sparse structure generation and CFG scale = 2 for slat generation. SAM3D (distilled) is official distillation setting, as opposed to SAM3D (both distilled) (Chen et al., 2026). Note that inference settings affect number of function evaluations of individual methods — the relationship can be seen in Table 4. Thus we study the relationship between the generation quality and the runtime to generate the 3D latents by varying the number of ODE integration steps. As shown in Fig. 6, this analysis allows us to identify the tradeoff between generation quality and runtime of individual methods.

Post-generation pose optimization. We notice the relative pose between the generated object and the evaluation cameras can affect the evaluation results. This effect is significant for methods like TRELLIS and MDT-dist that generate objects in the object coordinate (and hence the generated objects do not align with the input images), as well as SAM3D and Fast SAM3D whose pose decoder may predict a different scale than that used by the evaluation data. We study the effect of pose optimization in Fig. 7 and Fig. 8.

For SAM3D and its variants, we maximize the intersection over union between the rendered input-view alpha mask and the given input alpha mask on the generated object. We use ADAM optimizer for 200 iterations with a fixed learning rate of 0.03. As can be seen in Fig. 7, the post-generation pose optimization improves novel-view FID significantly. Thus in the paper, we report their evaluation

results after post-generation pose optimization. We do not include the pose optimization into the reported runtime in the paper.

When evaluating TRELLIS and MDT-dist, we transform the evaluation cameras to the object coordinate using the input camera poses in the evaluation set. In other words, the evaluation results utilize the input camera pose, which a privileged information that is typically not known during inference. As can be seen in Fig. 8, the transformation improves both TRELLIS's and MDT-dist's evaluation results significantly. Thus, we report TRELLIS and MDT-dist results using the privileged information to reduce the effect of object pose mismatch.

Note that we only apply post-generation pose optimization and adjustments to SAM3D (and its distilled variants), Fast-SAM3D, TRELLIS, and MDT-dist. FLiTo and LiTo do not use any post-generation pose optimization nor have access to any privileged information.

D Runtime analysis

We measure the runtime of individual methods on a single B200, a single H100, and a single M5 Max. All methods use the same inference settings used during evaluation (Section C). To measure the runtime fairly, we measure the runtime of models after compilation when possible.[1] We use SAM3D's compilation implementation, as well as SAM3D distillation variants and Fast-SAM3D. As TRELLIS and MDT-dist do not have official compilation implementation, we additionally implement compilation for these methods. We are also careful about the data types. SAM3D, Fast-SAM3D, TRELLIS, and MDT-dist use single-precision model weights and computation as their implementation intended. LiTo and FLiTo use PyTorch's autocast to bfloat16.

As SAM3D, TRELLIS, MDT-dist rely on sparse convolution, we only measure the runtime of LiTo and FLiTo on M5 Max. We use MLX with float16 for LiTo and FLiTo. No model weight quantization or compilation are used on M5 Max.

The runtime analysis results is shown in Table 4. As can be seen, from image to 3D latent, 1-step FLiTo only needs 67.5 ms on B200, 92.2 ms on H100, and 2.4 seconds on M5 Max. This is $\sim 70\times$ speed improvement on all devices over LiTo.

E Additional experiments

In Fig. 4 and Fig. 5, we show qualitative results of our 1-step and 4-step model in relation to distillation baselines (Chen et al., 2026; Zhou et al., 2026) as well as various many-step methods (Chang et al., 2026; Xiang et al., 2025b; Chen et al., 2026). Our approach can be seen to match much of the detail and fine appearance that the base LiTo (Chang et al., 2026) captures with just a few steps. In contrast, SAM3D-Distill (Chen et al., 2026) and MDT-dist (Zhou et al., 2026) fail to capture much of the complexity.

We also run an additional subtractive ablation in Table 6. Dropping KL can be seen to improve all image metrics but substantively hurt latent negative log likelihood: this manifests qualitatively as loss of view-dependent appearance. HPSv3 (Ma et al., 2025) can also be seen to be difficult to optimize well.

[1] The exceptions are models whose compute graph is input-dependent, like the SLAT generator and the 3DGS of TRELLIS and SAM3D that depend on the number of occupied voxels.

Table 4: Runtime measured on 50 objects in Objaverse, on B200, H100, and M5 Max.

Method	NFE for Gen	Pointmap (ms)	Sparse Struct Gen (ms)	Latent Gen (ms)	**Total Gen (ms)**	3DGS Dec (ms)
NVIDIA B200						
FLiTo (1 step, euler)	1	–	–	67.5 ± 0.1	67.5 ± 0.1	80.3 ± 3.7
FLiTo (4 steps, euler)	4	–	–	222.3 ± 0.0	222.3 ± 0.0	84.2 ± 8.3
MeanFlow LiTo (1 step, euler)	1	–	–	67.3 ± 0.1	67.3 ± 0.1	77.9 ± 2.9
MeanFlow LiTo (4 steps, euler)	4	–	–	249.0 ± 0.1	249.0 ± 0.1	77.8 ± 2.0
MDT-dist (1 step, euler, cfg)	2 + 2	–	38.0 ± 0.9	112.2 ± 9.5	150.3 ± 9.4	16.7 ± 5.2
MDT-dist (4 steps, euler, cfg)	7 + 7	–	99.7 ± 0.8	359.2 ± 52.2	458.9 ± 52.1	13.5 ± 3.6
SAM3D (ss distilled, 1 step, euler, cfg slat)	1 + 2	55.1 ± 0.2	51.2 ± 3.2	206.0 ± 37.5	312.3 ± 37.4	157.5 ± 151.9
SAM3D (ss distilled, 4 steps, euler, cfg slat)	4 + 8	55.1 ± 0.2	136.6 ± 0.2	521.9 ± 71.7	713.6 ± 71.7	106.9 ± 74.9
SAM3D (both distilled, 1 step, euler)	1 + 1	55.2 ± 0.2	50.6 ± 0.2	139.5 ± 18.5	245.3 ± 18.5	150.3 ± 125.1
SAM3D (both distilled, 4 steps, euler)	4 + 4	55.2 ± 0.2	136.3 ± 0.2	329.6 ± 40.5	521.1 ± 40.5	107.3 ± 74.6
SAM3D (25 steps, euler, cfg)	75 + 50	55.2 ± 0.2	1194.5 ± 0.7	2575.1 ± 502.7	3824.8 ± 502.7	118.9 ± 101.1
Fast-SAM3D (25 steps, euler, cfg)	30 + 9.8	56.3 ± 0.9	1130.1 ± 13.2	646.7 ± 54.9	1833.1 ± 58.7	37.2 ± 21.6
TRELLIS (25 steps, euler, cfg)	50 + 50	–	558.9 ± 0.6	2213.8 ± 194.8	2772.6 ± 194.8	11.9 ± 2.3
LiTo (25 steps, heun, cfg)	100	–	–	4999.8 ± 0.2	4999.8 ± 0.2	77.6 ± 2.2
NVIDIA H100 80GB HBM3						
FLiTo (1 step, euler)	1	–	–	92.2 ± 0.5	92.2 ± 0.5	84.1 ± 3.1
FLiTo (4 steps, euler)	4	–	–	271.9 ± 1.1	271.9 ± 1.1	86.3 ± 4.6
MeanFlow LiTo (1 step, euler)	1	–	–	92.8 ± 0.4	92.8 ± 0.4	83.5 ± 1.7
MeanFlow LiTo (4 steps, euler)	4	–	–	342.5 ± 1.2	342.5 ± 1.2	82.6 ± 1.7
MDT-dist (1 step, euler, cfg)	2 + 2	–	50.8 ± 1.3	154.7 ± 9.5	205.5 ± 9.6	20.6 ± 5.5
MDT-dist (4 steps, euler, cfg)	7 + 7	–	126.9 ± 1.0	523.4 ± 95.4	650.2 ± 95.5	18.2 ± 4.9
SAM3D (ss distilled, 1 step, euler, cfg slat)	1 + 2	64.3 ± 0.5	66.9 ± 0.3	295.2 ± 3.8	426.4 ± 3.9	26.0 ± 2.4
SAM3D (ss distilled, 4 steps, euler, cfg slat)	4 + 8	64.4 ± 0.8	166.3 ± 0.5	787.8 ± 9.8	1018.6 ± 10.2	25.3 ± 1.4
SAM3D (both distilled, 1 step, euler)	1 + 1	64.7 ± 1.1	67.2 ± 0.3	205.3 ± 4.5	337.1 ± 4.8	27.3 ± 2.3
SAM3D (both distilled, 4 steps, euler)	4 + 4	64.7 ± 1.2	166.2 ± 0.9	509.4 ± 7.6	740.3 ± 7.7	27.0 ± 1.4
SAM3D (25 steps, euler, cfg)	75 + 50	64.7 ± 1.1	1391.2 ± 1.5	3989.9 ± 74.3	5445.8 ± 73.8	27.0 ± 1.9
Fast-SAM3D (25 steps, euler, cfg)	30 + 9.9	63.2 ± 0.8	1779.5 ± 17.1	1028.4 ± 81.4	2871.2 ± 85.5	29.6 ± 16.5
TRELLIS (25 steps, euler, cfg)	50 + 50	–	696.0 ± 2.1	3025.1 ± 214.0	3721.1 ± 214.1	16.3 ± 3.4
LiTo (25 steps, heun, cfg)	100	–	–	6390.0 ± 5.4	6390.0 ± 5.4	83.5 ± 2.0
Apple M5 Max						
FLiTo (1 step, euler)	1	–	–	2351.7 ± 5.7	2351.7 ± 5.7	884.4 ± 249.9
FLiTo (4 steps, euler)	4	–	–	7659.5 ± 85.7	7659.5 ± 85.7	877.9 ± 281.1
LiTo (25 steps, heun, cfg)	100	–	–	187246.8 ± 44.1	187246.8 ± 44.1	703.3 ± 94.4

Table 5: Number of parameters of individual methods, separated by components.

Method	Image Encoder (M)	Depth Predictor (M)	Sparse Struct Gen (M)	Latent Gen (M)	Total Gen (M)
Flito 1-step (MF init)	305.2	–	–	727.0	727.0
FLiTo 4-step (DMD init)	305.2	–	–	626.7	626.7
FLiTo 4-step (MF init)	305.2	–	–	727.0	727.0
LiTo	305.2	–	–	626.7	626.7
SAM3D	344.8	316.8	960.0	600.4	1560.4
Fast-SAM3D	344.8	316.8	960.0	600.4	1560.4
MDT-dist	304.4	–	555.5	600.4	1156.0
TRELLIS	304.4	–	555.5	600.4	1156.0

Table 6: Small-scale subtractive ablation of different losses in a 1-step setting. Arrows give the direction of improvement. "cr" is short for "conditioning reconstruction loss". "clip", "fd" and "hps" refer to our proposed semantic consistency, feature-distribution matching and human-preference supervision rewards, respectively.

	input view		novel view				alignment		latent
method	fid↓	lpips↓	fid↓	kid↓	pick↑	hps↑	siglip↑	clip↑	nll↓
full	31.71	11.83	16.38	3.25	19.60	3.336	0.878	0.896	-0.704
− kl	30.5	11.48	14.6	2.4	19.67	3.860	0.880	0.901	-0.106
− fd	33.0	11.52	19.1	4.4	19.59	3.145	0.876	0.895	-0.688
− hps	31.0	11.53	16.1	3.1	19.62	3.431	0.880	0.899	-0.728
− clip	30.2	11.44	20.3	5.2	19.64	3.047	0.869	0.893	-0.756
− cr	37.1	12.68	18.5	4.0	19.58	3.134	0.874	0.893	-0.667

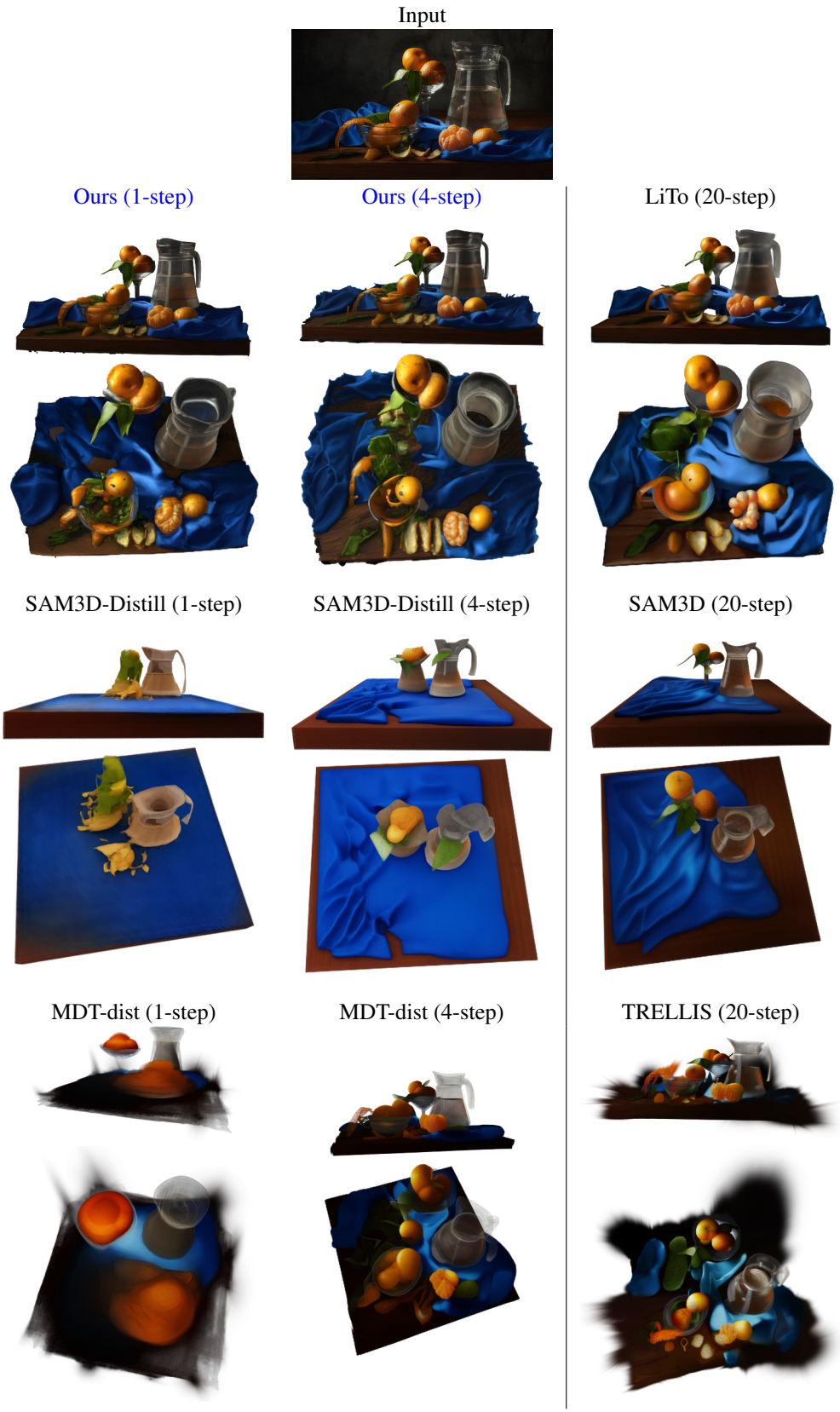

Figure 4: **Qualitative step comparison, table scene.** Our approach is clearly able to capture the complexity of this still-life scene with just a few steps, while other approaches fail to produce plausible results. Image credit: (Matvei, 2026)

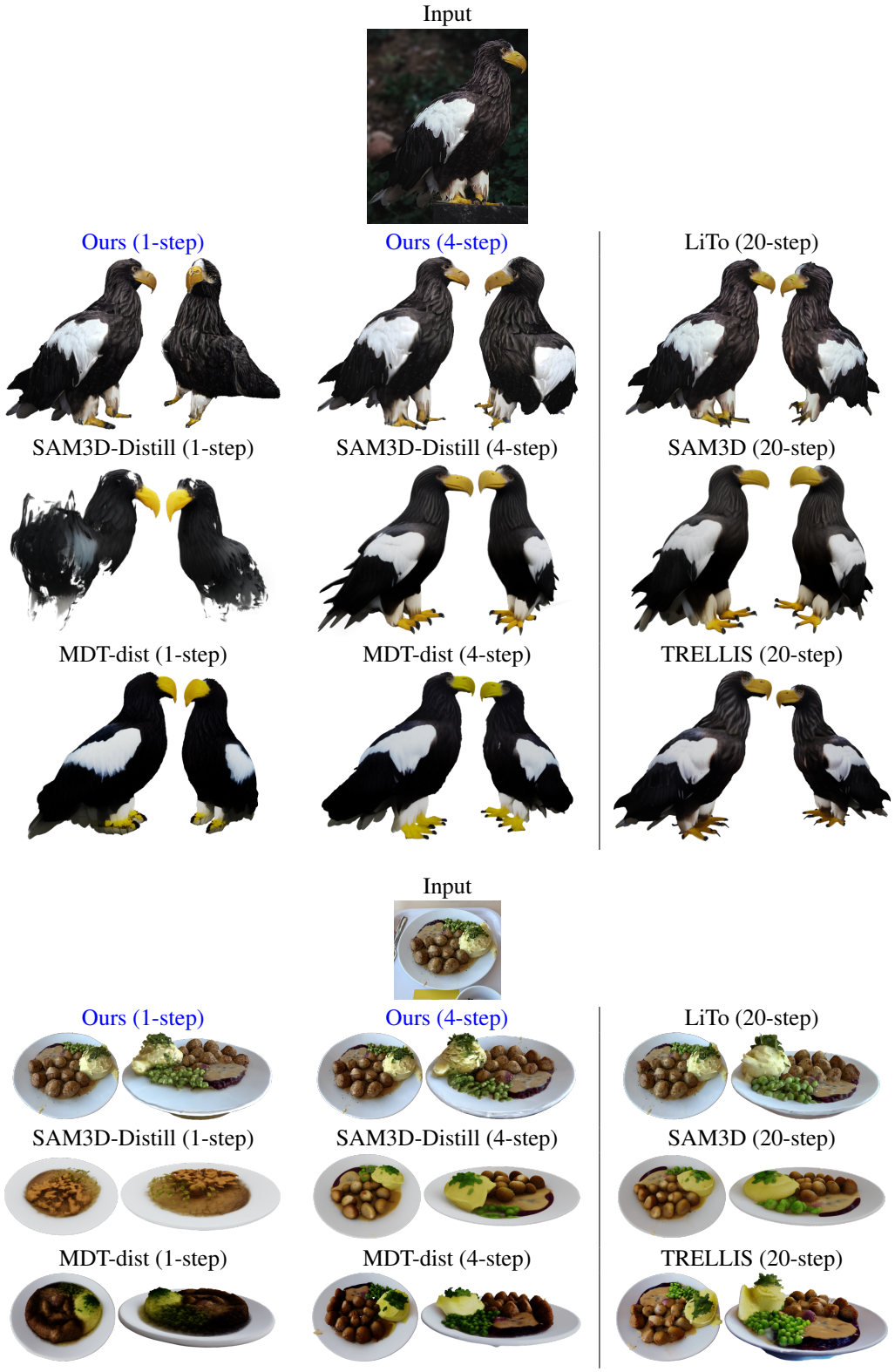

Figure 5: **Qualitative step comparison, continued.** Our approach well reproduces the details in the input image with just a few steps. In contrast, SAM3D-Distill (Chen et al., 2026) and MDT-dist (Zhou et al., 2026) fail to faithfully capture input content without significant visual distortion. Image credits: (Noiise, 2024)

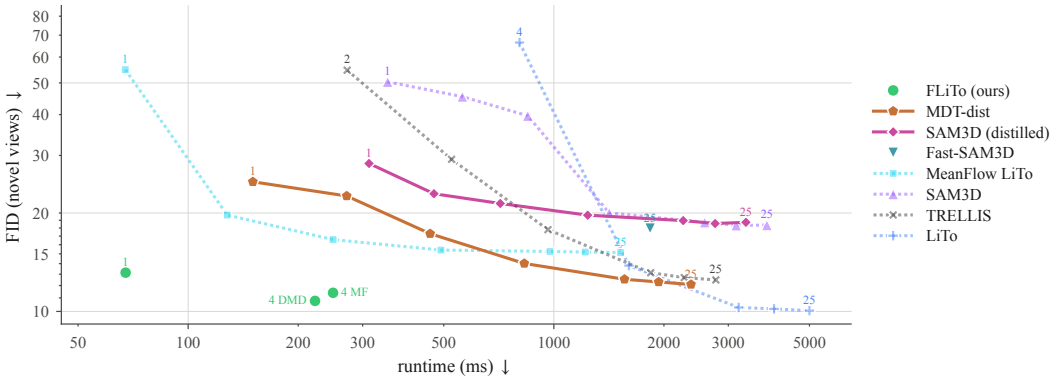

Figure 6: **FID *vs*. runtime.** The figure shows the novel-view FID and the total generation runtime on B200 for all methods we compare in the paper. For methods that support different number of ODE integration steps, we evaluate them at 1, 2, 4, 8, 16, 20, 25 steps. The number of steps are labeled by the markers. For TRELLIS and LiTo, we only show number of steps starting from 2 and 4, respectively, as they are trained to use small number of steps and generate low-quality samples in few-step settings. Both x and y axes are in log-scale.

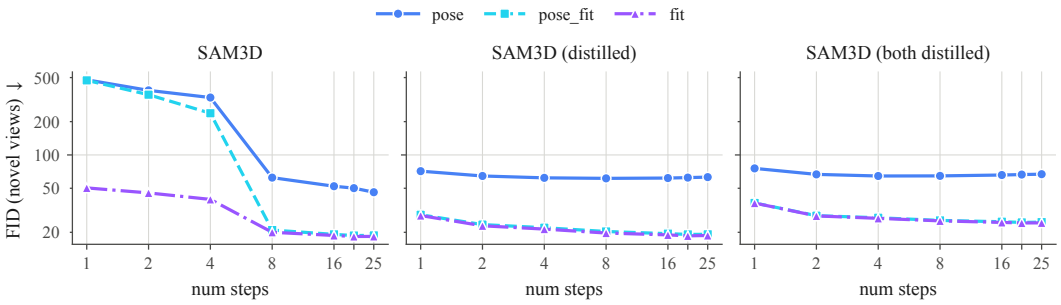

Figure 7: The figure shows the effect of pose optimization after generation. While SAM3D has a pose decoder, it might predict a different scene scale than that in the validation set, causing large FID (labeled as `pose`). To fairly compare with SAM3D and its variants, we optimize the scale and camera location, initialized by those prediced by the pose decoder (`pose_fit`) and randomly (`fit`), while keeping the predicted rotation matrix. The optimization maximizes the intersection over union between the rendered generated object's alpha map and the input alpha map with ADAM with 200 steps and a fixed learning rate (0.03). The pose optimization improves the FID values significantly. Hence, in our reported evaluations, we use the results from `fit` for SAM3D and its variants, including Fast-SAM3D. Note that we do not count the time used by pose optimization into the total runtime.

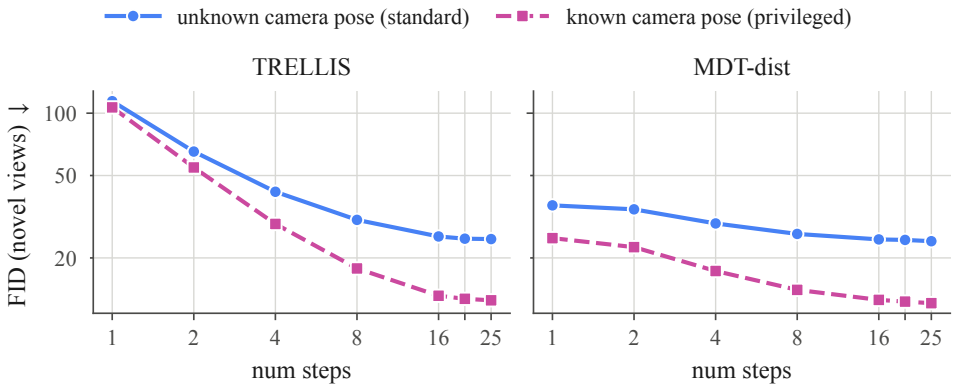

Figure 8: The figure shows the effect of known camera pose to evaluation of TRELLIS and MDT-dist. As TRELLIS and MDT-dist generates objects in the unknown object coordinate (*i.e.*, as artists would have created them in Blender), the generated objects do not necessary align with the input image. When evaluating TRELLIS and MDT-dist, we additionally transform the evaluation cameras to the object coordinate using the input camera poses in the evaluation set. Note that this is a privileged information as typically we do not know the relation transformation between the input camera and the object. As can be seen, this operation improves novel-view FID significantly. Hence in the paper, we report TRELLIS and MDT-dist results from `known camera pose (privileged)`.