VARIATIONAL TEST-TIME OPTIMIZATION FOR DIFFUSION SYNCHRONIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Collaborative generation, which coordinates multiple diffusion trajectories to extend the capabilities of pretrained priors, has emerged as a powerful paradigm for extending the applicability of diffusion models. Among existing approaches, diffusion synchronization provides a scenario-agnostic solution by introducing general guidance mechanisms. However, existing approaches rely heavily on heuristics and still require task-specific tailoring, which limits their generalizability and performance. In this work, we mathematically derive a synchronization framework based on optimal control, providing a principled explanation of diffusion synchronization. During sampling, we optimize control variables to guide multiple trajectories toward coherent solutions while remaining close to the underlying diffusion prior. Our method operates entirely at test time without additional training, thereby enabling broad applicability across diverse scenarios. Experimental results show that our method consistently improves generation quality. For wide image generation, our method reduces color and style inconsistency by 42.7% and 26.7% relative to the strongest baseline. It also achieves the best distributional alignment among the baselines for 3D mesh texturing. These results establish a principled framework for leveraging pretrained generative models for collaborative generation.

1 INTRODUCTION

Pretrained diffusion (Sohl-Dickstein et al., 2015; Ho et al., 2020) and flow matching models (Lipman et al., 2023; Liu et al., 2023) provide powerful generative priors for individual images or views (Rombach et al., 2022; Labs, 2024; Xie et al., 2025; Wu et al., 2025; Zhao et al., 2026). However, many generation problems require multiple outputs to jointly satisfy constraints that were never modeled during training. For instance, a wide panorama image can be assembled from overlapping patches that are each plausible under a pretrained image model, but only if the patches agree where they overlap; likewise, the views used to texture a 3D mesh must agree on shared surfaces. Training a dedicated joint model for every such task is undesirable: suitable training data is often unavailable, and for high-dimensional outputs, training is prohibitively expensive.

Collaborative generation (Lee et al., 2025) instead coordinates pretrained priors at test time to produce coherent samples in a larger output space: multiple diffusion trajectories each generate a patch, which must remain plausible under its prior while agreeing with the others. Existing methods address a range of collaborative generation tasks (Zhang et al., 2024a; Geng et al., 2024; Xu et al., 2025; Cheng et al., 2025; Park et al., 2026; Hayakawa et al., 2026), and diffusion synchronization seeks test-time guidance mechanisms shared across tasks (Bar-Tal et al., 2023; Kim et al., 2024; Lee et al., 2025; Yeo et al., 2025; Lee et al., 2026). However, most of these approaches rely largely on heuristic and empirical design (Kim et al., 2024; Lee et al., 2025; 2026). We ask the following question: *Can we derive a general framework that specifies what synchronized trajectories should optimize, while leaving interactions between trajectories as a modeling choice?*

Our key idea is to treat synchronization as posterior inference. We combine pretrained diffusion priors with a reward that scores cross-patch consistency. The goal is then to learn a posterior over the joint output. Consequently, we resort to variational inference where we approximate the joint posterior over patches via a controlled diffusion process over all trajectories at once. The resulting ELBO explicitly balances expected consistency against KL divergence from each prior and is optimized at test time through *variational controls* that steer every denoising step without modifying the pretrained

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.

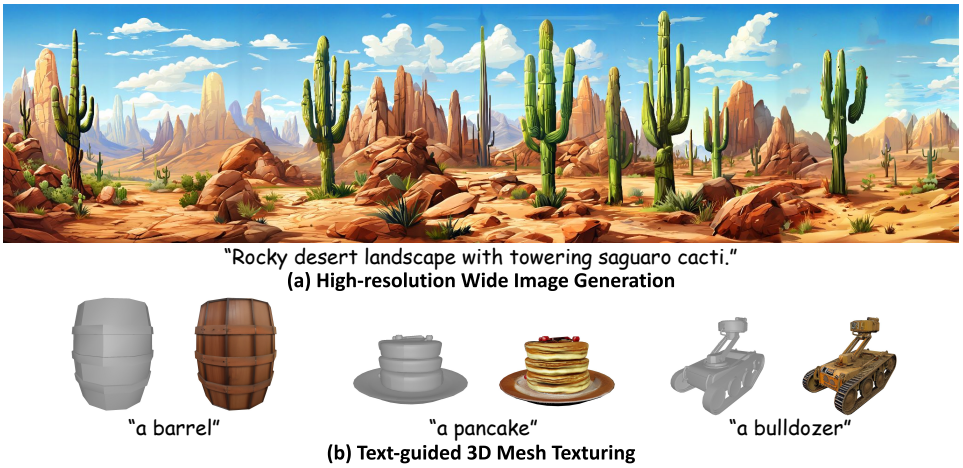


Figure 1: **Results of SyncVC on collaborative generation tasks,** where multiple diffusion trajectories, one per patch or view, are synchronized into a single consistent output. We show (a) an 8192×2048 wide image generated with SANA (Xie et al., 2025) and (b) textured 3D meshes.

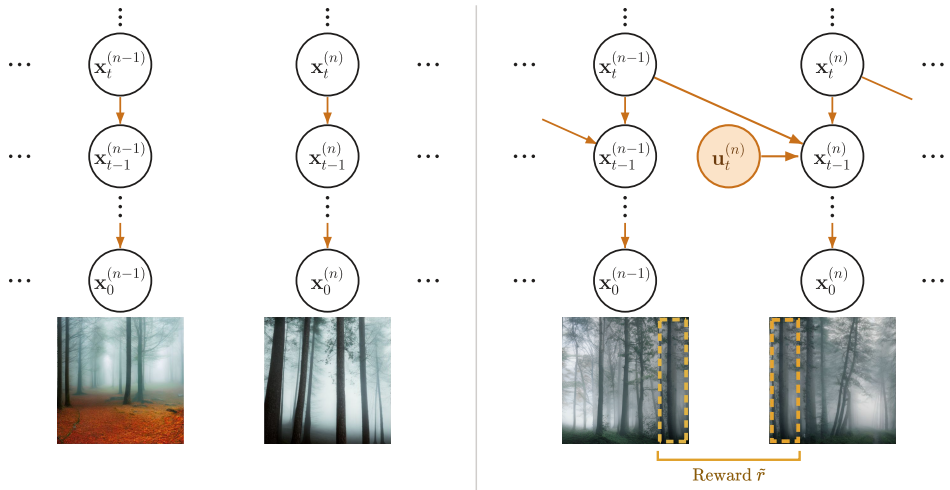
model. Crucially, this objective fixes *what* synchronized trajectories should optimize but not *how* they depend on one another: designing a synchronization method reduces to choosing a factorization of the joint variational process and a variational parameterization.

However, solving this variational inference problem is nontrivial since the reward couples the terminal outputs of several trajectories, requiring a joint variational process that represents their dependencies while remaining tractable at every denoising step. Consequently, we impose simplifying assumptions under which the controls can be optimized greedily at test time, one trajectory at a time. We denote this framework as **Synchronized Diffusion with Variational Controls (SyncVC)**. Because the pretrained model is never modified, SyncVC applies directly to diverse pretrained models (Rombach et al., 2022; Zhang et al., 2023; DeepFloyd Lab, 2023; Xie et al., 2025) without additional training. Task-specific constraints can be incorporated through reward terms. Experiments on wide image generation and text-guided 3D mesh texturing demonstrate improved color and style consistency and strong distributional alignment. Further analyses validate our factorization choices and demonstrate a tunable runtime–quality trade-off and flexibility in reward design. We also demonstrate high-resolution wide image generation using the pretrained SANA model (Xie et al., 2025), generating coherent 8192×2048 wide images (Figure 1). We summarize our contributions as follows:

- **A unified variational formulation for diffusion synchronization.** We formulate collaborative generation as inference over coupled trajectories and derive an ELBO balancing cross-patch consistency against fidelity to frozen diffusion priors. It is factorization-agnostic, decoupling the synchronization objective from how trajectories are coupled.
- **A tractable multi-trajectory control algorithm.** We instantiate this framework as SyncVC, using a chain-structured variational process that couples neighboring trajectories through test-time optimized controls, yielding practical synchronization without additional training.
- **Strong empirical validation across tasks.** SyncVC improves consistency and distributional quality across wide image generation, 3D mesh texturing, and optical illusion generation; it also supports small overlaps, high-resolution generation, and additional reward terms.

2 METHOD

We present our method through wide image generation, where each diffusion trajectory generates an overlapping image patch. We formulate their joint generation as posterior inference, combining pretrained diffusion priors with a consistency reward, and approximate this posterior with a controlled diffusion process as shown in Figure 2. At each denoising step, variational controls increase the expected reward while limiting KL divergence from the prior. The central modeling choice is how to factorize the joint transitions across trajectories. Sec. 3.2 and Appendix E.3 additionally cover 3D mesh texturing and optical illusion generation, respectively. Sec. 2.1 formalizes the task, Sec. 2.2 derives the variational objective, and Sec. 2.3 presents the proposed method, SyncVC.



(a) Prior (unsynchronized) distribution

(b) Variational (synchronized) distribution (SyncVC)

Figure 2: **Graphical model of SyncVC for wide image generation contrasting independent and synchronized trajectories.** (a) Without synchronization, each trajectory samples its patch independently from the pretrained prior, and neighboring patches differ in appearance. (b) In SyncVC, each transition is conditioned on the predecessor's noisy state and shifted by a control $\mathbf{u}_t^{(n)}$; the reward $\tilde{r}$ compares neighboring patches on their overlap (dashed boxes).

2.1 Problem formulation

We assume a distribution over high-dimensional data $\boldsymbol{X}$ for which learning a generative model as a whole is out of reach. Instead, we have a pretrained diffusion model over parts of $\boldsymbol{X}$, which we call *patches*; it maps Gaussian noise $\mathbf{x}_T^{(i)}$ to a patch $\mathbf{x}_0^{(i)} \in \mathbb{R}^d$. We represent $\boldsymbol{X} = \{\mathbf{x}_0^{(1)}, \dots, \mathbf{x}_0^{(N)}\}$ by N patches whose neighbors overlap and must agree on their shared content, and ask how to generate them jointly so that $\mathbf{X}$ is coherent as a whole. For instance, the patches are partially overlapping regions of a wide image. We encode this agreement in a reward, together with any additional constraints given by an observation $\boldsymbol{y} \in \mathbb{R}^m$. Formally, we seek to sample from the posterior:

$$p(\boldsymbol{X} \mid \boldsymbol{y}) \propto p(\boldsymbol{X})\, p(\boldsymbol{y} \mid \boldsymbol{X}), \quad \text{where} \quad p(\boldsymbol{y} \mid \boldsymbol{X}) \propto \exp\left(r(\boldsymbol{y}, \boldsymbol{X})/\lambda\right), \tag{1}$$

Here, $p(\boldsymbol{X})$ is a prior over the set and the likelihood is defined through an energy-based, task-specific reward $r : \mathbb{R}^m \times \mathbb{R}^{Nd} \to \mathbb{R}$ with temperature $\lambda > 0$.

Choice of reward. We consider rewards that are **known**, **differentiable**, and can be evaluated in **closed form**. Given an ordering of the patches, we define the overall reward as a sum of contributions associated with individual patches:

$$r(\boldsymbol{y}, \boldsymbol{X}) := \sum_{n=2}^{N} \tilde{r}_n\left(\boldsymbol{y}, \mathbf{x}_0^{(1:n)}\right), \qquad \mathbf{x}_0^{(1:n)} := \left(\mathbf{x}_0^{(1)}, \dots, \mathbf{x}_0^{(n)}\right). \tag{2}$$

The notation $\mathbf{x}_0^{(1:n)}$ allows each sub-reward to use the current patch and any relevant subset of preceding patches. For wide image generation, $\tilde{r}_n$ compares patches $n-1$ and n over their overlap; for stylized generation shown in Figure 7, a style reference is additionally supplied through $\boldsymbol{y}$. The general form of the reward is given in Eq. 9, and the task-specific definitions in Appendix E. We deliberately keep these designs simple and intuitive rather than relying on heavily engineered components; as shown in Sec. 3, these choices already outperform prior baselines, and we leave more sophisticated parameterizations to future work.

Choice of prior. We take the prior to factorize over patches, $p(\boldsymbol{X}) = \prod_{n=1}^{N} p(\mathbf{x}_0^{(n)})$, and model each factor with the same pretrained diffusion model using T denoising steps. Its joint distribution over the denoising trajectory $\mathbf{x}_{0:T}^{(n)}$ is

$$p(\mathbf{x}_{0:T}^{(n)}) = p(\mathbf{x}_T^{(n)}) \prod_{t=1}^{T} p_{\boldsymbol{\phi}}(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}), \tag{3}$$

where $p_{\boldsymbol{\phi}}$ denotes the Gaussian denoising kernel with mean $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}_t^{(n)}, t)$ and covariance $\sigma_t^2\mathbf{I}$, σ_t is a sampler-dependent step size, t is the diffusion timestep, and $p(\mathbf{x}_T^{(n)})$ is a standard Gaussian. We keep the pretrained model fixed throughout, in the regime of training-free test-time guidance, and drop any additional conditioning of the model (*e.g.* a text prompt (Rombach et al., 2022)) from the notation. The factorized prior is chosen for simplicity and compatibility with off-the-shelf pretrained models, which provide a prior over a single patch. Sec. 3 demonstrates that this factorization works well empirically. If a joint prior over several patches is available, it can be substituted directly.

2.2 Variational inference over the joint trajectory

The target posterior $p(\boldsymbol{X} \mid \boldsymbol{y})$ in Eq. 1 is generally intractable, hence we approximate it with a variational distribution and optimize a lower bound on the evidence. We first collect the noisy states of all N patches at diffusion step t into a joint state $\boldsymbol{X}_t := (\mathbf{x}_t^{(1)}, \dots, \mathbf{x}_t^{(N)})$, so that $\boldsymbol{X} = \boldsymbol{X}_0$, and introduce variational parameters $\boldsymbol{U}_t := (\mathbf{u}_t^{(1)}, \dots, \mathbf{u}_t^{(N)})$, with $\boldsymbol{U}_{1:T} := (\boldsymbol{U}_1, \dots, \boldsymbol{U}_T)$. The variational distribution is a diffusion process over the joint state,

$$q(\boldsymbol{X}_{0:T} \mid \boldsymbol{y}, \boldsymbol{U}_{1:T}) = q(\boldsymbol{X}_T) \prod_{t=1}^{T} q(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{y})\,. \tag{4}$$

Combining Eq. 4 with Eq. 3, the ELBO is given as follows (See Appendix A for the full derivation):

$$\mathcal{L}(\boldsymbol{y}) := \mathbb{E}_q[r(\boldsymbol{y}, \boldsymbol{X})] - \lambda \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{X}_t \mid \boldsymbol{y}, \boldsymbol{U}_{1:T})}[\, D_{\mathrm{KL}}(q(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{y}) \,\|\, p(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t))]\,, \tag{5}$$

where $p(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t) = \prod_{n=1}^{N} p_{\boldsymbol{\phi}}(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)})$ is the joint reverse transition of the pretrained prior. Eq. 5 captures the two competing goals of collaborative generation. The first term encourages sequences that maximize the reward, while the second pulls the variational distribution toward the prior diffusion model at each denoising step. The scalar hyperparameter λ controls the strength of this regularization. For any observation $\boldsymbol{y}$, we optimize Eq. 5 to infer the variational parameters $\boldsymbol{U}_{1:T}$ directly at test time. The objective is agnostic to how $q(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{y})$ is factorized over patches; that factorization is the modeling choice we turn to next.

2.3 SyncVC

Our central modeling choice is to factorize each joint transition along the sequence, so that every patch is conditioned on the current noisy state of its predecessor and shifted by its variational parameter:

$$q(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{y}) = q\Big(\mathbf{x}_{t-1}^{(1)} \mid \mathbf{x}_t^{(1)}\Big) \prod_{n=2}^{N} q\Big(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}, \mathbf{x}_t^{(n-1)}, \mathbf{u}_t^{(n)}, \boldsymbol{y}\Big)\,. \tag{6}$$

The first patch is not guided by default, *i.e.*, $\mathbf{u}_t^{(1)} = \mathbf{0}$ and its transition follows the pretrained model; the remaining $N-1$ variational parameters are optimized. By conditioning the transition of $\mathbf{x}_t^{(n)}$ on $\mathbf{x}_t^{(n-1)}$ of the previous trajectory, the generation of $\mathbf{x}_0^{(n)}$ is synchronized with $\mathbf{x}_0^{(n-1)}$ at every diffusion step, and the variational parameter $\mathbf{u}_t^{(n)}$ steers that transition toward the reward r, thereby enabling collaborative generation as shown in Figure 4. Following Lee et al. (2025), we model each transition as a conditional Gaussian with the same covariance $\sigma_t^2\mathbf{I}$ as the prior kernel and mean

$$\bar{\boldsymbol{\mu}}_t^{(n)} = \underbrace{\boldsymbol{\mu}_{\boldsymbol{\phi}}\Big(\bar{\mathbf{x}}_t^{(n)}, t\Big)}_{\text{Terminal Step}} - \frac{\gamma}{2}\sigma_t^2 \underbrace{\nabla_{\mathbf{x}_t^{(n)}} \Big\| f\left(\mathbf{x}_t^{(n-1)}, \boldsymbol{y}\right) - \bar{\mathbf{x}}_t^{(n)} \Big\|_2^2}_{\text{Regularizer}}, \tag{7}$$

where $\bar{\mathbf{x}}_t^{(n)} = \mathbf{x}_t^{(n)} + \beta \mathbf{u}_t^{(n)}$ and $\beta > 0$ is a hyperparameter that defines their strength. Here, σ_t is the sampler-dependent step size of Eq. 3. The task-dependent operator f maps previously generated outputs into the current coordinate frame, using a patch shift for wide images or re-rendering previously textured surfaces for meshes. The first term steers the denoising trajectory toward higher reward through $\mathbf{u}_t^{(n)}$, while the second acts as a regularizer to reduce the gap between

adjacent denoising chains. We empirically find that this sequential factorization works well. Note that our framework naturally accommodates another factorization, and we show an ablation comparing alternative factorizations in Sec. 3.3.

Since the variational parameters steer the trajectory in this way, they play the role of controls in stochastic optimal control (Pandey et al., 2025) and we refer to them as *variational controls*. We refer to this overall framework as **Synchronized Diffusion with Variational Controls (SyncVC)** and summarize it in Fig. 2.

Practical objective. The sub-rewards $\tilde{r}_n$ in Eq. 2 are defined on the clean sequence predictions $\mathbf{x}_0^{(n)}$. Direct optimization of the variational objective in Eq. 5 would therefore require rolling out the full reverse diffusion chain at every timestep to evaluate $r(\boldsymbol{y}, \boldsymbol{X})$, which is computationally prohibitive. Instead, we approximate the clean sample at the current timestep using Tweedie's estimate. Therefore, the loss function simplifies to the following objective:

$$\mathbf{u}_t^\star = \arg\min_{\mathbf{u}_t} \sum_{n=2}^{N} \mathbb{E}_{\mathbf{x}_t^{(1:n)}} \left[-\tilde{r}_n\left(\boldsymbol{y}, \hat{\mathbf{x}}_{0|t}^{(1:n)}\right) + \frac{\lambda}{2\sigma_t^2} \left\| \bar{\boldsymbol{\mu}}_t^{(n)} - \boldsymbol{\mu}_{\boldsymbol{\phi}}\left(\mathbf{x}_t^{(n)}, t\right) \right\|_2^2 \right], \tag{8}$$

where Tweedie's estimate is given by $\hat{\mathbf{x}}_{0|t}^{(n)} = \mathbb{E}\left[\mathbf{x}_0 \mid \bar{\mathbf{x}}_t^{(n)}\right]$. In all experiments, we use the DDIM sampler (Song et al., 2021a), under which Eq. 8 admits a closed form; the resulting objective is given in Appendix B. Algorithm 1 in the appendix summarizes the overall procedure.

3 Experiments

We evaluate our method on wide image generation (Sec. 3.1) and text-guided 3D mesh texturing (Sec. 3.2), with optical illusion generation presented in Appendix E.3. The goal of diffusion synchronization is to generate a consistent appearance across patches or views while maintaining strong distributional alignment with the diffusion prior. Our method achieves superior style and color consistency (Lee et al., 2023b) in wide image generation (Table 1, Figure 3) and produces high-quality textures with the best FID (Heusel et al., 2017) and KID (Bińkowski et al., 2018) scores among the mesh texturing baselines (Table 2, Figure 5). We further examine practical formulation choices, runtime–quality trade-offs, and flexibility in reward design (Sec. 3.3).

Datasets. For wide image generation, we use 15 text prompts from prior work (Bar-Tal et al., 2023; Lee et al., 2025; Kim et al., 2024; Yeo et al., 2025) and generate 50 images per prompt, yielding 750 wide images per method. For mesh texturing, we evaluate 350 mesh–prompt pairs sampled from the Objaverse (Deitke et al., 2022) dataset.

Evaluation protocol. For wide images, we measure perceptual and style consistency across patches using Intra-LPIPS and Intra-Style-Loss (Lee et al., 2023b), and color consistency using the χ^2 distance and intersection between HSV histograms. We assess distributional alignment using KID for wide images and FID and KID for rendered mesh views. For meshes, we additionally measure the CLIP image–text alignment score (CLIP-S) (Radford et al., 2021) to quantify alignment with the text prompt. Additional details are provided in Appendix E. In all tables, we **bold the best** and underline the second-best results.

Baselines. We compare the proposed method against the diffusion synchronization methods SyncTweedies (Kim et al., 2024), SyncSDE (Lee et al., 2025), and StochSync (Yeo et al., 2025). We additionally include MultiDiffusion (Bar-Tal et al., 2023) for wide image generation and TexPainter (Zhang et al., 2024b) and TEXTure (Richardson et al., 2023) for mesh texturing.

Reward. We instantiate the sub-rewards in Eq. 2 using a masked squared-error form:

$$\tilde{r}_n\left(\boldsymbol{y}, \mathbf{x}_0^{(1:n)}\right) = -\frac{\gamma}{2} \left\| \mathbf{M}^{(n)} \odot \left(f\left(\mathbf{x}_0^{(1:n-1)}, \boldsymbol{y}, n\right) - \mathbf{x}_0^{(n)} \right) \right\|_2^2, \tag{9}$$

Here, f is a task-specific operation that constructs a comparison variable from previous patches in the coordinate system of the current patch. Its definition differs across tasks and is specified in Appendix E. The binary mask $\mathbf{M}^{(n)}$ selects the comparison region, and γ is a scalar hyperparameter.

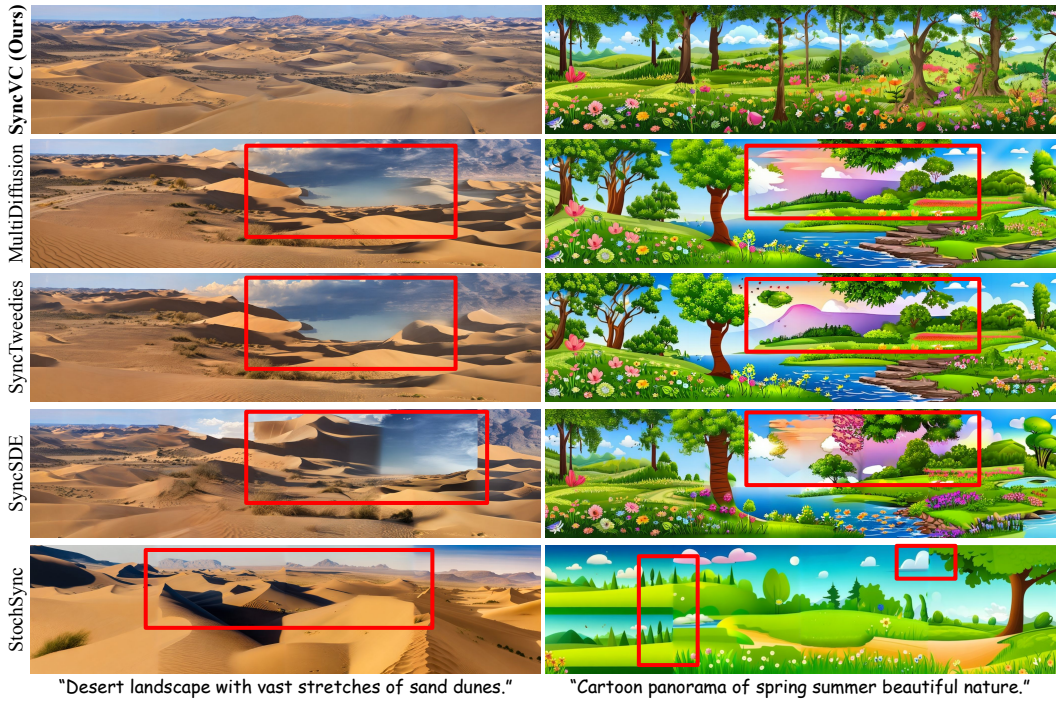


Figure 3: **Qualitative comparison on wide image generation**, illustrating consistency of appearance across patches. Each row shows one method for two prompts; red boxes mark regions where the sky, terrain, or color palette changes abruptly within a baseline's image.

Table 1: **Quantitative evaluation of wide image generation.** Intra-LPIPS and Intra-Style-Loss (Gatys et al., 2016) measure perceptual and style consistency across patches, χ^2-histogram distance and histogram intersection measure color consistency, and KID (Bińkowski et al., 2018) measures distributional alignment. KID is scaled by 10^3.

Metric	**SyncVC (Ours)**	MultiDiffusion	SyncTweedies	SyncSDE	StochSync
χ^2-Histogram dist. ↓	**0.75**	<u>1.21</u>	1.35	1.31	1.35
Intra-Style-Loss ↓	**44.34**	<u>58.46</u>	78.05	63.98	67.56
Histogram intersect. ↑	**0.67**	<u>0.55</u>	0.52	0.53	0.52
KID ↓	**52.07**	58.26	60.81	<u>57.08</u>	100.47
Intra-LPIPS ↓	**0.59**	0.64	<u>0.62</u>	0.65	<u>0.62</u>

3.1 Wide Image Generation

Setup. The goal is to synthesize a horizontally elongated image from a text prompt by combining partially overlapping patches, each generated by a separate diffusion trajectory. The patches should agree where they overlap and maintain a coherent appearance across the full image. We generate 2048×512 wide images using pretrained Stable Diffusion (Rombach et al., 2022), with each patch of size 512^2. Our method samples five patches with an overlap of 128 pixels and sequentially composites them into the final wide image.

Results. Table 1 shows that SyncVC outperforms baselines on all metrics, with the largest margins on color and style consistency: relative to the strongest baseline, it reduces the χ^2-histogram distance by 42.7% and Intra-Style-Loss by 26.7% relative to the strongest baseline for each metric. Figure 3 confirms this qualitatively. MultiDiffusion and SyncTweedies drift semantically in the center of the image, introducing a lake into the desert scene and a sunset sky to a daytime scene, while SyncSDE and StochSync produce abrupt seams. In contrast, SyncVC maintains consistent lighting, color, and style across the image. In Figure 1a, we further demonstrate this consistency at high resolution, generating an 8192×2048 image with the pretrained SANA model (Xie et al., 2025).

Robustness to small overlaps. We further evaluate a more challenging setting with an overlap of only 16 pixels between adjacent patches. This setting is also of practical interest, since smaller overlaps require fewer trajectories to cover a given resolution, reducing computational cost. We compare against MultiDiffusion (Bar-Tal et al., 2023). As shown in Figure 4, with little overlap to average over, MultiDiffusion's patches are only weakly coupled and diverge in appearance, producing visible seams with abrupt shifts in color and lighting, such as the fog tone in the forest and the sky hue across the skyline. In contrast, SyncVC produces a coherent image with consistent style across the full width. Additional baseline comparisons are provided in Figure 9 in the Appendix.



Figure 4: **Wide image generation with a 16-pixel overlap** (3.125% of the patch width), testing whether patches remain consistent when they share little content. SyncVC (top) and MultiDiffusion (Bar-Tal et al., 2023) (bottom) for two prompts.

3.2 Text-guided 3D mesh texturing

Setup. We aim to generate a high-quality texture for an input mesh from a text prompt, using separate diffusion trajectories to synthesize images from different viewpoints. These views must agree on the appearance of shared surfaces so that they collectively define a coherent texture. Our method generates eight 768^2 views using pretrained depth-conditioned ControlNet (Zhang et al., 2023), with uniformly sampled azimuth angles at a fixed elevation. These partially overlapping views are used to synthesize a 1024^2 RGB texture map.

Results. Table 2 reports quantitative results. SyncVC achieves the lowest KID and FID among all methods, while remaining comparable to the baselines in text consistency as measured by CLIP-S. Qualitatively (Figure 5), while all methods produce reasonable outputs, alternative methods encounter issues with texture quality. For example, the baselines fail to render the flashlight's lens in detail, whereas SyncVC reproduces it clearly. This is also evident in the bulldozer's tracks, which are clearly visible in SyncVC's output but with artifacts in SyncSDE, TexPainter, and TEXTure.

Table 2: **Quantitative evaluation of text-guided 3D mesh texturing.** FID (Heusel et al., 2017) and KID (Bińkowski et al., 2018) measure distributional alignment of rendered views, and CLIP-S (Radford et al., 2021) measures image–text alignment. KID is scaled by 10^3.

Metric	**SyncVC (Ours)**	TexPainter	TEXTure	SyncTweedies	SyncSDE	StochSync
KID ↓	**75.17**	110.84	96.28	82.05	85.12	<u>80.11</u>
FID ↓	**161.63**	192.08	188.26	163.08	172.74	<u>162.71</u>
CLIP-S ↑	<u>28.96</u>	28.30	28.81	**29.21**	28.83	28.91

3.3 Additional analysis

Ablation on model factorization and control optimization. We study the effect of the variational factorization and the control optimization scheme. First, we consider a bidirectional factorization of the variational distribution: at a diffusion timestep t, each reverse transition is conditioned not only on the preceding trajectory, as in Eq. 6, but also on the succeeding one. We derive the corresponding ELBO in Appendix D. Table 3 shows that conditioning only on the predecessor, as in our autoregressive factorization, achieves better overall performance than the bidirectional factorization on both tasks, and we therefore adopt it. Next, under the autoregressive factorization, we compare our greedy control optimization with two alternatives: timestep-wise joint optimization

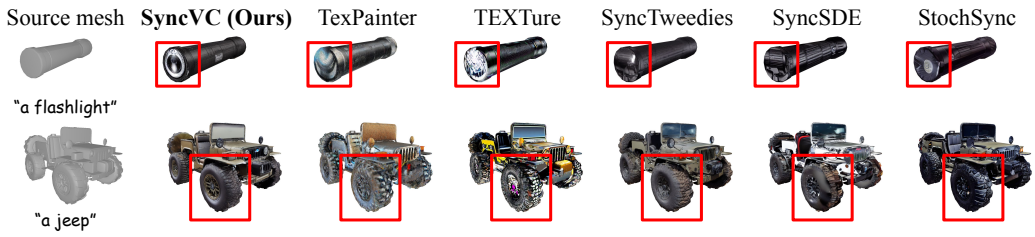


Figure 5: [Best viewed when magnified.] **Qualitative comparison on text-guided 3D mesh texturing**, focusing on fine texture detail. Columns show the source mesh and the texture produced by each method; red boxes highlight the flashlight lens (Row 1) and the jeep's tires (Row 2).

Table 3: **Quantitative ablation study on model factorization and control optimization.** Our method uses autoregressive factorization and greedy control optimization. Bidirectional changes the factorization of the graphical model, while Joint opt. uses timestep-wise joint control optimization under the same autoregressive factorization. KID is scaled by 10^3.

(a) Wide image generation

Metric	**SyncVC (Ours)**	Bidirectional	Joint opt.
χ^2-Histogram dist. ↓	**0.75**	0.88	1.16
Intra-Style-Loss ↓	**44.34**	58.56	70.49
Histogram intersect. ↑	**0.67**	0.63	0.57
KID ↓	**52.07**	52.17	112.01
Intra-LPIPS ↓	**0.59**	0.62	**0.59**

(b) Text-guided 3D mesh texturing

Metric	**SyncVC (Ours)**	Bidirectional
FID ↓	**161.63**	164.45
KID ↓	**75.17**	81.11
CLIP-S ↑	**28.96**	**28.96**

and full-rollout joint optimization, the latter presented in Appendix E.4.1. The greedy scheme optimizes controls sequentially across trajectories, allowing each trajectory to align with an already-synchronized predecessor, whereas timestep-wise joint optimization updates the controls of all trajectories simultaneously at each diffusion timestep. As shown in Table 3a, greedy optimization outperforms timestep-wise joint optimization, most notably in KID (52.07 vs. 112.01).

Runtime–quality trade-off. Our method offers a tunable trade-off between runtime and color consistency through the number of optimization steps K and denoising steps T. All runtimes are measured on a single NVIDIA A6000 GPU. Figure 6 (connected lines) shows that, with $T = 50$, increasing K reduces χ^2-Histogram distance at the cost of longer runtime. Even at $K = 1$, our method achieves better color consistency than MultiDiffusion (Bar-Tal et al., 2023) at a comparable runtime. With $T = 18$ and $K = 2$, our method generates images approximately $1.7\times$ faster than MultiDiffusion while retaining better color consistency. Results across all evaluation metrics and additional runtime comparisons with baselines are provided in Appendix E.4.2.

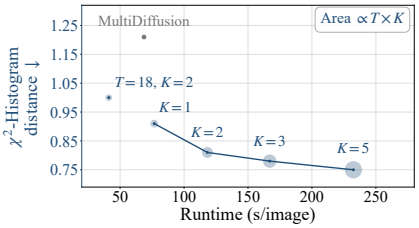


Figure 6: **Runtime–quality trade-off in wide image generation.** χ^2-histogram distance versus runtime for SyncVC and MultiDiffusion. More optimization steps improve color consistency at higher runtime. Connected points use $T = 50$, and circle area is proportional to $T \times K$.

Flexibility in reward design. Additional reward terms let us impose style or semantic constraints while retaining the same synchronization framework. In particular, multiple reward terms can be combined to guide a single generation process, allowing different conditioning signals to be incorporated within the same formulation. We demonstrate this flexibility through stylized wide image generation, subtracting a style transfer loss (Gatys et al., 2016) between a reference image and the generated patch $\mathbf{x}_0^{(n)}$ from Eq. 9. Figure 7 shows that our method incorporates the overall style of the reference while preserving the semantic content specified by the text prompt. This example illustrates how a reward term can introduce an additional conditioning signal into the generation. We additionally evaluate a CLIP-augmented reward for text–image alignment; its formulation, the objective for guiding all patches, and quantitative results are provided in Appendix E.4.

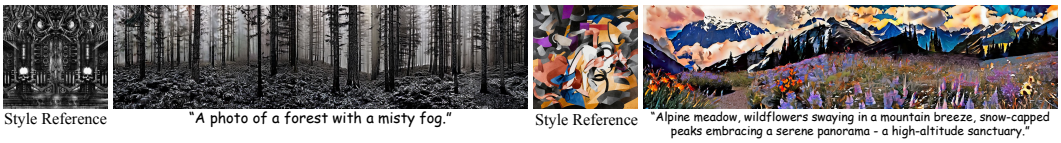


Figure 7: **Stylized wide image generation with an additional style reward**, showing that further constraints can be added through the reward. Each wide image is generated from the prompt below it and the style reference to its left.

4 Related work

Task-specific methods for collaborative generation. Task-specific approaches coordinate diffusion trajectories through specialized mechanisms. SyncDiffusion (Lee et al., 2023b) ensures consistent style across the wide image by minimizing LPIPS distance (Zhang et al., 2018) between patches. Text-guided 3D mesh texturing also has been addressed using diffusion-based approaches (Zhang et al., 2024b; Youwang et al., 2024; Richardson et al., 2023; Liu et al., 2024; Zeng et al., 2024; Chen et al., 2023; Dongyu et al., 2025). For example, TexPainter (Zhang et al., 2024b) enforces multi-view consistency via color-space fusion at each diffusion step. Meanwhile, TEXTure (Richardson et al., 2023) employs a diffusion process tailored to texturing with a trimap representation and iteratively updates texture maps from different viewpoints. However, these methods are tailored to specific tasks and lack generalizability across different collaboration scenarios. In contrast, our method does not rely on task-specific designs but provides a general framework with strong performance.

Diffusion synchronization. Diffusion synchronization methods (Bar-Tal et al., 2023; Kim et al., 2024; Lee et al., 2025; Yeo et al., 2025; Lee et al., 2026) aim to provide general mechanisms across diverse collaborative generation scenarios. MultiDiffusion (Bar-Tal et al., 2023) aligns trajectories by averaging diffusion latents at each timestep, while SyncTweedies (Kim et al., 2024) identifies averaged Tweedie estimates as the optimal strategy. However, their reliance on heuristics limits their scalability and generalization beyond the evaluated settings. On the other hand, SyncSDE (Lee et al., 2025) proposes autoregressive trajectory sampling, where the conditional score of the current trajectory given the previous trajectory is approximated with a Gaussian. This strong assumption limits its applicability. StochSync (Yeo et al., 2025) interprets synchronization via score distillation sampling (Poole et al., 2023), but still depends on heuristic engineering techniques such as non-overlapping view sampling. In contrast, our method introduces control variables and optimizes them at test time using a loss function derived from variational inference.

Diffusion with optimal controls. Optimal control (Kappen, 2008) has been used to design guidance in diffusion models (Huang et al., 2024; Pandey et al., 2025; Li & Pereira, 2024; Berner et al., 2024; Azangulov et al., 2026; Chen et al., 2024; Rout et al., 2025). Stochastic Control Guidance (Huang et al., 2024) uses path-integral control for plug-and-play diffusion guidance with non-differentiable rewards. For general inverse problems, Diffusion Trajectory Matching (Pandey et al., 2025) formulates guidance for a single trajectory as a variational control problem, where control variables are optimized to satisfy terminal constraints while regularizing deviations from the pretrained prior. Azangulov et al. (2026) formulate adaptive guidance scheduling as a stochastic optimal control problem, dynamically selecting the guidance scale. Building on this control perspective, our work studies collaborative generation through a structured variational process over multiple diffusion trajectories.

5 Conclusion

In this work, we propose a principled framework for collaborative generation based on optimal control. Unlike prior approaches that rely heavily on heuristic designs, our method is derived from a mathematically grounded formulation, providing a novel perspective on diffusion synchronization. The proposed method demonstrates strong performance across diverse collaborative generation tasks, establishing a promising direction for extending pretrained generative priors to more versatile settings. Despite these advantages, our approach has several limitations. Because it relies on test-time optimization, it incurs additional computational cost compared to optimization-free approaches, motivating the development of more efficient guidance strategies. Furthermore, the formulation is currently restricted to differentiable rewards; extending it to incorporate non-differentiable objectives would enable broader applicability across diverse scenarios.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing, including grammar and readability. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We have identified no specific ethical concerns associated with this research.

REFERENCES

Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM computing surveys (CSUR)*, 1991.

Iskander Azangulov, Peter Potaptchik, Qinyu Li, Eddie Aamari, George Deligiannidis, and Judith Rousseau. Adaptive diffusion guidance via stochastic optimal control. *AISTATS*, 2026.

Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. Multidiffusion: Fusing diffusion paths for controlled image generation. *ICML*, 2023.

Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling. *TMLR*, 2024.

Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *ICLR*, 2018.

Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. In *ICCV*, 2023.

Tianrong Chen, Jiatao Gu, Laurent Dinh, Evangelos Theodorou, Joshua M. Susskind, and Shuangfei Zhai. Generative modeling with phase stochastic bridge. In *ICLR*, 2024.

Wei Cheng, Juncheng Mu, Xianfang Zeng, Xin Chen, Anqi Pang, Chi Zhang, Zhibin Wang, Bin Fu, Gang Yu, Ziwei Liu, et al. Mvpaint: Synchronized multi-view diffusion for painting anything 3d. In *CVPR*, 2025.

Kevin Clark, Paul Vicol, Kevin Swersky, and David Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *ICLR*, 2024.

Stability AI DeepFloyd Lab. Deepfloyd if. `https://github.com/deep-floyd/IF`, 2023.

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. *arXiv:2212.08051*, 2022.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021.

Yan Dongyu, Leyi Wu, Jiantao Lin, Luozhou Wang, Tianshuo Xu, Zhifei Chen, Zhen Yang, Lie Xu, Shunsi Zhang, and Yingcong Chen. Flexpainter: Flexible and multi-view consistent texture generation. *arXiv:2506.02620*, 2025.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

Daniel Geng, Inbum Park, and Andrew Owens. Visual anagrams: Generating multi-view optical illusions with diffusion models. In *CVPR*, 2024.

Akio Hayakawa, Yusuke Mukuta, and Tatsuya Harada. Linear fusion multidiffusion for fast training-free spherical panorama generation. *ECCV*, 2026.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NIPS*, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

Yujia Huang, Adishree Ghatare, Yuanzhe Liu, Ziniu Hu, Qinsheng Zhang, Chandramouli S Sastry, Siddharth Gururani, Sageev Oore, and Yisong Yue. Symbolic music generation with non-differentiable rule guided diffusion. *ICML*, 2024.

HJ Kappen. Stochastic optimal control theory. *ICML*, 2008.

Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer. In *ICCV*, 2021.

Jaihoon Kim, Juil Koo, Kyeongmin Yeo, and Minhyuk Sung. Synctweedies: A general generative framework based on synchronized diffusions. *NeurIPS*, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM TOG*, 2020.

Hyunjun Lee, Hyunsoo Lee, and Sookwan Han. Syncsde: A probabilistic framework for diffusion synchronization. In *CVPR*, 2025.

Hyunsoo Lee, Minsoo Kang, and Bohyung Han. Conditional score guidance for text-driven image-to-image translation. *NeurIPS*, 2023a.

Hyunsoo Lee, Inwoo Hwang, and Young Min Kim. Accelerated likelihood maximization for diffusion-based versatile content generation. In *ECCV*, 2026.

Yuseung Lee, Kunho Kim, Hyunjin Kim, and Minhyuk Sung. Syncdiffusion: Coherent montage via synchronized joint diffusions. *NeurIPS*, 2023b.

Henry Li and Marcus Pereira. Solving inverse problems via diffusion optimal control. *NeurIPS*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *ICLR*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *ICLR*, 2023.

Yuxin Liu, Minshan Xie, Hanyuan Liu, and Tien-Tsin Wong. Text-guided texturing by synchronized multi-view diffusion. *SIGGRAPH Asia*, 2024.

Kushagra Pandey, Farrin Marouf Sofian, Felix Draxler, Theofanis Karaletsos, and Stephan Mandt. Variational control for guidance in diffusion models. *ICML*, 2025.

Minho Park, Taewoong Kang, Jooyeol Yun, Sungwon Hwang, and Jaegul Choo. Spherediff: Tuning-free 360° static and dynamic panorama generation via spherical latent representation. In *AAAI*, 2026.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 2019.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *ICLR*, 2023.

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 *ICML*, 2021.

Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *SIGGRAPH*, 2023.

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

Litu Rout, Yujia Chen, Nataniel Ruiz, Abhishek Kumar, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. RB-modulation: Training-free stylization using reference-based modulation. In *ICLR*, 2025.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

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

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *ICLR*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021b.

Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review. *arXiv:2407.13734*, 2024.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv:2508.02324*, 2025.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. SANA: Efficient high-resolution text-to-image synthesis with linear diffusion transformers. In *ICLR*, 2025.

Zhiyuan Xu, Yinhe Chen, Huan-ang Gao, Weiyan Zhao, Guiyu Zhang, and Hao Zhao. Diffusion-based visual anagram as multi-task learning. In *WACV*, 2025.

Kyeongmin Yeo, Jaihoon Kim, and Minhyuk Sung. Stochsync: Stochastic diffusion synchronization for image generation in arbitrary spaces. *ICLR*, 2025.

Kim Youwang, Tae-Hyun Oh, and Gerard Pons-Moll. Paint-it: Text-to-texture synthesis via deep convolutional texture map optimization and physically-based rendering. In *CVPR*, 2024.

Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. Paint3d: Paint anything 3d with lighting-less texture diffusion models. In *CVPR*, 2024.

Cheng Zhang, Qianyi Wu, Camilo Cruz Gambardella, Xiaoshui Huang, Dinh Phung, Wanli Ouyang, and Jianfei Cai. Taming stable diffusion for text to 360 panorama image generation. In *CVPR*, 2024a.

Hongkun Zhang, Zherong Pan, Congyi Zhang, Lifeng Zhu, and Xifeng Gao. Texpainter: Generative mesh texturing with multi-view consistency. In *SIGGRAPH*, 2024b.

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

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Bing Zhao, Chenfei Wu, Deqing Li, Hao Meng, Jiahao Li, Jie Zhang, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kuan Cao, et al. Qwen-image-2.0 technical report. *arXiv:2605.10730*, 2026.

This appendix derives the ELBO for the autoregressive formulation (Appendix A) and the objective for DDIM sampling (Appendix B), and provides pseudocode for our method (Appendix C). Then, it derives the bidirectional ELBO (Appendix D), presents implementation details with additional experimental results (Appendix E), and discusses societal impacts (Appendix F). Please also refer to the project website submission for additional results.

A Derivation of the ELBO (Eq. 5)

We use the joint states $\boldsymbol{X}_t = (\mathbf{x}_t^{(1)}, \dots, \mathbf{x}_t^{(N)})$, with $\boldsymbol{X} = \boldsymbol{X}_0$, and controls $\boldsymbol{U}_{1:T} = (\boldsymbol{U}_1, \dots, \boldsymbol{U}_T)$, where $\boldsymbol{U}_t = (\mathbf{u}_t^{(1)}, \dots, \mathbf{u}_t^{(N)})$, as defined in Sec. 2.2. In the autoregressive formulation, $\mathbf{u}_t^{(1)} = \mathbf{0}$ is fixed and only the controls for $n \geq 2$ are optimized. The prior over the joint trajectory factorizes as

$$p(\boldsymbol{X}_{0:T}) = \prod_{n=1}^{N} p(\mathbf{x}_T^{(n)}) \prod_{n=1}^{N} \prod_{t=1}^{T} p_{\boldsymbol{\phi}}(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}). \tag{10}$$

The posterior in Eq. 1 is the tilted distribution $p(\boldsymbol{X} \mid \boldsymbol{y}) = p(\boldsymbol{X}) \exp(r(\boldsymbol{y}, \boldsymbol{X})/\lambda)/Z(\boldsymbol{y})$, where $Z(\boldsymbol{y}) := \mathbb{E}_{p(\boldsymbol{X})}[\exp(r(\boldsymbol{y}, \boldsymbol{X})/\lambda)]$ depends on $\boldsymbol{y}$ only. We extend it to the joint trajectory as

$$\pi(\boldsymbol{X}_{0:T} \mid \boldsymbol{y}) := \frac{1}{Z(\boldsymbol{y})}\, p(\boldsymbol{X}_{0:T}) \exp(r(\boldsymbol{y}, \boldsymbol{X}_0)/\lambda)\,, \tag{11}$$

whose marginal over $\boldsymbol{X}_0$ is $p(\boldsymbol{X} \mid \boldsymbol{y})$. Writing out the variational distribution of Eqs. 4 and 6 patch-wise gives

$$q(\boldsymbol{X}_{0:T} \mid \boldsymbol{y}, \boldsymbol{U}_{1:T}) = q(\mathbf{x}_{0:T}^{(1)}) \prod_{n=2}^{N} q(\mathbf{x}_T^{(n)}) \prod_{n=2}^{N} \prod_{t=1}^{T} q(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}, \mathbf{x}_t^{(n-1)}, \mathbf{u}_t^{(n)}, \boldsymbol{y}). \tag{12}$$

The first trajectory is sampled from the pretrained prior and reverse sampling is initialized at the prior's terminal Gaussian; we therefore set $q(\mathbf{x}_{0:T}^{(1)}) = p(\mathbf{x}_{0:T}^{(1)})$ and $q(\mathbf{x}_T^{(n)}) = p(\mathbf{x}_T^{(n)})$ for $n \geq 2$. Non-negativity of the KL divergence between q and the target π yields

$$0 \leq D_{\mathrm{KL}}(q \,\|\, \pi) = \mathbb{E}_q[\log q(\boldsymbol{X}_{0:T} \mid \boldsymbol{y}, \boldsymbol{U}_{1:T}) - \log p(\boldsymbol{X}_{0:T})] - \frac{1}{\lambda}\, \mathbb{E}_q[r(\boldsymbol{y}, \boldsymbol{X})] + \log Z(\boldsymbol{y}). \tag{13}$$

Substituting Eqs. 10 and 12 and cancelling the factors that coincide under the assumed q,

$$\log p(\boldsymbol{X}_{0:T}) - \log q(\boldsymbol{X}_{0:T} \mid \boldsymbol{y}, \boldsymbol{U}_{1:T}) = \sum_{n=2}^{N} \sum_{t=1}^{T} \log \frac{p_{\boldsymbol{\phi}}(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)})}{q(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}, \mathbf{x}_t^{(n-1)}, \mathbf{u}_t^{(n)}, \boldsymbol{y})}. \tag{14}$$

The tower property turns each transition log-ratio into a negative KL divergence under the marginal $q(\mathbf{x}_t^{(n)}, \mathbf{x}_t^{(n-1)})$. Rearranging Eq. 13,

$$\begin{aligned} \log Z(\boldsymbol{y}) \geq\ & \frac{1}{\lambda}\, \mathbb{E}_q[r(\boldsymbol{y}, \boldsymbol{X})] \\ & - \sum_{n=2}^{N} \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_t^{(n)}, \mathbf{x}_t^{(n-1)}} \left[D_{\mathrm{KL}} \Big(q(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}, \mathbf{x}_t^{(n-1)}, \mathbf{u}_t^{(n)}, \boldsymbol{y}) \,\|\, p_{\boldsymbol{\phi}}(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}) \Big) \right]. \end{aligned} \tag{15}$$

Here $\log Z(\boldsymbol{y})$ plays the role of the log evidence and is independent of $\boldsymbol{U}_{1:T}$. Multiplying the bound by $\lambda > 0$ preserves its maximizer and yields the objective in Eq. 5. Under the factorization of Eq. 6, the joint KL divergence term in Eq. 5 equals the sum of per-patch KL divergence terms above. □

B Objective for the DDIM sampler

Under the DDIM (Song et al., 2021a) parameterization, $\boldsymbol{\mu}_{\boldsymbol{\phi}}\left(\mathbf{x}_t^{(n)}, t\right)$ is defined as follows:

$$\boldsymbol{\mu}_{\boldsymbol{\phi}}\left(\mathbf{x}_t^{(n)}, t\right) = \underbrace{\sqrt{\frac{\alpha_{t-1}}{\alpha_t}}}_{a_t} \mathbf{x}_t^{(n)} + \underbrace{\left(\sqrt{1-\alpha_{t-1}} - \sqrt{\frac{(1-\alpha_t)\alpha_{t-1}}{\alpha_t}} \right)}_{b_t} \epsilon_\theta(\mathbf{x}_t^{(n)}, t). \tag{16}$$

For $\bar{\boldsymbol{\mu}}_t^{(n)}$ in Eq. 7, we use an analogous parameterization:

$$\bar{\boldsymbol{\mu}}_t^{(n)} = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}}\bar{\mathbf{x}}_t^{(n)} + \left(\sqrt{1-\alpha_{t-1}} - \sqrt{\frac{(1-\alpha_t)\alpha_{t-1}}{\alpha_t}}\right)\bar{\epsilon}_\theta(\bar{\mathbf{x}}_t^{(n)}, t), \tag{17}$$

where $\bar{\epsilon}_\theta(\bar{\mathbf{x}}_t^{(n)}, t)$ is calculated using conditional score-based sampling (Dhariwal & Nichol, 2021; Song et al., 2021b; Lee et al., 2023a) as follows:

$$\bar{\epsilon}_\theta(\bar{\mathbf{x}}_t^{(n)}, t) = \epsilon_\theta(\bar{\mathbf{x}}_t^{(n)}, t) + \frac{\gamma}{2}\sqrt{1-\alpha_t}\nabla_{\mathbf{x}_t^{(n)}}\left\|f\left(\mathbf{x}_t^{(n-1)}, \boldsymbol{y}\right) - \bar{\mathbf{x}}_t^{(n)}\right\|_2^2. \tag{18}$$

Substituting into $\bar{\boldsymbol{\mu}}_t^{(n)} = a_t\bar{\mathbf{x}}_t^{(n)} + b_t\bar{\epsilon}_\theta(\bar{\mathbf{x}}_t^{(n)}, t)$ recovers Eq. 7 with step size $\sigma_t^2 = -b_t\sqrt{1-\alpha_t}$, which is positive since $b_t < 0$ for any monotone noise schedule. Using these parameterizations and starting from Eq. 8, we obtain the practical objective optimized with the DDIM sampler:

$$\begin{aligned}
\mathcal{J} &= -\tilde{r}_n\left(\boldsymbol{y}, \hat{\mathbf{x}}_{0|t}^{(1:n)}\right) + \frac{\lambda}{2\sigma_t^2}\left\|\bar{\boldsymbol{\mu}}_t^{(n)} - \boldsymbol{\mu}_{\boldsymbol{\phi}}\left(\mathbf{x}_t^{(n)}, t\right)\right\|_2^2 \\
&= -\tilde{r}_n\left(\boldsymbol{y}, \hat{\mathbf{x}}_{0|t}^{(1:n)}\right) + \frac{\lambda}{2\sigma_t^2}\left\|a_t\beta\mathbf{u}_t^{(n)} + b_t(\bar{\epsilon}_\theta(\bar{\mathbf{x}}_t^{(n)}, t) - \epsilon_\theta(\mathbf{x}_t^{(n)}, t))\right\|_2^2 \\
&\le -\tilde{r}_n\left(\boldsymbol{y}, \hat{\mathbf{x}}_{0|t}^{(1:n)}\right) + \frac{\lambda}{\sigma_t^2}a_t^2\beta^2\left\|\mathbf{u}_t^{(n)}\right\|_2^2 \\
&\quad + \frac{\lambda}{\sigma_t^2}b_t^2\left\|\epsilon_\theta(\bar{\mathbf{x}}_t^{(n)}, t) + \frac{\gamma}{2}\sqrt{1-\alpha_t}\nabla_{\mathbf{x}_t^{(n)}}\left\|f\left(\mathbf{x}_t^{(n-1)}, \boldsymbol{y}\right) - \bar{\mathbf{x}}_t^{(n)}\right\|_2^2 - \epsilon_\theta(\mathbf{x}_t^{(n)}, t)\right\|_2^2.
\end{aligned} \tag{19}$$

The last step uses $\|\mathbf{a}+\mathbf{b}\|_2^2 \le 2\|\mathbf{a}\|_2^2 + 2\|\mathbf{b}\|_2^2$, so that we minimize an upper bound on $\mathcal{J}$. We absorb the sampler-dependent factor $1/\sigma_t^2$ into λ and treat it as a single tunable constant; this also covers the deterministic DDIM limit ($\eta = 0$), in which the kernels become point masses and the mean discrepancy is retained as the regularizer. Here, we additionally omit the coefficient β^2 in the second term of Eq. 19, which empirically improves performance and yields the final objective presented in Eq. 20. The final objective optimized in all experiments is therefore

$$\begin{aligned}
\mathbf{u}_t^\star = \arg\min_{\mathbf{u}_t} \sum_{n=2}^{N} \mathbb{E}_{\mathbf{x}_t^{(n)}, \mathbf{x}_t^{(n-1)}} \Bigg[&-\tilde{r}_n\left(\boldsymbol{y}, \hat{\mathbf{x}}_{0|t}^{(1:n)}\right) + \lambda a_t^2\left\|\mathbf{u}_t^{(n)}\right\|_2^2 \\
&+\lambda b_t^2\left\|\epsilon_\theta(\bar{\mathbf{x}}_t^{(n)}, t) + \frac{\gamma}{2}\sqrt{1-\alpha_t}\nabla_{\mathbf{x}_t^{(n)}}\left\|f\left(\mathbf{x}_t^{(n-1)}, \boldsymbol{y}\right) - \bar{\mathbf{x}}_t^{(n)}\right\|_2^2 - \epsilon_\theta(\mathbf{x}_t^{(n)}, t)\right\|_2^2\Bigg],
\end{aligned} \tag{20}$$

where $\epsilon_\theta(\cdot,\cdot)$ denotes the noise prediction network,

$$a_t = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}} \quad \text{and} \quad b_t = \sqrt{1-\alpha_{t-1}} - \sqrt{\frac{(1-\alpha_t)\alpha_{t-1}}{\alpha_t}}. \tag{21}$$

C Pseudocode of SyncVC

Algorithm 1 provides a high-level overview of the practical implementation. In practice, controls are optimized sequentially in a greedy manner. Specifically, for each diffusion timestep t, instead of jointly optimizing $N-1$ control variables $\{\mathbf{u}_t^{(2)}, \ldots, \mathbf{u}_t^{(N)}\}$, we iterate over n and optimize each control variable $\mathbf{u}_t^{(n)}$ using the previously optimized variables $\{\mathbf{u}_t^{(2)}, \cdots, \mathbf{u}_t^{(n-1)}\}$. The optimization objective in Eq. 8 is decomposed into $N-1$ terms (*i.e.*, each term in the summation) and the $(n-1)$-th term is used as the optimization objective for $\mathbf{u}_t^{(n)}$. We adopt this setting since we empirically observe that greedy optimization yields better results.

D Derivation of the ELBO for the bidirectional formulation (Eq. 29)

We consider a bidirectional formulation in which each reverse transition conditions on the noisy states of its adjacent trajectories at the same timestep. We use the joint states $\boldsymbol{X}_t$ and controls $\boldsymbol{U}_{1:T}$

Algorithm 1 SyncVC

Inputs: Observation $\boldsymbol{y}$, Noisy latent variables $\mathbf{x}_T^{(1)}, \ldots, \mathbf{x}_T^{(N)}$, Denoising kernel $p_{\boldsymbol{\phi}}$, Number of optimization steps K, Hyperparameters γ, λ, β
for $t \leftarrow T, \cdots, 1$ **do**
  Initialize $\{\mathbf{u}_t^{(2)}, \ldots, \mathbf{u}_t^{(N)}\}$ to zero
  Calculate $\mathbf{x}_{t-1}^{(1)}$ using the denoising kernel $p_{\boldsymbol{\phi}}$
  for $n \leftarrow 2, \cdots, N$ **do**
    for $i \leftarrow 1, \cdots, K$ **do**
      Calculate the $(n-1)$-th term of the objective in Eq. 8
      Optimize $\mathbf{u}_t^{(n)}$ using the calculated optimization objective
    end for
    Calculate $\mathbf{x}_{t-1}^{(n)}$ using Eq. 7 with optimized $\mathbf{u}_t^{(n)}$ and $\mathbf{x}_t^{(n)}$
  end for
end for
Outputs: Sequence $\boldsymbol{X} = \{\mathbf{x}_0^{(1)}, \ldots, \mathbf{x}_0^{(N)}\}$

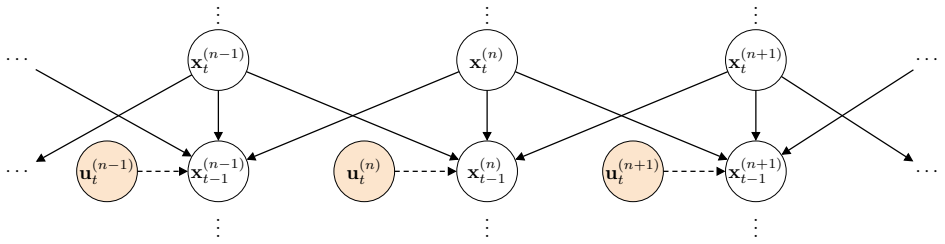


Figure 8: **Graphical model for bidirectional formulation.** Each reverse transition depends on its own noisy state and the states of its available neighbors at timestep t.

defined in Sec. 2.2. Let $\mathcal{N}(n) := \{n-1, n+1\} \cap \{1, \ldots, N\}$ denote the available neighbors of patch n, and write $\mathbf{x}_t^{\mathcal{N}(n)} := \{\mathbf{x}_t^{(j)} : j \in \mathcal{N}(n)\}$. Every trajectory is controlled, including the first, as illustrated in Figure 8, and its terminal distribution equals the prior, $q(\mathbf{x}_T^{(n)}) = p(\mathbf{x}_T^{(n)})$. The prior groups the independent factors by trajectory:

$$p(\boldsymbol{X}_{0:T}) = \prod_{n=1}^{N} p(\mathbf{x}_{0:T}^{(n)}), \qquad p(\mathbf{x}_{0:T}^{(n)}) = p(\mathbf{x}_T^{(n)}) \prod_{t=1}^{T} p_{\boldsymbol{\phi}}(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}). \tag{22}$$

As in Appendix A, the target is the tilted distribution $\pi(\boldsymbol{X}_{0:T} \mid \boldsymbol{y}) = p(\boldsymbol{X}_{0:T}) \exp(r(\boldsymbol{y}, \boldsymbol{X}_0)/\lambda)/Z(\boldsymbol{y})$ of Eq. 11. The variational distribution has the analogous grouping

$$q(\boldsymbol{X}_{0:T} \mid \boldsymbol{y}, \boldsymbol{U}_{1:T}) = \prod_{n=1}^{N} Q^{(n)}, \qquad Q^{(n)} = p(\mathbf{x}_T^{(n)}) \prod_{t=1}^{T} q\Big(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}, \mathbf{x}_t^{\mathcal{N}(n)}, \mathbf{u}_t^{(n)}, \boldsymbol{y}\Big). \tag{23}$$

Each $Q^{(n)}$ depends on the states of its neighboring trajectories; this grouping does not imply independence between trajectories. For fixed $\boldsymbol{X}_t$, $\boldsymbol{U}_t$, and $\boldsymbol{y}$, however, the transitions to $\boldsymbol{X}_{t-1}$ factorize across patches. All neighbor dependencies refer to time-t states, so the joint distribution remains a normalized directed model despite the two-sided coupling. The same neighbor notation covers the boundaries: $\mathcal{N}(1) = \{2\}$ and $\mathcal{N}(N) = \{N-1\}$.

Non-negativity of $D_{\mathrm{KL}}(q \,\|\, \pi)$ gives

$$\log Z(\boldsymbol{y}) \geq \frac{1}{\lambda} \mathbb{E}_q[r(\boldsymbol{y}, \boldsymbol{X})] + \mathbb{E}_q[\log p(\boldsymbol{X}_{0:T}) - \log q(\boldsymbol{X}_{0:T} \mid \boldsymbol{y}, \boldsymbol{U}_{1:T})]\,. \tag{24}$$

Substituting Eqs. 22 and 23, the log-ratio separates across trajectories:

$$\log p(\boldsymbol{X}_{0:T}) - \log q(\boldsymbol{X}_{0:T} \mid \boldsymbol{y}, \boldsymbol{U}_{1:T}) = \sum_{n=1}^{N} \log \frac{p(\mathbf{x}_{0:T}^{(n)})}{Q^{(n)}}. \tag{25}$$

For each n, the terminal prior factors cancel, leaving the timestep-wise expansion

$$\log \frac{p(\mathbf{x}_{0:T}^{(n)})}{Q^{(n)}} = \sum_{t=1}^{T} \log \frac{p_{\boldsymbol{\phi}}(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)})}{q(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}, \mathbf{x}_t^{\mathcal{N}(n)}, \mathbf{u}_t^{(n)}, \boldsymbol{y})}. \tag{26}$$

Conditioning on $\mathbf{x}_t^{(n)}$ and $\mathbf{x}_t^{\mathcal{N}(n)}$ turns each transition log-ratio into a negative conditional KL divergence. Applying the tower property therefore yields

$$\mathbb{E}_q\left[\log \frac{p(\mathbf{x}_{0:T}^{(n)})}{Q^{(n)}}\right] = -\mathcal{K}_n, \tag{27}$$

$$\mathcal{K}_n = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{x}_t^{(n)}, \mathbf{x}_t^{\mathcal{N}(n)} \mid \boldsymbol{y}, \boldsymbol{U}_{1:T})}\Big[D_{\mathrm{KL}}\Big(q\Big(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}, \mathbf{x}_t^{\mathcal{N}(n)}, \mathbf{u}_t^{(n)}, \boldsymbol{y}\Big) \,\Big\|\, p_{\boldsymbol{\phi}}\Big(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}\Big)\Big)\Big].$$

Combining these terms gives

$$\log Z(\boldsymbol{y}) \geq \frac{1}{\lambda}\mathbb{E}_q[r(\boldsymbol{y}, \boldsymbol{X})] - \sum_{n=1}^{N} \mathcal{K}_n. \tag{28}$$

Multiplying by $\lambda > 0$ gives the final bidirectional objective:

$$\mathcal{L}(\boldsymbol{y}) = \mathbb{E}_q[r(\boldsymbol{y}, \boldsymbol{X})] \tag{29}$$
$$- \lambda \sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{x}_t^{(n)}, \mathbf{x}_t^{\mathcal{N}(n)} \mid \boldsymbol{y}, \boldsymbol{U}_{1:T})}\Big[D_{\mathrm{KL}}\Big(q\Big(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}, \mathbf{x}_t^{\mathcal{N}(n)}, \mathbf{u}_t^{(n)}, \boldsymbol{y}\Big) \,\Big\|\, p_{\boldsymbol{\phi}}\Big(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}\Big)\Big)\Big].$$

Unlike the autoregressive formulation, the sum includes the first trajectory because its transitions are also controlled.

Analogously to Eq. 8, we evaluate the objective at the current samples during inference, approximating $\mathbb{E}_q[r(\boldsymbol{y}, \boldsymbol{X})]$ through Tweedie's estimates and each KL through the discrepancy between the corresponding Gaussian means. The per-step objective over $\boldsymbol{U}_t$ becomes

$$\boldsymbol{U}_t^{\star} = \arg\min_{\boldsymbol{U}_t} \; -\sum_{n=2}^{N} \tilde{r}_n\Big(\boldsymbol{y}, \hat{\mathbf{x}}_{0|t}^{(1:n)}\Big) + \frac{\lambda}{2\sigma_t^2} \sum_{n=1}^{N} \Big\| \bar{\boldsymbol{\mu}}_t^{(n)} - \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big(\mathbf{x}_t^{(n)}, t\Big)\Big\|_2^2, \tag{30}$$

where $\bar{\mathbf{x}}_t^{(n)} = \mathbf{x}_t^{(n)} + \beta \mathbf{u}_t^{(n)}$ and $\hat{\mathbf{x}}_{0|t}^{(n)} = \mathbb{E}[\mathbf{x}_0 \mid \bar{\mathbf{x}}_t^{(n)}]$ now hold for all $n \geq 1$, and

$$\bar{\boldsymbol{\mu}}_t^{(n)} = \begin{cases} \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big(\bar{\mathbf{x}}_t^{(1)}, t\Big) - \dfrac{\gamma}{2}\sigma_t^2 \nabla_{\mathbf{x}_t^{(1)}} \Big\| f^{-1}\Big(\mathbf{x}_t^{(2)}, \boldsymbol{y}\Big) - \bar{\mathbf{x}}_t^{(1)}\Big\|_2^2 & \text{if } n = 1, \\ \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big(\bar{\mathbf{x}}_t^{(n)}, t\Big) - \dfrac{\gamma}{2}\sigma_t^2 \nabla_{\mathbf{x}_t^{(n)}} \Big(\big\| f\big(\mathbf{x}_t^{(n-1)}, \boldsymbol{y}\big) - \bar{\mathbf{x}}_t^{(n)}\big\|_2^2 & \\ \qquad + \big\| f^{-1}\big(\mathbf{x}_t^{(n+1)}, \boldsymbol{y}\big) - \bar{\mathbf{x}}_t^{(n)}\big\|_2^2\Big) & \text{if } 2 \leq n \leq N-1, \\ \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big(\bar{\mathbf{x}}_t^{(N)}, t\Big) - \dfrac{\gamma}{2}\sigma_t^2 \nabla_{\mathbf{x}_t^{(N)}} \Big\| f\Big(\mathbf{x}_t^{(N-1)}, \boldsymbol{y}\Big) - \bar{\mathbf{x}}_t^{(N)}\Big\|_2^2 & \text{if } n = N. \end{cases}$$

Here, $f^{-1}(\cdot, \boldsymbol{y})$ denotes the reverse-direction operator associated with $f(\cdot, \boldsymbol{y})$, such as the reverse spatial shift or inverse view transformation. The three cases above explicitly expand the neighbors in $\mathcal{N}(n)$: f acts on the predecessor when present, and f^{-1} acts on the successor when present.

E Additional experimental results

Our method is implemented in PyTorch (Paszke et al., 2019), and control variables are optimized using Adam (Kingma & Ba, 2015). We use DDIM (Song et al., 2021a) sampling and classifier-free guidance (Ho & Salimans, 2021) across all tasks. The following subsections specify reward design, task-specific settings, reference image construction, and metric computation.

E.1 WIDE IMAGE GENERATION

Reward. Given a text prompt $\boldsymbol{y}$, each sequence element $\mathbf{x}_0^{(n)}$ represents an image patch. In the wide image generation task, f uses only the preceding patch $\mathbf{x}_0^{(n-1)}$ and shifts it along the x-axis by the patch stride, while $\mathbf{M}^{(n)}$ selects the overlap region. This reward encourages consistency between the left side of the n-th patch and the right side of the $(n-1)$-th patch. We also evaluate style transfer guidance (Gatys et al., 2016) in Sec. 3.3, which encourages agreement with a reference image's style, and CLIP guidance (Radford et al., 2021) in Appendix E.4, which encourages alignment between $\mathbf{x}_0^{(n)}$ and the text prompt $\boldsymbol{y}$, as additional reward terms.

Experimental details. We generate wide images at a resolution of 2048×512 for evaluation, with each patch of size 512^2. For our method, five distinct trajectories are sampled with an overlap of 128 pixels. The generated patches are then sequentially concatenated so that the patch from the n-th trajectory is placed on top of that from the $(n+1)$-th trajectory, yielding a single wide image. Each control variable $\mathbf{u}_t^{(n)}$ is optimized for five steps with a learning rate of 10^{-2} using the Adam optimizer (Kingma & Ba, 2015). Hyperparameters are set to $\beta = 1.0$, $\gamma = 2.5$, and $\lambda = 2.0$. Across all methods, we use the pretrained Stable Diffusion v2.1-base (Rombach et al., 2022) with 50 steps of DDIM (Song et al., 2021a) sampling and classifier-free guidance (Ho & Salimans, 2021) to ensure a fair comparison. For baselines, we run their official implementations. In addition, we follow the baseline setup for noise initialization. Instead of independently sampling noise for each patch, a single wide latent noise map is first sampled from a Gaussian distribution, then cropped into the corresponding patch regions for each trajectory. We adopt the same process to ensure fairness in comparison.

To measure Intra-LPIPS (Zhang et al., 2018), Intra-Style-Loss (Gatys et al., 2016), χ^2-Histogram distance, and Histogram intersection, each wide image is cropped into four non-overlapping views of size 512^2, and the distances over all six pairwise combinations are calculated. Note that the color histograms are computed in the HSV space. KID (Bińkowski et al., 2018) score is calculated using randomly cropped 512^2 views from each wide image. These five metrics are measured across all prompts and then averaged. Reference images for KID measurement are constructed by generating 1,000 images per prompt using the pretrained Stable Diffusion v1.5 (Rombach et al., 2022).

Details of style guidance and the CLIP-augmented reward are provided in Appendix E.4.

Robustness to small overlaps. For the sample shown in Figure 4 of the main paper, we additionally visualize the full figure including the results of SyncTweedies (Kim et al., 2024) and SyncSDE (Lee et al., 2025) in Figure 9. This corresponds to the wide image generation setting with a small overlap of only 16 pixels (over the patch width of 512 pixels). As shown, baselines exhibit noticeable color inconsistencies between patches, while the proposed method produces consistent outputs.

Additional qualitative results. We show additional qualitative comparisons with baselines in Figure 10, and additional results of our method in Figure 11. Furthermore, we show that the proposed method can also be applied to a recent generative model with stronger priors that synthesize high-resolution images. Specifically, we use the pretrained SANA model (Xie et al., 2025) to generate wide images at resolutions of 4096×1024 and 8192×2048, and visualize the results in Figure 12 and 13.



Figure 9: **Our method demonstrates superior performance in wide image generation under a small-overlap setting**, maintaining strong style and color consistency across the horizontal axis. All baseline methods exhibit significant color changes.

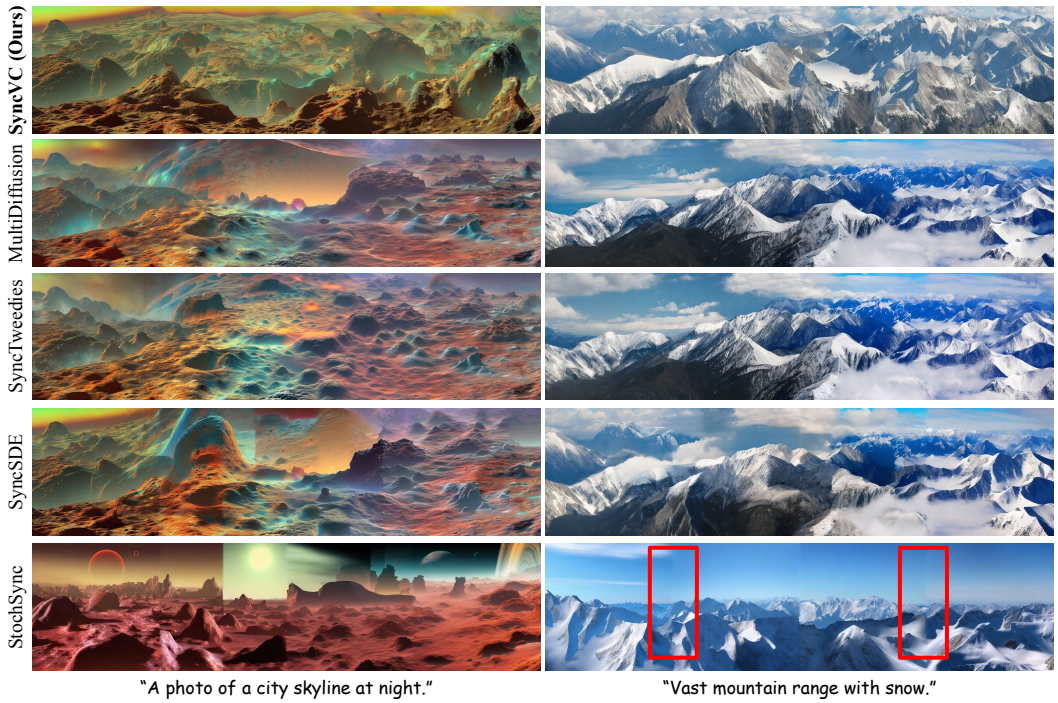


Figure 10: **Our method shows superior performance in wide image generation.** (Left) SyncVC maintains a unified color and style, while baselines suffer from varying mountain and sky colors, or discontinuities (see bounding box). (Right) Our method generates a cartoon-like wide image with consistent styles of tree and flowers, while baselines result in artifacts with inconsistent colors or discontinuities (see bounding box).

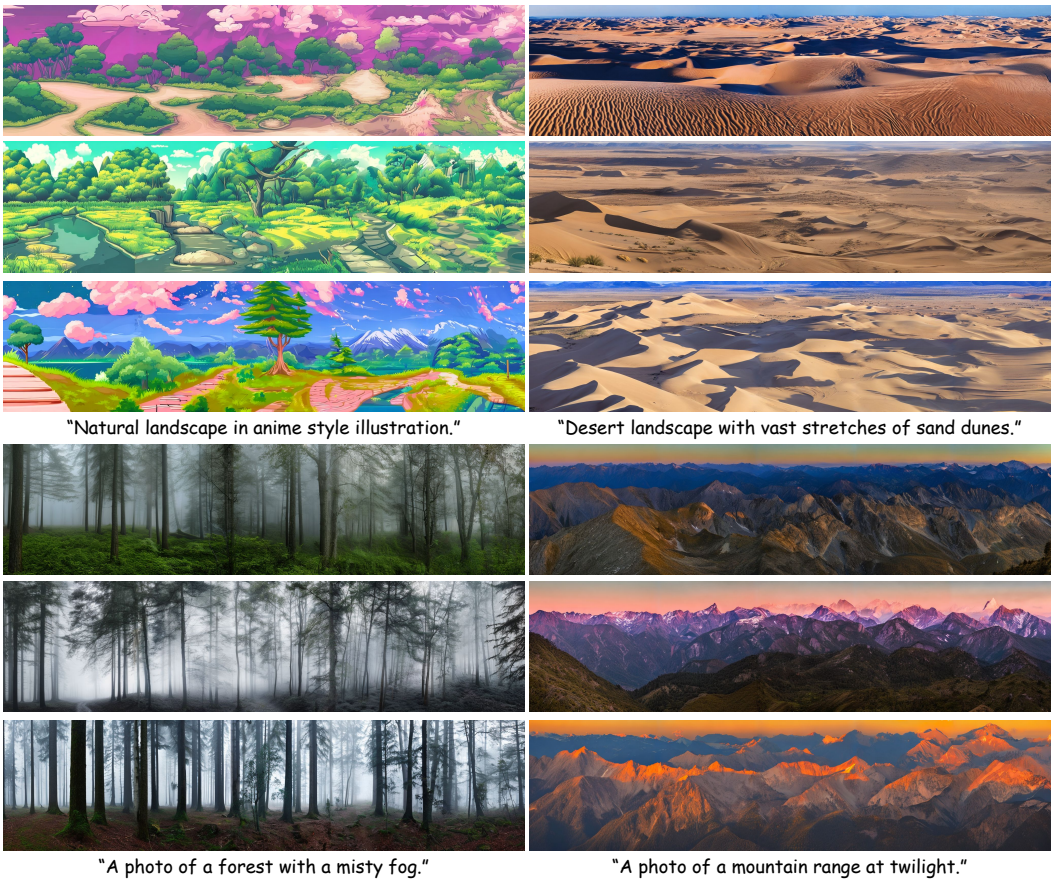


Figure 11: **Our method generates high-quality wide images conditioned on diverse text prompts.** We present multiple wide image samples generated using the pretrained Stable Diffusion (Rombach et al., 2022) for various text prompts, all exhibiting strong style consistency.



Figure 12: **Our method can also synthesize high-resolution wide images when combined with the SANA model (Xie et al., 2025).** We visualize various wide images at the resolution of 4096×1024 using the pretrained SANA model, where each generated patch has a resolution of 1024^2.



Figure 13: **Our method is even capable of generating wide imagews at a resolution of** 8192×2048**.** We use the pretrained SANA model (Xie et al., 2025) that generates patches at a resolution of 2048^2, and extend it along the horizontal axis using our method to generate ultra-high-resolution images.

E.2 TEXT-GUIDED 3D MESH TEXTURING

Reward. Here, $\boldsymbol{y}$ consists of the text prompt and source mesh, and $\mathbf{x}_0^{(n)}$ denotes the image from the n-th viewpoint. In Eq. 9, f first bakes an auxiliary texture from all previously generated views $\{\mathbf{x}_0^{(j)}\}_{j=1}^{n-1}$, then renders it from the n-th viewpoint. The mask $\mathbf{M}^{(n)}$ selects the foreground region in the rendered image. The reward encourages the current view to agree with the appearance implied by the texture assembled from earlier views.

Experimental details. We use the pretrained Stable Diffusion v1.5 (Rombach et al., 2022) with the pretrained depth-conditioned ControlNet (Zhang et al., 2023) for synchronization-based methods (Kim et al., 2024; Lee et al., 2025; Yeo et al., 2025) (including ours), and Stable Diffusion v2-depth model for task-specific methods (Zhang et al., 2024b; Richardson et al., 2023), following their original configuration. Regarding the viewpoint setting, we fix the elevation to 15° and uniformly sample eight azimuth angles from $[0^\circ, 360^\circ)$, resulting in eight diffusion trajectories. We use 8 DDIM (Song et al., 2021a) steps, with resolution of 768^2 for each patch. Meanwhile, we follow the default viewpoint sampling and diffusion sampling configurations for baselines and run their official implementations. For all methods, we prepend the prompt with the phrase "Best quality, extremely detailed" and use classifier-free guidance (Ho & Salimans, 2021) with the negative prompt "oversmoothed, blurry, depth of field, out of focus, low quality, bloom, glowing effect". To synthesize the texture map, we optimize it by minimizing the distance between the rendered view of the texture-projected mesh and the generated patch at each viewpoint using the SGD optimizer. Source meshes used for experiments are borrowed from the Objaverse dataset (Deitke et al., 2022). Following prior works (Liu et al., 2024; Kim et al., 2024; Lee et al., 2025), we apply Voronoi diagram-augmented filling (Aurenhammer, 1991) and a modified self-attention mechanism in the noise prediction network. The latent texture map resolution is set to 1536^2, and the RGB texture map resolution is 1024^2. Each control variable $\mathbf{u}_t^{(n)}$ is optimized for three iterations with a learning rate of 10^{-2} using the Adam optimizer (Kingma & Ba, 2015). We use $\beta = 1.0$, $\gamma = 0.1$, and $\lambda = 2.0$ as default hyperparameters.

For evaluation, we render the textured meshes from 10 different viewpoints using the PyTorch3D renderer (Ravi et al., 2020). Eight views have an elevation of 0° and azimuths uniformly sampled from $[0^\circ, 360^\circ)$, while two additional views have an elevation of 30° with azimuths of 0° and 180°, corresponding to the front and back views, respectively. FID (Heusel et al., 2017) and KID (Bińkowski et al., 2018) scores are calculated between the rendered images and the reference sets for each (mesh, prompt) pair, then averaged. For each (mesh, prompt) pair, we render depth maps from the same 10 viewpoints that are used for evaluation and generate 50 images per depth map using the pretrained depth-conditioned ControlNet, resulting in 500 reference images. CLIP-S (Radford et al., 2021) is measured by averaging the CLIP score between each of the 10 rendered views and the corresponding prompt for every (mesh, prompt) pair. For qualitative visualization in Figure 5, we use Nvdiffrast rasterizer (Laine et al., 2020) for more sophisticated rasterization.

Additional qualitative results. We show additional qualitative results of SyncVC on the text-guided 3D mesh texturing task in Figure 14. As shown, our method synthesizes textures that are not only realistic but also rich in fine-grained details.

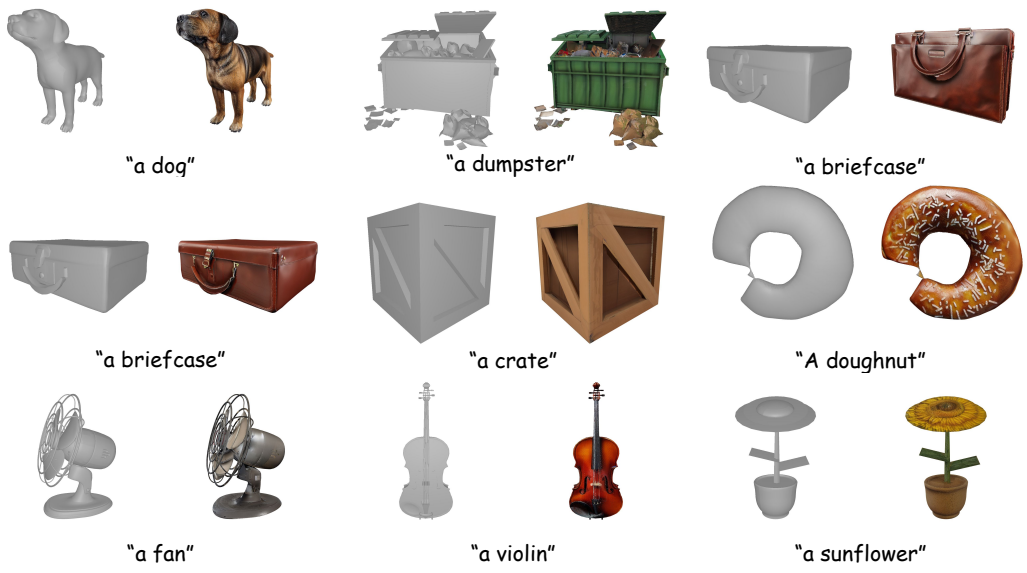


Figure 14: [Best viewed when magnified.] **Our method generates artifact-free and realistic textures for diverse 3D meshes.**

E.3 Optical illusion generation

The optical illusion generation task aims to create a single image that contains different subjects depending on how it is viewed. For example, the same image should depict a horse in its original orientation and a snowy mountain village when rotated clockwise (Figure 15, right). Given two text prompts and a specified transformation, the task is to find one image that satisfies the first prompt in its original view and the second prompt in its transformed view.

Reward. We use $N = 2$ diffusion trajectories, each conditioned on one of the two text prompts in $\boldsymbol{y}$. Their outputs, $\mathbf{x}_0^{(1)}$ and $\mathbf{x}_0^{(2)}$, are coupled through Eq. 9 with $n = 2$ and $\mathbf{M}^{(2)} = \mathbf{1}$, where f applies the specified rotation or vertical flip to the first image. The reward encourages the transformed first image to match the second, so that the two outputs define different views of a single image.

Setup and baselines. Our method generates 256^2 images using the pretrained two-stage DeepFloyd-IF model (DeepFloyd Lab, 2023), with two diffusion trajectories corresponding to the two prompt-conditioned views. We compare against synchronization methods SyncTweedies (Kim et al., 2024) and SyncSDE (Lee et al., 2025), and the task-specific method Anagram-MTL (Xu et al., 2025). The two stages use IF-I-M-v1.0 and IF-II-M-v1.0, respectively, and all baselines are run using their official implementations. We use five evaluation configs from prior work (Geng et al., 2024; Lee et al., 2025), each specifying a transformation and two text prompts, and generate 100 images per config. For our method, guidance is applied only during the first-stage sampling. We use 30 DDIM (Song et al., 2021a) sampling steps with classifier-free guidance (Ho & Salimans, 2021), and the noise scale parameter for the second stage of the DeepFloyd-IF model is fixed at 50 across all methods. Each control variable $\mathbf{u}_t^{(n)}$ is optimized for five iterations with a learning rate of 10^{-2} using the Adam optimizer (Kingma & Ba, 2015). We use $\beta = 2.0$, $\gamma = 0.05$, and $\lambda = 0.2$ as default hyperparameters.

Evaluation protocol. For evaluation, two views are sampled from each generated image. The 2nd view is obtained directly from the second trajectory, while the 1st view is constructed by applying the inverse illusion transformation to the second view (*e.g.*, a counterclockwise rotation for a clockwise illusion transformation). Note that each view is associated with its corresponding prompt. FID (Heusel et al., 2017) and KID (Bińkowski et al., 2018) values are computed between the images of each view and the reference images for each (transformation, prompt) pair, then averaged. We generate the reference images by synthesizing 1,000 images per prompt using the pretrained Stable Diffusion v1.5 (Rombach et al., 2022). Furthermore, MUSIQ (Ke et al., 2021) is computed for both views, and the scores are averaged over all generated images.

Results. The proposed method achieves the lowest FID and KID and the highest MUSIQ among the evaluated baselines (Table 4), indicating strong distributional alignment and image quality. Figure 15 shows that our method produces clear images that depict both prompt-specified meanings under the transformation. In comparison, SyncTweedies tends to produce blurry images, while SyncSDE and Anagram-MTL sometimes fail to capture one of the two meanings. Figure 16 shows additional qualitative results of our method on two additional illusion transformations. These results demonstrate that our method supports consistency between views related by an image transformation.

Table 4: Quantitative evaluation on optical illusion generation. Our method achieves the lowest FID (Heusel et al., 2017) and KID (Bińkowski et al., 2018) for distributional alignment and the highest MUSIQ (Ke et al., 2021) for image quality. KID score is scaled by 10^3.

Metric	**SyncVC (Ours)**	SyncTweedies	SyncSDE	Anagram-MTL
MUSIQ ↑	**54.40**	32.28	<u>52.74</u>	41.92
FID ↓	**252.87**	<u>255.37</u>	264.10	256.96
KID↓	**175.47**	195.52	<u>176.14</u>	190.64

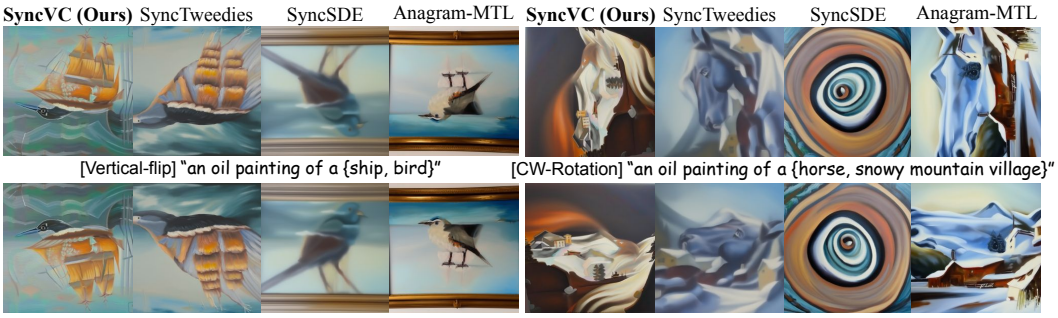


Figure 15: [Best viewed when magnified.] **Our method generates clear images that encode both prompt-specified semantics under the illusion transformation.** For each method, the top and bottom rows show the original and transformed views of the same image. Note that Anagram-MTL (Xu et al., 2025) is tailored specifically for optical illusion generation, whereas our method uses a general synchronization framework without requiring task-specific redesign. (Left) SyncSDE (Lee et al., 2025) and Anagram-MTL struggle to depict the bird and ship, respectively. (Right) SyncTweedies (Kim et al., 2024) produces a blurry image, while SyncSDE fails to capture the overall shape of both horse and village.

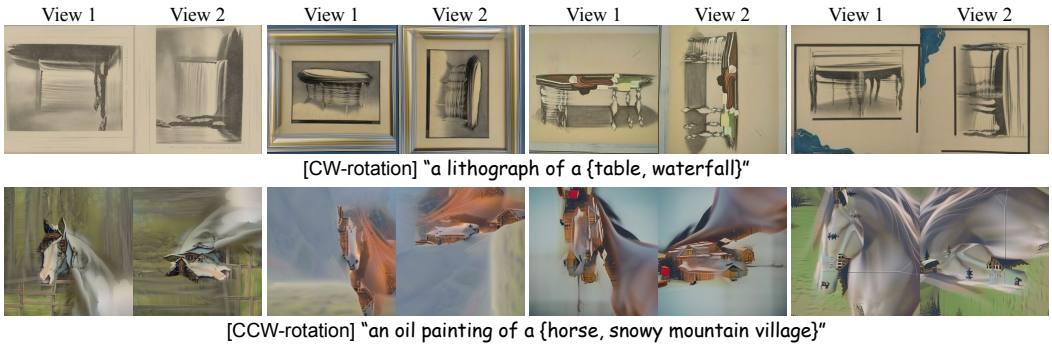


Figure 16: **Our method shows superior performance on the optical illusion generation task** by clearly incorporating two semantics specified by different text prompts. (Row 1) The generated image can be viewed as both a table and a waterfall under clockwise rotation. (Row 2) Each view encodes both a horse and a snowy mountain village under counterclockwise rotation.

E.4 Additional analysis

E.4.1 Model factorization and control optimization

Tweedie estimates and full-rollout optimization. As shown in Eq. 8, our method evaluates the reward using Tweedie estimates, enabling control optimization without backpropagating through the entire diffusion chain. Such estimates are used in diffusion guidance and reward-based fine-tuning (Clark et al., 2024) and are discussed as a practical approximation for tractable value estimation in a tutorial on RL-based fine-tuning of diffusion models (Uehara et al., 2024). To evaluate this approximation together with autoregressive greedy control optimization, we compare our method against full-rollout joint optimization, which evaluates the reward after rolling out the diffusion chain and backpropagates through the rollout. Full-rollout joint optimization exceeds the memory capacity of a single NVIDIA A6000 GPU at the default $T = 50$ in our implementation, so we compare both strategies with $T = 8$ and $N = 2$. Table 5 shows that our method improves four of the five quality metrics and achieves a shorter runtime.

Table 5: Quantitative comparison of our method with full-rollout joint control optimization at $T = 8$ and $N = 2$. Our method improves four of the five quality metrics and reduces runtime, supporting the combined use of Tweedie estimates and greedy control optimization. KID score is scaled by 10^3.

Metric	**SyncVC (Ours)**	Full-rollout joint optimization
χ^2-Histogram dist. ↓	**0.56**	0.98
Intra-Style-Loss ↓	**15.90**	35.14
Histogram intersect. ↑	**0.72**	0.60
KID ↓	77.22	**76.17**
Intra-LPIPS ↓	**0.60**	0.65
Runtime (s/image) ↓	**10.48**	18.07

E.4.2 Runtime analysis

Runtime–quality trade-off. We evaluate the runtime of each method on a single NVIDIA A6000 GPU using its official implementation. Table 6 shows the quantitative results across all evaluation metrics for different choices of T and K. With $T = 18$ and $K = 2$, our method is approximately $1.7\times$ faster than MultiDiffusion and $5.6\times$ faster than our default configuration. This faster configuration still generally improves over MultiDiffusion (Bar-Tal et al., 2023).

Table 6: Runtime–quality trade-off in wide image generation with fewer denoising steps T and optimization steps K. Although reducing T lowers quality relative to the default configuration, our method maintains strong performance against the baseline while achieving a shorter runtime with $T = 18$ and $K = 2$. KID score is scaled by 10^3.

Metric	**SyncVC** ($T = 18$, $K = 2$)	**SyncVC** ($T = 18$, $K = 5$)	**SyncVC** ($T = 25$, $K = 5$)	**SyncVC** ($T = 50$, $K = 5$, default)	MultiDiffusion
χ^2-Histogram dist. ↓	1.00	0.87	0.81	**0.75**	1.21
Intra-Style-Loss ↓	38.10	**35.36**	38.95	44.34	58.46
Histogram intersect. ↑	0.60	0.64	0.65	**0.67**	0.55
KID ↓	59.76	55.74	53.35	**52.07**	58.26
Intra-LPIPS ↓	0.62	0.60	0.60	**0.59**	0.64
Runtime (s/image) ↓	**41.19**	94.28	122.01	232.41	68.70

Runtime comparison with baselines. Table 7 compares runtime and color consistency across wide image generation methods. All three configurations of our method in the table achieve lower χ^2-Histogram distance than the baselines. The $T = 18$, $K = 2$ configuration is faster than MultiDiffusion, SyncTweedies, and StochSync, while SyncSDE has the shortest runtime. Our default configuration achieves the lowest χ^2-Histogram distance at a higher runtime.

E.4.3 Flexibility in reward design

We detail two extensions of the wide image consistency reward: style guidance, illustrated in Sec. 3.3, and CLIP guidance for text–image alignment. When additional guidance is applied to all patches,

Table 7: Quantitative runtime and performance measurements for wide image generation.

Method	Runtime (s/image)	χ^2-Histogram dist. ↓
SyncVC ($T = 50$, $K = 1$)	76.70	0.91
SyncVC ($T = 18$, $K = 2$)	41.19	1.00
SyncVC ($T = 50$, $K = 5$, default)	232.41	**0.75**
MultiDiffusion	68.70	1.21
SyncTweedies	73.98	1.35
SyncSDE	**17.51**	1.31
StochSync	42.37	1.35

as in both CLIP-guided and stylized wide image generation, we also optimize the first control $\mathbf{u}_t^{(1)}$, which is otherwise fixed to zero. The ELBO therefore includes an additional KL term associated with $\mathbf{u}_t^{(1)}$, yielding

$$\mathcal{L}(\boldsymbol{y}) := \mathbb{E}_q[r(\boldsymbol{y}, \boldsymbol{X})] - \lambda \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_t^{(1)}} \left[D_{\mathrm{KL}} \left(q\left(\mathbf{x}_{t-1}^{(1)} \mid \mathbf{x}_t^{(1)}, \mathbf{u}_t^{(1)}, \boldsymbol{y} \right) \parallel p\left(\mathbf{x}_{t-1}^{(1)} \mid \mathbf{x}_t^{(1)} \right) \right) \right]$$
$$- \lambda \sum_{n=2}^{N} \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_t^{(n)}, \mathbf{x}_t^{(n-1)}} \left[D_{\mathrm{KL}} \left(q\left(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)}, \mathbf{x}_t^{(n-1)}, \mathbf{u}_t^{(n)}, \boldsymbol{y} \right) \parallel p\left(\mathbf{x}_{t-1}^{(n)} \mid \mathbf{x}_t^{(n)} \right) \right) \right]. \quad (31)$$

Style guidance. We incorporate style guidance by subtracting a style transfer loss between a style reference image and $\mathbf{x}_0^{(n)}$ $(1 \le n \le N)$ to the wide image specialization of Eq. 9. Since style guidance also applies to the first patch $\mathbf{x}_t^{(1)}$, stylized wide image generation uses the all-patch guidance objective in Eq. 31.

Style references used for experiments are borrowed from the internet. Specifically, following Gatys et al. (Gatys et al., 2016), the style transfer loss is defined as a weighted sum of a content loss and a style loss. Let $\mathbf{F}(\mathbf{I}) \in \mathbb{R}^{C \times M}$ denote the features of an image $\mathbf{I}$ extracted using the pretrained VGG network (Simonyan & Zisserman, 2015), where C is the total number of feature maps and M is the spatial resolution of each feature map. The content loss is defined as the squared error between features of two images:

$$\mathcal{L}_{\text{content}}(\mathbf{I}_1, \mathbf{I}_2) = \frac{1}{2} \|\mathbf{F}(\mathbf{I}_1) - \mathbf{F}(\mathbf{I}_2)\|_F^2 \quad (32)$$

where $\mathbf{I}_1$ denotes the content image and $\mathbf{I}_2$ is the stylized image. In the stylized wide image generation scenario, the content image is defined as $\mathbf{x}_0^{(n)}$ obtained without optimizing controls. For the style representation of image $\mathbf{I}$, we use the Gram matrix $\mathbf{G}(\mathbf{I}) \in \mathbb{R}^{C \times C}$:

$$\mathbf{G}(\mathbf{I})[i, j] = \sum_k \mathrm{Vec}(\mathbf{F}_{ik}(\mathbf{I}))\mathrm{Vec}(\mathbf{F}_{jk}(\mathbf{I})) \quad \text{for} \quad 1 \le i, j \le C, \quad (33)$$

where $\mathrm{Vec}(\cdot)$ stands for the vectorization operation of a given matrix. The style loss is then defined as

$$\mathcal{L}_{\text{style}}(\mathbf{I}_2, \mathbf{I}_3) = \frac{1}{4C^2M^2} \sum_{i,j} \left(G_{ij}(\mathbf{I}_2) - G_{ij}(\mathbf{I}_3) \right)^2, \quad (34)$$

where $\mathbf{I}_3$ denotes the style reference image. In practice, we use features from multiple layers of the VGG network to calculate the style loss and use the averaged value. The style transfer loss is finally given by

$$\mathcal{L}_{\text{style-transfer}} = w_{\text{content}} \mathcal{L}_{\text{content}} + w_{\text{style}} \mathcal{L}_{\text{style}}. \quad (35)$$

We choose $w_{\text{content}} = 1.0$ and $w_{\text{style}} = 10^{-4}$.

CLIP-augmented reward. To complement the style-guidance experiment in Sec. 3.3, we combine the wide image consistency reward with a text–image similarity term from CLIP (Radford et al., 2021):

$$\tilde{r}_n\left(\boldsymbol{y}, \mathbf{x}_0^{(1:n)} \right) = -\frac{\gamma}{2} \left\| \mathbf{M} \odot \left(f\left(\mathbf{x}_0^{(n-1)} \right) - \mathbf{x}_0^{(n)} \right) \right\|_2^2 + \lambda^{\text{clip}} \cdot \mathrm{Sim}\left(f^{\text{img}}(\mathbf{x}_0^{(n)}), f^{\text{txt}}(\boldsymbol{y}) \right), \quad (36)$$

where f^{img} and f^{txt} denote the image and text encoders of the pretrained CLIP model, respectively, $\text{Sim}(\cdot, \cdot)$ denotes cosine similarity, and λ^{clip} is a scalar hyperparameter. We set $\lambda^{\text{clip}} = 0.05$ and apply the CLIP guidance term to all patches, *i.e.*, for $1 \leq n \leq N$. Note that for $n = 1$, we omit the consistency term and apply only the CLIP guidance term.

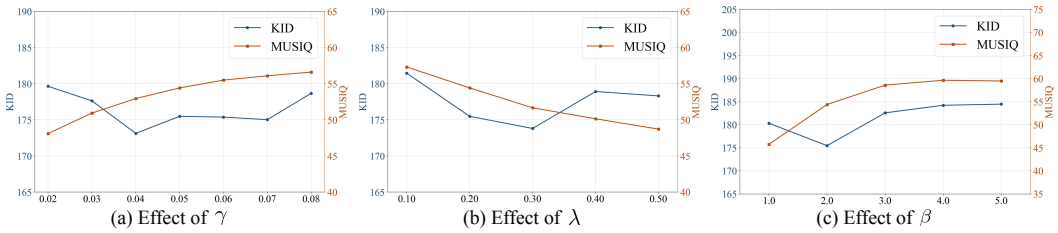
Table 8 reports the quantitative results for the CLIP-augmented reward in Eq. 36 (the "SyncVC*" column). This variant outperforms the best baseline, MultiDiffusion (Bar-Tal et al., 2023), across all reported metrics, although its scores are slightly worse than those obtained with the original reward. Together, these results demonstrate that our method can incorporate additional guidance through different reward terms while maintaining strong performance.

Table 8: Quantitative evaluation of wide image generation with a CLIP-augmented reward. SyncVC* denotes our method using the reward function in Eq. 36. Both reward variants outperform the best baseline, MultiDiffusion (Bar-Tal et al., 2023), across all metrics, showing that our method maintains strong performance when an additional guidance term is incorporated. KID (Bińkowski et al., 2018) score is scaled by 10^3.

Metric	**SyncVC (Ours, Eq. 9)**	**SyncVC* (Ours, Eq. 36)**	MultiDiffusion
χ^2-Histogram dist. ↓	**0.75**	0.78	1.21
Intra-Style-Loss ↓	**44.34**	47.13	58.46
Histogram intersect. ↑	**0.67**	0.66	0.55
KID ↓	**52.07**	53.43	58.26
Intra-LPIPS ↓	**0.59**	0.59	0.64

E.4.4 Effects of hyperparameters

Our parameterization contains three tunable coefficients: the weight of the reward function (γ), the weight of the KL-divergence term in the ELBO (λ), and the control strength (β). Figure 17 shows the effects of these coefficients in optical illusion generation. First, increasing γ improves the depiction of both prompt-specified meanings, leading to better KID scores. However, excessively large γ over-constrains the 2nd trajectory to resemble the 1st one, causing only one semantic to dominate. As a result, although the visual quality (MUSIQ) may improve, both semantics are no longer jointly captured and KID degrades. Secondly, increasing λ regularizes the 2nd trajectory toward the original diffusion prior, thereby reducing the influence of the 1st trajectory. This produces an effect analogous to decreasing γ. Lastly, increasing β initially improves the overall quality as the controls strongly guide the trajectory to satisfy the objective in Eq. 8. However, an excessively large β may weaken the ability to jointly capture both semantics.



(a) Effect of γ (b) Effect of λ (c) Effect of β

Figure 17: **Effects of hyperparameters (γ, λ, β) on the optical illusion generation task.** These values offer a trade-off between jointly capturing both semantics (KID) and visual quality (MUSIQ).

E.4.5 Visualization of controls

To provide intuition on the role of controls, we visualize the optimized control variables for optical illusion generation in Figure 18. At early timesteps (large t), the controls focus on shaping the overall semantics of an image to satisfy the optimization objective, whereas at later timesteps (small t), they progressively manipulate fine-grained details.

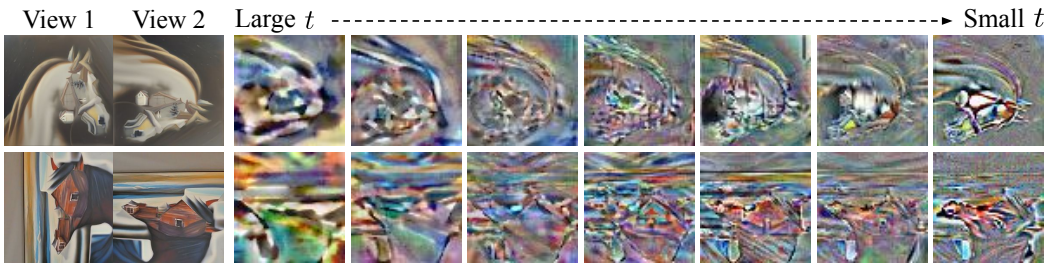


Figure 18: **Visualization of optimized controls.** The controls first shape the overall image structure, then refine fine-grained details. We use the text prompts of "an oil painting of a horse" and "an oil painting of a snowy mountain village", with clockwise rotation.

F SOCIETAL IMPACTS

Our method enables collaborative generation in diverse and challenging scenarios, making it applicable to various visual generation tasks that require globally consistent outputs. It may improve the practicality of generative models in real-world applications such as content creation and design. However, it may also inherit the pretrained generative model's potential limitations, including the generation of harmful or unethical content.