FLOWMORPHER: INVERSION-FREE TEXTURED 3D MORPHING VIA VELOCITY FIELD INTERPOLATION

Anonymous authors
Paper under double-blind review

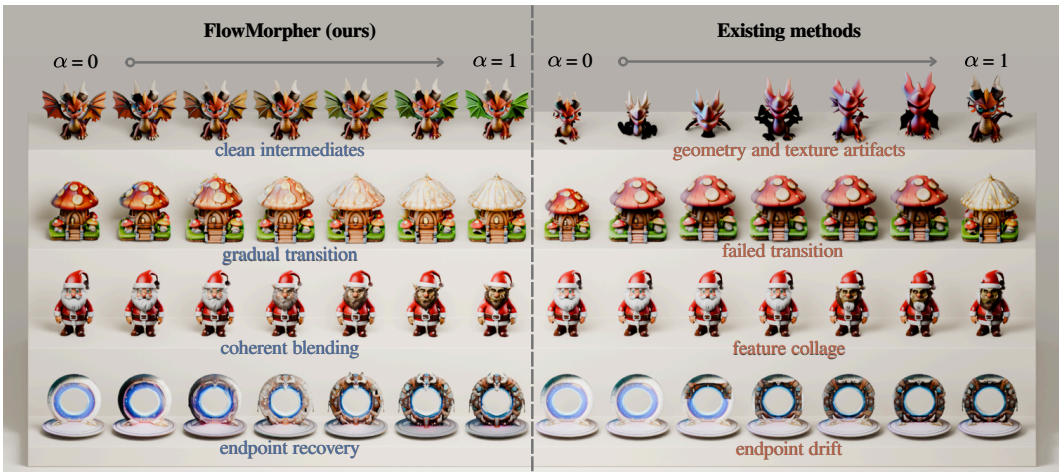

Figure 1: We propose **FlowMorpher**, a training-free and inversion-free method for textured 3D morphing. It interpolates conditional velocity fields of pretrained flow models to construct a family of sampling trajectories without modifying model internals. Here, we present clean intermediate 3D assets with gradual transition, coherent blending, and endpoint recovery.

ABSTRACT

Textured 3D morphing aims to achieve smooth changes in geometry and appearance while recovering both endpoints. However, existing training-free approaches require access to model internals for feature matching and interpolation, and often exhibit abrupt transitions. Other methods require costly caching of intermediate latent states and attention features obtained through 3D inversion. To address these gaps, we introduce **FlowMorpher**, a training-free and inversion-free method that constructs sampling dynamics by interpolating conditional velocity fields of pretrained image-to-3D flow models without modifying their internals. FlowMorpher constructs sampling trajectories directly from shared Gaussian noise, eliminating the need for inversion. In addition, we define the sampling update rule by interpolating the model's conditional velocity predictions, without modifying internal attention operations. Under the stated regularity assumptions, our formulation guarantees Lipschitz continuity of the terminal latent with respect to the morphing weight within a fixed latent space. This analysis provides theoretical support for smooth transitions by establishing latent continuity under the stated assumptions. To evaluate FlowMorpher, we assemble MorphEdit3D from existing datasets and benchmarks, covering color, shape, and semantic changes. Our quantitative results improve over the strongest baseline on every reported metric, reducing KID by 11.2% and PDV by 21.4% while raising target-endpoint DINO-I by 0.039. Qualitative comparisons show intermediate assets with smooth transitions in the generated 3D sequences. These findings show that textured 3D morphing can be controlled through sampling dynamics defined by pretrained velocity predictions.

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.

1 INTRODUCTION

Textured 3D morphing aims to connect a source object and a target object through a sequence of intermediate 3D assets (Yang et al., 2025; Liu et al., 2026). Each intermediate asset should exhibit plausible geometry and appearance, while the sequence should evolve smoothly without visual artifacts or abrupt changes. Closely related to correspondence estimation and shape registration (Yu et al., 2018; Sun et al., 2024; Anguelov et al., 2005), this task encompasses both transitions between different states of the same object and semantic transformations between distinct objects (Zhang et al., 2024a; Cao et al., 2025). The ability to generate such continuous variations supports a range of applications (Ren et al., 2024; Miao et al., 2025), including animation, morphological evolution (Stevens et al., 2024), dynamic transformation effects (Sun et al., 2024), and visual style transitions (Sun et al., 2026).

Pretrained generative models have advanced image morphing by providing priors learned from large datasets, enabling plausible transitions in structure and appearance (Zhang et al., 2024a; Cao et al., 2025). Although these ideas have been extended to 3D morphing (Sun et al., 2024; Yang et al., 2025), independently lifting morphed images into 3D or morphing individual views can introduce severe artifacts, as illustrated in Figures 1 and 2. Drawing on 3D editing methods (Li et al., 2026; Ye et al., 2026), existing 3D morphing approaches perform morphing directly within pretrained 3D generators (Yin et al., 2026; Liu et al., 2026; Sun et al., 2026; Yang et al., 2025). However, the inversion-based methods (Li et al., 2026; Sun et al., 2026) require costly caching of intermediate latent states and attention KV features for subsequent sampling and rely on a specific model. In addition, other methods (Liu et al., 2026; Yang et al., 2025) that control morphing transitions by establishing correspondences between internal features often suffer from failed transitions or visible abrupt changes. Recent methods (Kulikov et al., 2025; Ye et al., 2026) enable inversion-free, training-free editing with pretrained flow models, providing a promising way to connect source and target. However, naively adapting these editing dynamics yields unnatural feature collages and fails to recover the intended target endpoint.

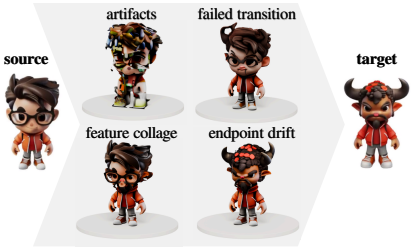

Figure 2: Failure modes of 3D inversion, modification of the model internals, and morphing by editing.

In this work, we present **FlowMorpher**, a training-free and inversion-free method for textured 3D morphing using pretrained image-to-3D flow models without modifying their internals. Motivated by the above failure modes, we formulate textured 3D morphing through a family of sampling trajectories defined by interpolated conditional velocity fields. (1) *Initialization*: FlowMorpher starts every trajectory from the same Gaussian noise used to generate the endpoints, eliminating both inversion and feature caching and making it easy to apply to other rectified flow models. (2) *Update rule*: we define the update rule on model outputs by linearly interpolating the two conditional velocity predictions at each step, without modifying internal attention operations. (3) *Endpoint symmetry*: FlowMorpher treats the two endpoints symmetrically: both conditional velocities are evaluated at the same evolving state, so the interpolation sets the generation direction at every step. Under matched sampling settings, $\alpha = 0, 1$ reduce exactly to the source and target, recovering both endpoints. *Moreover*, under the stated regularity assumptions, our formulation guarantees Lipschitz continuity of the terminal latent with respect to the morphing weight within a fixed latent space. This analysis provides theoretical support for smooth transitions by establishing latent continuity under the stated assumptions. Our quantitative and qualitative results complement this analysis with empirical evidence of smooth transitions in the generated 3D sequences. *Then*, we instantiate FlowMorpher on TRELLIS (Xiang et al., 2025), applying the same interpolation rule to its two stages and corresponding rectified flow models, including sparse structure (SS) and structured latent (SLAT). Within each stage, both conditional velocity fields are evaluated at the same evolving state and time. The SS initial noise is shared across α. SLAT initialization uses a fixed noise field indexed by voxel coordinates, so overlapping supports receive identical noise at shared coordinates.

We curated MorphEdit3D from existing datasets and benchmarks (Liu et al., 2026; Li et al., 2026; Deitke et al., 2023) and evaluated FlowMorpher through quantitative and qualitative comparisons of generation quality, transition smoothness, and endpoint recovery. MorphEdit3D covers color, shape, and semantic changes and a manually refined source-side 3D mask for each pair. These masks

support training-free editing baselines that require region annotations, enabling comparisons with morphing methods on the same source–target pairs. On MorphEdit3D, FlowMorpher achieves the best scores across all reported metrics, reducing KID by 11.2% and PDV by 21.4%, and improving target-endpoint DINO-I from 0.658 to 0.697. Qualitative comparisons illustrate plausible intermediate assets with smooth transitions in geometry and appearance, where baselines produce unnatural feature collages or abrupt jumps. An ablation applying velocity field interpolation to the SS and SLAT stages in isolation shows that each stage improves the aspect it governs in the pretrained generator, motivating its application to both (coarse structure for SS; fine geometry and appearance for SLAT). Our main contributions can be summarized as follows:

- We introduce FlowMorpher, a training-free and inversion-free method that constructs sampling trajectories by interpolating conditional velocity fields of pretrained flow models. The terminal outputs form a textured 3D morphing sequence, while model internals remain unchanged.
- We analyze the initialization and update rule in training-free methods. Under standard regularity assumptions, we establish continuity of FlowMorpher's terminal latents with respect to the morphing weight within a fixed latent space, providing the theoretical basis for continuous transitions.
- We assemble MorphEdit3D from existing datasets and benchmarks in 3D morphing and editing, covering color, shape, and semantic changes. Manually refined source-side 3D masks support comparisons between morphing methods and editing baselines adapted for morphing on common source-target pairs. Experiments on MorphEdit3D show improvements over the evaluated baselines in gradual transition, coherent blending, and endpoint recovery.

2 Related Work

3D Morphing. Traditional 3D morphing constructs smooth geometric deformations between source and target shapes (Yan et al., 2007), often relying on shape correspondences. Related shape matching techniques estimate these correspondences using geometric descriptors and functional maps (Ovsjanikov et al., 2012). Subsequently, learning-based methods jointly learn shape interpolation and correspondence (Eisenberger et al., 2021), although reliable matching across semantically distant shapes remains challenging. Generative priors offer another route to synthesizing plausible intermediate states. In 2D, DiffMorpher (Zhang et al., 2024a) and FreeMorph (Cao et al., 2025) exploit pretrained diffusion models through latent interpolation and attention manipulation. SRIF (Sun et al., 2024) connects image morphing to 3D registration by reconstructing intermediate point clouds from morphed views and using them to guide continuous shape deformation. Recent methods directly exploit pretrained 3D generative priors to synthesize transitions (Liu et al., 2026; Sun et al., 2026; Yin et al., 2026; Yang et al., 2025). Interp3D (Liu et al., 2026) progressively aligns and fuses source and target information at the condition, structure, and texture levels, while other work combines source and target features through costly 3D inversion (Sun et al., 2026). However, interpolating attention keys and values disrupts the pretrained generative dynamics, leading to abrupt geometric changes and degraded generation quality. Therefore, FlowMorpher performs inversion-free 3D morphing by interpolating velocity predictions. Its core interpolation rule operates on model outputs without modifying internal attention mechanisms.

3D Generative Models. Recent advances in generative modeling have improved the fidelity and efficiency of 3D asset creation (Li et al., 2025; Xiang et al., 2025; 2026; Ma et al., 2024; Zhang et al., 2024b; Tang et al., 2024; Xu et al., 2024). Early diffusion-based approaches optimize 3D representations using pretrained 2D image priors (Ho et al., 2020; Poole et al., 2022; Lin et al., 2023; Peebles & Xie, 2023), but require costly per-asset optimization and can suffer from multi-view inconsistencies (Shi et al., 2024). Subsequent pipelines combine multi-view diffusion with feed-forward reconstruction to accelerate 3D generation (Xu et al., 2024; Tang et al., 2024; Zhao et al., 2023). Other work learns generative models directly in 3D latent spaces (Zhang et al., 2024b; Wu et al., 2024), with recent rectified flow models enabling high-fidelity 3D synthesis (Li et al., 2025; Xiang et al., 2025; 2026). In particular, these flow models employ rectified flow transformers to generate structured latents that encode geometry and appearance and can be decoded into multiple 3D representations (Xiang et al., 2025). Building on these 3D generative priors, we investigate velocity field interpolation during inference for training-free 3D morphing.

3D Editing. Early 3D editing methods rely on costly score distillation sampling (Poole et al., 2022) to optimize 3D representations (Chen et al., 2024b; Sella et al., 2023; Zhuang et al., 2024). Other approaches reconstruct 3D assets from edited multi-view images but can suffer from inconsistencies across views (Chen et al., 2024a; Barda et al., 2025; Erkoç et al., 2025). Recently, VoxHammer (Li et al., 2026) performs editing in native 3D latent space through inversion, providing insights for morphing via 3D inversion (Sun et al., 2026). Nano3D (Ye et al., 2026) adapts the inversion-free editing formulation of FlowEdit (Kulikov et al., 2025) to 3D, constructing a direct trajectory from the source under the target condition. These methods focus on modifying a source asset while preserving unrelated content, whereas morphing requires a sequence of transitions connecting two endpoints.

3 PRELIMINARIES

Rectified Flow models. Flow matching learns a time-dependent velocity field to transport samples from a noise distribution toward a data distribution (Lipman et al., 2022). Let X_1 and X_0 denote random vectors drawn from the noise and data distributions, respectively. The learned velocity field defines an ordinary differential equation (ODE) over $t \in [0, 1]$, $\mathrm{d}Z_t = v_\theta(Z_t, t)\,\mathrm{d}t$, where v_θ is a time-dependent velocity field parameterized by a neural network with parameters θ. Here, $t = 1$ corresponds to noise and $t = 0$ to data. The goal is for Z_0 to match the distribution of X_0 when the ODE is initialized with $Z_1 = X_1$. A common choice is $X_1 \sim \mathcal{N}(0, I)$, which is easy to sample from. To generate a sample, we initialize the ODE at $t = 1$ with a standard Gaussian sample and numerically integrate it backward in time. The resulting Z_0 is a sample from the model distribution, which approximates the data distribution. This ODE formulation defines a deterministic transport from the noise distribution to the model distribution.

However, curved sampling trajectories in flow models can require many velocity field evaluations for accurate numerical integration (Pooladian et al., 2023; Liu et al., 2022). Rectified flow aims to reduce this cost by encouraging straight sampling trajectories (Albergo & Vanden-Eijnden, 2022; Lipman et al., 2022; Liu et al., 2022). It can be formulated as flow matching with linear interpolation between paired data and noise samples. Specifically, for the interpolated state $(1 - t)X_0 + tX_1$, the conditional velocity target is the endpoint difference $X_1 - X_0$. With the exact marginal velocity field and under suitable regularity conditions, the ODE solution satisfies $Z_t \overset{d}{=} (1 - t)X_0 + tX_1$, where $\overset{d}{=}$ denotes equality in distribution. Although their individual trajectories may differ, the ODE solution and the linear interpolation have the same marginal distribution at each time t.

TRELLIS structure. Rectified flow models are also used for 3D generation. In this work, we use TRELLIS (Xiang et al., 2025), a pretrained 3D generation model, as our 3D generative prior. TRELLIS generates 3D assets through a two-stage pipeline. The first stage generates the sparse structure (SS) through latent-space sampling and decoding. This structure consists of the active voxel positions $\{\boldsymbol{p}_i\}_{i=1}^{L}$, where L is the number of active voxels. Each position satisfies $\boldsymbol{p}_i \in \{0, 1, \ldots, N-1\}^3$, where N is the number of voxels along each axis of the 3D grid. Given this structure, the second stage generates local latents $\{\boldsymbol{z}_i\}_{i=1}^{L}$. Each $\boldsymbol{z}_i \in \mathbb{R}^C$ is a C-dimensional latent vector associated with the voxel at $\boldsymbol{p}_i$. The positions and local latents together form the structured latent representation (SLAT), $\boldsymbol{z} = \{(\boldsymbol{z}_i, \boldsymbol{p}_i)\}_{i=1}^{L}$. A representation-specific decoder $\boldsymbol{\mathcal{D}}$ then converts $\boldsymbol{z}$ into a 3D asset $\mathcal{O}$. Both generation stages use rectified flow transformers to predict velocity fields.

4 REVISITING INVERSION-FREE EDITING FOR MORPHING

Before presenting FlowMorpher, we examine inversion-free editing (Kulikov et al., 2025; Ye et al., 2026) in the model's continuous latent space. A naive adaptation of such editing methods to morphing retains source-anchored initialization and reference states. This adaptation does not guarantee target endpoint recovery, and its intermediate results can exhibit unnatural combinations of source and target features. These limitations motivate FlowMorpher, which constructs a family of sampling trajectories from a shared initial condition by interpolating conditional velocity fields.

Throughout this section, X^{src} and X^{tar} denote the continuous latent representations of the two given endpoints, representing source and target, respectively. Morphing aims to connect them through a sequence of latent outputs $\{X^\alpha\}_{\alpha \in [0,1]}$, where α denotes progress through the morph and is distinct from the ODE integration time t. The resulting path should preserve both endpoints, with

$X^{\alpha=0} = X^{\mathrm{src}}$ and $X^{\alpha=1} = X^{\mathrm{tar}}$. The corresponding 3D outputs should also change smoothly as α varies. To analyze how a pretrained model can be used for this task, we focus on two choices: the *initialization* and the *update rule*. The initial condition specifies the latent state from which integration starts for each α, while the update rule specifies how this state evolves over t.

Inversion-free editing. Editing is closely related to morphing but imposes a different objective. Given a source sample and a target condition, editing aims to satisfy the target condition while modifying the source content as little as possible. It updates over the full integration interval from $t = 1$ to $t = 0$, as shown in Figure 3 (a). The editing state is initialized as $Z_1^{\Delta} = X^{\mathrm{src}}$. At each integration step t, we draw $\epsilon_t \sim \mathcal{N}(0, I)$ independently of the noise at other steps and construct a noisy source query, $\hat{Z}_t^{\mathrm{src}} = (1-t)X^{\mathrm{src}} + t\epsilon_t$. The paired target query applies the same perturbation, $\hat{Z}_t^{\mathrm{src}} - X^{\mathrm{src}}$, to the current editing state: $\hat{Z}_t^{\mathrm{tar}} = Z_t^{\Delta} + \hat{Z}_t^{\mathrm{src}} - X^{\mathrm{src}}$. These queries also satisfy $\hat{Z}_t^{\mathrm{tar}} - \hat{Z}_t^{\mathrm{src}} = Z_t^{\Delta} - X^{\mathrm{src}}$, where their relative displacement matches the displacement of the current editing state from the source. The pretrained model evaluates the conditioned velocities of source and target at these paired queries. Their difference gives the sampled editing velocity,

$$v_t^{\Delta}\left(\hat{Z}_t^{\mathrm{src}}, \hat{Z}_t^{\mathrm{tar}}\right) = v^{\mathrm{tar}}\left(\hat{Z}_t^{\mathrm{tar}}, t\right) - v^{\mathrm{src}}\left(\hat{Z}_t^{\mathrm{src}}, t\right). \tag{1}$$

Here, $v^{\mathrm{src}}(\cdot, t) \triangleq v_\theta(\cdot, t, c^{\mathrm{src}})$ and $v^{\mathrm{tar}}(\cdot, t) \triangleq v_\theta(\cdot, t, c^{\mathrm{tar}})$ denote the pretrained velocity field conditioned on the source and target conditions c^{src} and c^{tar}, respectively. As t decreases from 1 to 0, the editing state is updated using the sampled velocity difference, which estimates the expected editing velocity in the ODE and can be averaged over multiple noise draws at each step. The two queries share the same noise sample. The difference between the corresponding velocity predictions updates the editing state Z_t^{Δ}, which is maintained separately from the noisy queries.

Morphing by editing. This editing formulation provides a starting point for constructing a candidate morphing sequence by varying the initial condition and the update rule. This naive method varies the editing strength to connect zero editing to the full editing procedure, as shown in Figure 3 (b). For each α, we retain the source initialization $Z_1^{\alpha} = X^{\mathrm{src}}$ and construct the paired target query as $\hat{Z}_t^{\mathrm{tar},\alpha} = Z_t^{\alpha} + \hat{Z}_t^{\mathrm{src}} - X^{\mathrm{src}}$. We reuse the same integration schedule and noise draws across all α values, keeping sampling randomness fixed when comparing their outputs. Based on Eq. 1, we scale the sampled editing velocity by α:

$$v_t^{\alpha} = \alpha v_t^{\Delta}\left(\hat{Z}_t^{\mathrm{src}}, \hat{Z}_t^{\mathrm{tar},\alpha}\right). \tag{2}$$

We integrate from $t = 1$ to $t = 0$ using this update velocity and take the terminal latent $X^{\alpha} = Z_0^{\alpha}$ as the output of morphing by editing. This method requires no preliminary inversion from data to noise. All trajectories start from the source and use it throughout integration. Although α scales each velocity difference, its effect on the final output is not generally linear because the target query also depends on the evolving state Z_t^{α}. Thus, α controls the strength of the editing update without directly prescribing progress between the two given endpoints.

Failure modes. Morphing by editing does not guarantee recovery of the given target sample X^{tar}. In Figure 3 (b), the generated endpoint target asset differs in appearance from the target image. At $\alpha = 0$, the update velocity vanishes, so the state remains at X^{src} throughout integration, yielding $X^{\alpha=0} = X^{\mathrm{src}}$. At $\alpha = 1$, this method recovers the editing procedure described above. The source enters both the initial condition and the reference state used at every step, whereas the target enters only through the conditioning of the velocity field v^{tar}.

In addition, it produces unnatural feature collages along the morphing sequence. As shown at the bottom of Figure 3 (b), the intermediate result exhibits an unnatural feature collage that combines facial and hair appearance from the two endpoints. Even if the latent outputs vary continuously with α, this alone does not ensure gradual changes in the geometry and appearance of the resulting outputs. Sharing the same noise sample across all α reduces stochastic variation within the sequence, but does not ensure smooth transitions. These observations motivate revisiting the initialization and update rule to incorporate both endpoints into the morphing. The resulting sequence should be assessed for both smooth transition and endpoint recovery of the target.

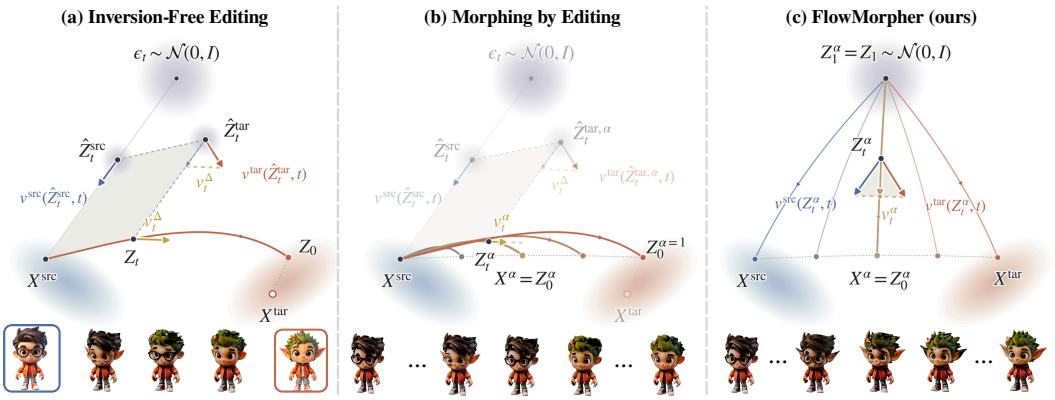

Figure 3: **Morphing by editing vs. morphing from shared noise (FlowMorpher). (a)** Inversion-free editing evolves a noise-free state Z_t, initialized at X^{src}, with the velocity difference v_t^{Δ} evaluated at the paired noisy queries $\hat{Z}_t^{\text{src}}$ and $\hat{Z}_t^{\text{tar}}$ (gray parallelogram). The intermediate states of the editing trajectory are themselves 3D assets, so the trajectory has the potential to form a morph sequence. **(b)** Morphing by editing starts every trajectory at X^{src} and uses the scaled editing update αv_t^{Δ}. However, the intermediates show feature collage, and the endpoint drifts away ($Z_0^{\alpha=1} \neq X^{\text{tar}}$). Faded elements repeat (a). **(c)** FlowMorpher interpolates the generative dynamics, without inversion or changes to the network. For each α, it starts from shared noise Z_1 and follows $v_t^{\alpha} = (1-\alpha)\, v^{\text{src}}(Z_t^{\alpha}, t) + \alpha\, v^{\text{tar}}(Z_t^{\alpha}, t)$, so the interpolation sets the generation direction at every step. This yields a complete generation trajectory that reduces exactly to the source or target generation at $\alpha \in \{0, 1\}$, and the outputs $X^{\alpha} = Z_0^{\alpha}$ form the morphing sequence.

5 FlowMorpher

FlowMorpher realizes textured 3D morphing by interpolating conditional velocity fields to construct a family of sampling trajectories. This formulation enables smooth transitions and endpoint recovery without inversion or modifications to model internals. We first introduce our key design, velocity field interpolation (VFI), in Section 5.1. We then articulate the pipeline in Section 5.2.

5.1 Velocity Field Interpolation for Morphing

Given source and target images, either provided directly or rendered from input 3D assets, we sample from the pretrained model under each condition using the same initial Gaussian noise and identical sampling settings. For each α, FlowMorpher starts from this shared noise and integrates a weighted combination of both conditional velocities evaluated at the same evolving state and time. The terminal outputs form the textured 3D morphing sequence, as shown in Figure 3 (c). With identical sampling settings, FlowMorpher recovers the corresponding generation trajectories and endpoint samples.

Let X^{src} and X^{tar} denote the latent samples generated by the pretrained model under the source and target conditions, respectively, from the same Gaussian noise $Z_1 \sim \mathcal{N}(0, I)$. For each $\alpha \in [0, 1]$, we initialize a separate sampling trajectory using this shared noise $Z_1^{\alpha} = Z_1$, and use the same integration schedule and solver for all trajectories. At each integration time t, we evaluate the source- and target-conditioned velocity fields at the same current latent state Z_t^{α}. We linearly combine their predictions to define the update rule, which promotes smooth transitions:

$$v_t^{\alpha} = (1-\alpha) v^{\text{src}}(Z_t^{\alpha}, t) + \alpha v^{\text{tar}}(Z_t^{\alpha}, t). \tag{3}$$

We integrate from $t = 1$ to $t = 0$ and take the terminal latent $X^{\alpha} = Z_0^{\alpha}$ as the output. The model is queried under each original condition separately, and only the resulting velocities are interpolated. At $\alpha = 0$ and $\alpha = 1$, Eq. 3 reduces to standard generation under the source and target conditions, respectively, recovering $X^{\alpha=0} = X^{\text{src}}$ and $X^{\alpha=1} = X^{\text{tar}}$ given the same initial noise and deterministic solver. For intermediate α, we evaluate both velocity fields at the current state Z_t^{α} at each integration step and use their weighted combination to update the state. These successive

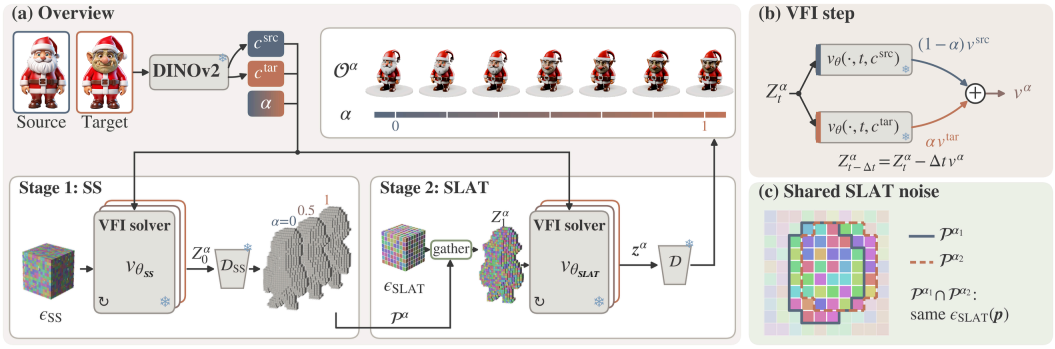

Figure 4: **FlowMorpher on TRELLIS.** For each morph weight α, both the SS and SLAT stages integrate the interpolated velocity with both fields evaluated at the same state (and, for SLAT, the same support $\mathcal{P}^\alpha$). All networks are frozen. The SS noise is shared across α; the SLAT noise is a single field on the voxel grid, so a voxel shared by several supports receives identical initial noise. Each α is solved independently, and $\alpha = 0$ and $\alpha = 1$ reproduce the endpoint generations.

updates produce the terminal latent X^α. Equivalently, the update rule can be written as,

$$v_t^\alpha = v^{\text{src}}(Z_t^\alpha, t) + \alpha \left[v^{\text{tar}}(Z_t^\alpha, t) - v^{\text{src}}(Z_t^\alpha, t) \right]. \tag{4}$$

This form connects interpolation to morphing by editing in Eq. 2 through the α-scaled velocity difference, while including the conditioned velocity and evaluating both fields at the same state.

The construction also applies to classifier-free guidance (CFG) (Ho & Salimans, 2022) by replacing each conditional velocity with its guided counterpart. For affine CFG with the same strength for both conditions, interpolating the guided velocities is algebraically equivalent to interpolating the conditional velocities first and then applying guidance with a shared unconditional prediction. This equivalence also holds for a shared time-dependent guidance schedule. The endpoint identities then refer to the two generations obtained with the same guided fields and sampling settings.

Continuity in α. Consider a fixed pair of conditions and the sampling velocity fields, including CFG when applied. Suppose all trajectories remain in a common compact region, on which these fields are piecewise continuous in t, bounded, and uniformly Lipschitz in the latent state. With shared initial noise, the terminal latents satisfy $\|X^\alpha - X^{\alpha'}\| \le K|\alpha - \alpha'|$ for all $\alpha, \alpha' \in [0, 1]$, where $K = M(e^L - 1)/L$ is a finite constant independent of α and α', determined by the Lipschitz constant L and the supremum M of the difference between the two sampling fields on this region. In particular, the terminal latents vary continuously with α. The derivation separates the velocity difference between two trajectories into a term bounded by L times their state separation and a term bounded by $M|\alpha - \alpha'|$. Shared initialization gives zero initial separation, and Grönwall's inequality, or its discrete analogue for Euler steps, bounds its growth during integration. This result applies within a fixed continuous latent space. More details are provided in Appendix B.

5.2 The FlowMorpher Pipeline

Here, we provide details on applying velocity field interpolation to TRELLIS, and Figure 4 provides an overview. The SS and SLAT stages of TRELLIS each have their own flow model and both stages can apply Eq. 3 using the same morph weight α. In what follows, Z_t^α denotes the sampling state of the current stage. The fields v^{src} and v^{tar} denote the velocity fields for that stage under the source and target conditions, respectively. The initialization and support are specified below.

Sparse-structure generation. In this stage, Z_t^α denotes the continuous SS latent. We sample a Gaussian tensor $\epsilon_{\text{SS}} \sim \mathcal{N}(0, I)$ with the SS latent shape and initialize $Z_1^\alpha = \epsilon_{\text{SS}}$ for every α. The trajectory evolves according to this formulation,

$$v_t^\alpha = (1 - \alpha) v^{\text{src}}(Z_t^\alpha, t) + \alpha v^{\text{tar}}(Z_t^\alpha, t). \tag{5}$$

Table 1: Quantitative comparison on MorphEdit3D. Best result per metric in **bold**. †3D editing method adapted for morphing. KID and PDV values are multiplied by 10^3.

	Generation quality		Transition smoothness		DINO-I ↑	
Method	FID ↓	KID ↓	$\text{LPIPS}_{\text{adj}}$ ↓	PDV ↓	α=0	α=1
FreeMorph	129.69	85.81	0.222	6.41	0.600	0.630
VoxHammer†	63.38	28.34	0.110	7.27	0.686	0.653
DiffMorpher	59.80	24.87	0.156	5.42	0.607	0.635
Interp3D	41.87	14.35	0.114	3.51	0.696	0.658
Nano3D†	36.27	10.19	0.099	1.40	0.693	0.657
FlowMorpher (ours)	**35.00**	**9.05**	**0.096**	**1.10**	**0.703**	**0.697**

After integration from $t = 1$ to $t = 0$, we decode the terminal latent using the SS decoder $\boldsymbol{\mathcal{D}}_{\text{SS}}$ and obtain the support $\mathcal{P}^\alpha = \text{Active}(\boldsymbol{\mathcal{D}}_{\text{SS}}(Z_0^\alpha))$, where Active applies the backbone's occupancy threshold and extracts the active voxel coordinates. We write this support as $\mathcal{P}^\alpha = \{\boldsymbol{p}_i^\alpha\}_{i=1}^{L^\alpha}$, with $L^\alpha = |\mathcal{P}^\alpha|$ and $\boldsymbol{p}_i^\alpha \in \{0, 1, \ldots, N-1\}^3$. It specifies the coarse spatial structure for the SLAT stage.

Structured-latent generation. For SLAT, $Z_t^\alpha \in \mathbb{R}^{L^\alpha \times C}$ contains the feature vectors at the positions in $\mathcal{P}^\alpha$. Independently of the SS noise, we sample a Gaussian noise field once on the voxel grid, with $\epsilon_{\text{SLAT}}(\boldsymbol{p}) \overset{\text{iid}}{\sim} \mathcal{N}(0, I_C)$ across voxel coordinates. For each α, we initialize $[Z_1^\alpha]_i = \epsilon_{\text{SLAT}}(\boldsymbol{p}_i^\alpha)$, so a voxel present in multiple supports receives the same initial noise, as shown in Figure 4 (c). We hold $\mathcal{P}^\alpha$ fixed during each SLAT solve and evaluate both velocity fields at the same feature state,

$$v_t^\alpha = (1-\alpha)\, v^{\text{src}}(Z_t^\alpha, t; \mathcal{P}^\alpha) + \alpha\, v^{\text{tar}}(Z_t^\alpha, t; \mathcal{P}^\alpha). \tag{6}$$

TRELLIS's standard latent denormalization maps the terminal state Z_0^α to the local latents $\{\boldsymbol{z}_i^\alpha\}_{i=1}^{L^\alpha}$, which encode local geometry and appearance. The resulting structured latent is $\boldsymbol{z}^\alpha = \{(\boldsymbol{z}_i^\alpha, \boldsymbol{p}_i^\alpha)\}_{i=1}^{L^\alpha}$, and the decoder produces the final asset $\mathcal{O}^\alpha = \boldsymbol{\mathcal{D}}(\boldsymbol{z}^\alpha)$.

6 Experiments

Quantitative Evaluation. As shown in Table 1, FlowMorpher achieve the best scores across all reported metrics on MorphEdit3D. For generation quality, it obtains an FID of 35.00 and a KID of 9.05, corresponding to reductions of 3.5% and 11.2% relative to Nano3D, respectively. For transition smoothness, FlowMorpher reduces $\text{LPIPS}_{\text{adj}}$ by 3.0% and PDV by 21.4% relative to Nano3D, indicating smaller and more uniform perceptual changes between adjacent states. For endpoint recovery, FlowMorpher achieves DINO-I scores of 0.703 and 0.697 at $\alpha = 0$ and $\alpha = 1$, respectively, compared with 0.696 and 0.658 for Interp3D, the strongest baseline at both endpoints. The improvement is larger at the target endpoint (+0.039) than at the source endpoint (+0.007).

Qualitative Comparison. Figure 5 presents qualitative comparisons involving a semantic transition of the portal. FreeMorph (Cao et al., 2025) and DiffMorpher (Zhang et al., 2024a) exhibit pronounced inconsistencies and artifacts in structure and texture. VoxHammer (Li et al., 2026) produces recognizable assets but exhibits failed transitions. Nano3D (Ye et al., 2026) yields feature collages, resulting in visibly fragmented combinations of source and target structures. This behavior is consistent with the source-dominated transitions discussed in our analysis of morphing by editing. Interp3D (Liu et al., 2026) generates plausible intermediate shapes, although abrupt changes remain visible between some adjacent states. In contrast, FlowMorpher produces a more gradual transition and preserves finer structure than Interp3D, such as the thin, high-frequency geometric details at the top of the portal frame. These comparisons illustrate its ability to accommodate both substantial semantic transformations and localized geometric changes.

Ablation Study. We apply FlowMorpher to SS and SLAT separately (Table 2). TRELLIS factorizes generation into two rectified flow stages: the SS stage determines the coarse occupancy of the asset, and the SLAT stage synthesizes fine-grained geometry and appearance on top of this structure.

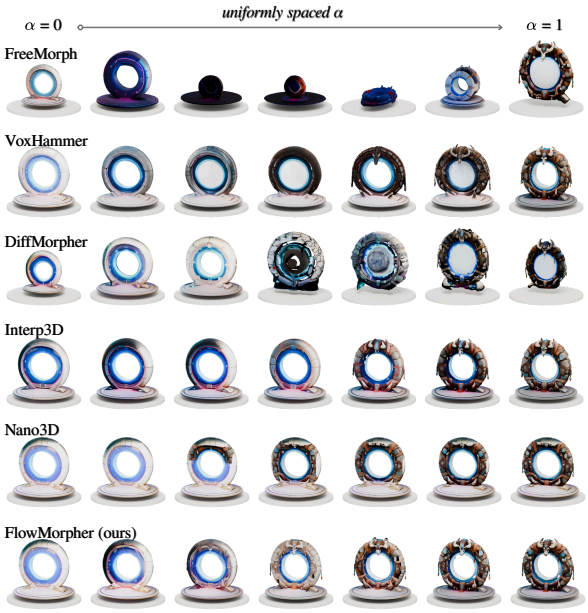

Figure 5: Qualitative comparison. FlowMorpher achieves gradual and coherent transition.

Table 2: Ablation on the TRELLIS stage at which FlowMorpher is applied. We report mean over all pairs and the 45 pairs of each level. Bold marks the better variant within each level.

Level	Variant	FID ↓	KID ↓	$\text{LPIPS}_{\text{adj}}$ ↓	PDV ↓
Overall	SS-only	**35.66**	**9.24**	**0.097**	**1.14**
	SLAT-only	35.91	9.31	0.100	1.41
Color	SS-only	**33.39**	**8.92**	**0.089**	**0.97**
	SLAT-only	33.48	9.29	**0.089**	1.13
Shape	SS-only	**34.38**	**8.57**	**0.092**	**1.06**
	SLAT-only	35.85	9.61	0.100	1.44
Semantic	SS-only	39.21	10.22	**0.109**	**1.38**
	SLAT-only	**38.41**	**9.04**	0.110	1.65

Compared with SLAT-only, SS-only yields lower PDV and no higher $\text{LPIPS}_{\text{adj}}$ spanning different levels, and better scores on all four metrics for shape changes. SLAT-only yields lower FID and KID for semantic changes. Therefore, each single-stage variant improves the aspect for which its stage is responsible in the pretrained generator. These results suggest complementary effects of velocity field interpolation across the two stages, motivating its application to both in FlowMorpher.

7 Conclusion

We present FlowMorpher, an inversion-free method for textured 3D morphing via velocity field interpolation. Building on an analysis of training-free methods, it combines velocities predicted under the source and target conditions at the same evolving state without modifying model internals. Under standard regularity assumptions, we establish continuity of the terminal latents with respect to α within a fixed latent space. We instantiate FlowMorpher in TRELLIS's SS and SLAT stages and introduce MorphEdit3D, a dataset spanning different levels with annotated 3D masks. Experiments demonstrate improvements over baselines in generation quality, transition smoothness, and endpoint recovery. Qualitative comparisons further show coherent changes in geometry and appearance. Our findings provide a simple basis for textured 3D morphing through sampling dynamics.

AI USE STATEMENT

In this work, we used generative AI tools (large language models, e.g., Claude and ChatGPT) only to aid and polish the writing of the paper. This included correcting grammar and spelling, improving wording and readability, checking the consistency of terminology and notation, and suggesting alternative phrasings for text drafted by the authors.

REFERENCES

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM Siggraph 2005 Papers*, pp. 408–416. 2005.

Amir Barda, Matheus Gadelha, Vladimir G Kim, Noam Aigerman, Amit H Bermano, and Thibault Groueix. Instant3dit: Multiview inpainting for fast editing of 3d objects. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16273–16282. IEEE, 2025.

Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

Yukang Cao, Chenyang Si, Jinghao Wang, and Ziwei Liu. Freemorph: Tuning-free generalized image morphing with diffusion model. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18111–18120. IEEE, 2025.

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

Minghao Chen, Junyu Xie, Iro Laina, and Andrea Vedaldi. Shap-editor: Instruction-guided latent 3d editing in seconds. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26446–26456. IEEE, 2024b.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36:35799–35813, 2023.

Marvin Eisenberger, David Novotny, Gael Kerchenbaum, Patrick Labatut, Natalia Neverova, Daniel Cremers, and Andrea Vedaldi. Neuromorph: Unsupervised shape interpolation and correspondence in one go. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7469–7479. IEEE, 2021.

Ziya Erkoç, Can Gümeli, Chaoyang Wang, Matthias Nießner, Angela Dai, Peter Wonka, Hsin-Ying Lee, and Peiye Zhuang. Preditor3d: Fast and precise 3d shape editing. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 640–649. IEEE, 2025.

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. *Advances in neural information processing systems*, 30, 2017.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19721–19730. IEEE, 2025.

Lin Li, Zehuan Huang, Haoran Feng, Gengxiong Zhuang, Rui Chen, Chunchao Guo, and Lu Sheng. Voxhammer: Training-free precise and coherent 3d editing in native 3d space. In *2026 International Conference on 3D Vision (3DV)*, pp. 1281–1292. IEEE, 2026.

Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, et al. Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 300–309. IEEE, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xiaolu Liu, Yicong Li, Qiyuan He, Jiayin Zhu, Wei Ji, Angela Yao, and Jianke Zhu. Interp3d: Correspondence-aware interpolation for generative textured 3d morphing. *arXiv preprint arXiv:2601.14103*, 2026.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024.

Qiaowei Miao, Kehan Li, Jinsheng Quan, Zhiyuan Min, Shaojie Ma, Yichao Xu, Yi Yang, Ping Liu, and Yawei Luo. Advances in 4d generation: A survey. *arXiv preprint arXiv:2503.14501*, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (ToG)*, 31(4):1–11, 2012.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky TQ Chen. Multisample flow matching: Straightening flows with minibatch couplings. *arXiv preprint arXiv:2304.14772*, 2023.

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

Xuebin Qin, Hang Dai, Xiaobin Hu, Deng-Ping Fan, Ling Shao, and Luc Van Gool. Highly accurate dichotomous image segmentation. In *European conference on computer vision*, pp. 38–56. Springer, 2022.

Jiawei Ren, Kevin Xie, Ashkan Mirzaei, Hanxue Liang, Xiaohui Zeng, Karsten Kreis, Ziwei Liu, Antonio Torralba, Sanja Fidler, Seung W Kim, et al. L4gm: Large 4d gaussian reconstruction model. *Advances in Neural Information Processing Systems*, 37:56828–56858, 2024.

Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 430–440. IEEE, 2023.

Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. In *International conference on learning representations*, volume 2024, pp. 39838–39859, 2024.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Samuel Stevens, Jiaman Wu, Matthew J Thompson, Elizabeth G Campolongo, Chan Hee Song, David Edward Carlyn, Li Dong, Wasila M Dahdul, Charles Stewart, Tanya Berger-Wolf, et al. Bioclip: A vision foundation model for the tree of life. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19412–19424. IEEE, 2024.

Mingze Sun, Chen Guo, Puhua Jiang, Shiwei Mao, Yurun Chen, and Ruqi Huang. Srif: Semantic shape registration empowered by diffusion-based image morphing and flow estimation. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

Xiaokun Sun, Zeyu Cai, Hao Tang, Ying Tai, Jian Yang, and Zhenyu Zhang. Morphany3d: Unleashing the power of structured latent in 3d morphing. *arXiv preprint arXiv:2601.00204*, 2026.

Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *European Conference on Computer Vision*, pp. 1–18. Springer, 2024.

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*, 37:121859–121881, 2024.

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

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14419–14429, 2026.

Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

Han-Bing Yan, Shi-Min Hu, and Ralph R Martin. 3d morphing using strain field interpolation. *Journal of Computer Science and Technology*, 22(1):147–155, 2007.

Songlin Yang, Yushi Lan, Honghua Chen, and Xingang Pan. Textured 3d regenerative morphing with 3d diffusion prior. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15159–15170. IEEE, 2025.

Junliang Ye, Shenghao Xie, Ruowen Zhao, Zhengyi Wang, Hongyu Yan, Wenqiang Zu, Lei Ma, and Jun Zhu. Nano3d: A training-free approach for efficient 3d editing without masks. In *International Conference on Learning Representations*, volume 2026, pp. 46900–46922, 2026.

Minghao Yin, Yukang Cao, and Kai Han. Wukong's 72 transformations: High-fidelity textured 3d morphing via flow models. *Advances in Neural Information Processing Systems*, 38:16121–16149, 2026.

Haiyang Ying, Yixuan Yin, Jinzhi Zhang, Fan Wang, Tao Yu, Ruqi Huang, and Lu Fang. Omniseg3d: Omniversal 3d segmentation via hierarchical contrastive learning. *arXiv preprint arXiv:2311.11666*, 2023.

Tao Yu, Zerong Zheng, Kaiwen Guo, Jianhui Zhao, Qionghai Dai, Hao Li, Gerard Pons-Moll, and Yebin Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7287–7296, 2018.

Kaiwen Zhang, Yifan Zhou, Xudong Xu, Bo Dai, and Xingang Pan. Diffmorpher: Unleashing the capability of diffusion models for image morphing. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7912–7921. IEEE, 2024a.

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, 2024b.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. *Advances in neural information processing systems*, 36:73969–73982, 2023.

Jingyu Zhuang, Di Kang, Yan-Pei Cao, Guanbin Li, Liang Lin, and Ying Shan. Tip-editor: An accurate 3d editor following both text-prompts and image-prompts. *ACM Transactions on Graphics (ToG)*, 43(4):1–12, 2024.

A Implementation and Experimental Details

A.1 FlowMorpher: Algorithm and Implementation

We use the publicly available image-large model of TRELLIS (Xiang et al., 2025) as our pretrained 3D generative prior and implement FlowMorpher without additional training. Both the SS and SLAT stages use 25 Euler integration steps with a timestep rescaling factor of 3.0. The voxel grid has resolution N^3 with $N = 64$, and the SLAT feature dimension is $C = 8$.

At each integration step, we linearly interpolate the two conditional velocity predictions evaluated at the same evolving latent state and time, following Section 5.2. The morphing weight α remains fixed throughout each generation. SLAT generation uses the sparse coordinates produced by the SS stage. Algorithm 1 summarizes this two-stage velocity field interpolation pipeline. The operators $\mathrm{VFI}_{\mathrm{SS}}$ and $\mathrm{VFI}_{\mathrm{SLAT}}$ integrate from the supplied initialization using the corresponding stage's sampling settings. Throughout the algorithm, Z_t^α denotes the latent state of the stage currently being processed. The SLAT call additionally takes $\mathcal{P}^\alpha$ as its support, and Denorm denotes the backbone's latent denormalization.

Following TRELLIS's guidance convention, we set the CFG strength to 5.0 for both conditions in both stages, with guidance active over $t \in [0.5, 1.0]$. For each image pair, we generate seven outputs at uniformly spaced values of α from 0 to 1, including both endpoints. We reset the random seed to 1 before each generation. The endpoints are generated using standard TRELLIS image-to-3D sampling with the guidance settings described above. The resulting SLATs are decoded into meshes and 3D Gaussians. We conduct all experiments on a single NVIDIA RTX 4090 GPU.

A.2 Experimental Setup

A.2.1 Dataset Construction

We construct MorphEdit3D, a dataset consisting of 135 source–target pairs grouped into three levels: color, shape, and semantic changes. We collect data from Objaverse-XL (Deitke et al., 2023), example images from the TRELLIS repository[1], the 3D editing dataset Edit3D-Bench (Li et al., 2026), and the 3D morphing dataset Interp3DData (Liu et al., 2026). Each level contains 45 pairs spanning humanoids, creatures, objects, buildings, and vehicles. We additionally provide 3D masks for the source assets, obtained using 3D segmentation (Ying et al., 2023) followed by manual refinement, enabling the evaluation of editing baselines on the same pairs used for morphing. Overall, each pair includes a source image, a target image, and a 3D mask for the source asset. All data, code, and evaluation scripts will be released upon publication, along with licensing information.

[1] `https://github.com/microsoft/TRELLIS`

Algorithm 1 FlowMorpher, instantiated on TRELLIS

Require: Pretrained SS and SLAT flow models; source and target conditions; decoders $\boldsymbol{\mathcal{D}}_{\mathrm{SS}}, \boldsymbol{\mathcal{D}}$; finite set of morph weights $\mathcal{A} \subset [0,1]$ containing 0 and 1; stage sampling settings
Ensure: Morph sequence $\{\mathcal{O}^\alpha\}_{\alpha\in\mathcal{A}}$
Sample $Z_{1,\mathrm{SS}} \sim \mathcal{N}(0, I)$ with the SS latent shape ▷ shared by all α
Independently sample $\epsilon_{\mathrm{SLAT}}(\boldsymbol{p}) \overset{\mathrm{iid}}{\sim} \mathcal{N}(0, I_C)$ for each voxel coordinate $\boldsymbol{p}$
for $\alpha \in \mathcal{A}$ **do**
  $Z^\alpha_{0,\mathrm{SS}} \leftarrow \mathrm{VFI}_{\mathrm{SS}}(Z_{1,\mathrm{SS}}, \alpha)$ ▷ SS stage
  $\mathcal{P}^\alpha \leftarrow \mathrm{Active}(\boldsymbol{\mathcal{D}}_{\mathrm{SS}}(Z^\alpha_{0,\mathrm{SS}}))$ ▷ $\mathcal{P}^\alpha = \{\boldsymbol{p}^\alpha_i\}_{i=1}^{L^\alpha}$
  $Z^\alpha_{1,\mathrm{SLAT}} \leftarrow [\epsilon_{\mathrm{SLAT}}(\boldsymbol{p}^\alpha_i)]_{i=1}^{L^\alpha}$
  $Z^\alpha_{0,\mathrm{SLAT}} \leftarrow \mathrm{VFI}_{\mathrm{SLAT}}(Z^\alpha_{1,\mathrm{SLAT}}, \alpha; \mathcal{P}^\alpha)$ ▷ SLAT stage
  $\{\boldsymbol{z}^\alpha_i\}_{i=1}^{L^\alpha} \leftarrow \mathrm{Denorm}(Z^\alpha_{0,\mathrm{SLAT}})$
  $\boldsymbol{z}^\alpha \leftarrow \{(\boldsymbol{z}^\alpha_i, \boldsymbol{p}^\alpha_i)\}_{i=1}^{L^\alpha}$
  $\mathcal{O}^\alpha \leftarrow \boldsymbol{\mathcal{D}}(\boldsymbol{z}^\alpha)$ ▷ decode asset
end for
return $\{\mathcal{O}^\alpha\}_{\alpha\in\mathcal{A}}$

A.2.2 BASELINES

We select baselines to cover representative technical approaches across dedicated morphing methods and 3D editing methods adapted for morphing: native textured 3D morphing, 3D editing with and without inversion, and image morphing followed by 3D generation. We use their available implementations with the adaptations described below.

Interp3D (Liu et al., 2026) is our native textured 3D morphing baseline. It combines semantically aligned condition interpolation, SLAT-guided structure interpolation, and texture fusion to generate intermediate textured 3D assets, morphing through internal feature correspondence.

Nano3D (Ye et al., 2026) is our inversion-free 3D editing baseline. It integrates FlowEdit into TRELLIS and uses region-aware voxel and SLAT merging to preserve unedited content. We adapt it to morphing following the morphing-by-editing scheme in Section 4.

VoxHammer (Li et al., 2026) is our inversion-based 3D editing baseline. It operates in the 3D latent space and uses 3D masks to guide latent and attention feature replacement in preserved regions. We adapt it for morphing with the morphing cross-attention of MorphAny3D (Sun et al., 2026), retaining its 3D inversion and region-preservation components.

DiffMorpher (Zhang et al., 2024a) and **FreeMorph** (Cao et al., 2025) represent diffusion-based image morphing followed by 3D generation. We generate intermediate images with each method and feed each image into TRELLIS to obtain a sequence of 3D assets.

Two further native 3D morphing methods are not evaluated directly. MorphAny3D (Sun et al., 2026) obtains its initial latents through the 3D inversion of VoxHammer and controls transitions with morphing cross-attention (MCA) and temporal-fused self-attention (TFSA). Because TFSA attends to the attention features of the preceding frame, these features must be cached and carried along the sequence, which raises the memory requirement as the original implementation runs on an NVIDIA RTX A6000. Our adapted VoxHammer† baseline combines 3D inversion with both attention mechanisms, MCA and TFSA. It covers MorphAny3D at the level of its attention mechanisms while running on a single RTX 4090, the same GPU used for all other methods, so that every method is compared under the same hardware budget. In addition, Wukong's 72 Transformations (Yin et al., 2026) is discussed in Section 2 but excluded from the quantitative comparison, as no public implementation was available at the time of submission. All compared methods share the same pretrained TRELLIS model as their 3D generative prior, so performance differences reflect the morphing methods. Although FlowMorpher itself is backbone-agnostic, we evaluate it quantitatively only on TRELLIS because the dedicated 3D baselines are all built on this backbone.

A.2.3 METRICS

We evaluate generated 3D sequences in terms of three complementary aspects: distributional alignment, transition smoothness, and endpoint recovery. For distributional alignment, we employ Fréchet Inception Distance (FID) (Heusel et al., 2017) and Kernel Inception Distance (KID) (Bińkowski et al., 2018). For each source–target pair, we render the intermediate states at $\alpha = 1/3$ and $2/3$, together with both generated endpoints, from 1,000 shared viewpoints at 512×512 resolution with a white background. FID and KID compare the pooled 2,000 intermediate renders against the 2,000 endpoint renders using 2,048-dimensional Inception features. KID uses a cubic polynomial kernel and is averaged over 100 random subsets of 1,000 images per distribution. For transition smoothness, we compute $\mathrm{LPIPS}_{\mathrm{adj}}$ using AlexNet-based LPIPS (Zhang et al., 2018), averaged over all six adjacent transitions and 64 shared viewpoints. We additionally report perceptual distance variance (PDV) (Zhang et al., 2024a), defined as the sample variance of the six adjacent LPIPS distances within each viewpoint, averaged across views, to quantify the uniformity of perceptual change along the sequence. For endpoint recovery, we report DINO-I at $\alpha = 0$ and $\alpha = 1$, measuring cosine similarity between each endpoint render and its corresponding source or target input image using mean-pooled DINOv2-base token features (Oquab et al., 2023), averaged over the same 64 viewpoints. All metrics are computed per pair and subsequently averaged across pairs.

A.3 APPLICATION TO ADDITIONAL FLOW MODELS

Velocity field interpolation requires only conditional velocity evaluations at a shared latent state and time, including a velocity-parameterized rectified flow sampler and a conditional forward call. It makes no assumptions about the latent structure (dense grid or sparse voxels), the backbone architecture, and transfers to other conditional flow matching models (Lipman et al., 2022) with only noise assignment and decoding adapted to each latent representation.

We further apply FlowMorpher to TripoSG (Li et al., 2025) and TRELLIS.2 (Xiang et al., 2026) without additional training, retaining the same velocity field interpolation rule. For TRELLIS.2, we integrate this rule into sparse structure, shape, and texture generation. For TripoSG, we apply it to shape generation. As shown in Figure 6 and Figure 7, both integrations produce plausible intermediate shapes, with progressive appearance changes also observed in TRELLIS.2. These results provide qualitative evidence that FlowMorpher is applicable across different flow-based 3D generators.

TripoSG. We implement velocity field interpolation in pretrained TripoSG without additional training. Given a source–target image pair, each image is independently foreground-cropped, padded, and composited onto a white background, using its valid alpha mask or a mask estimated by RMBG-1.4 (Qin et al., 2022). The processed images are passed through TripoSG's image processor and DINOv2 encoder to obtain separate conditioning token sequences c^{src} and c^{tar}; the unconditional condition is a zero tensor of the same shape. VFI is applied throughout the single latent-generation stage. At each step k, both conditions are evaluated at the same current latent z_k^α and timestep t_k. We first apply classifier-free guidance separately,

$$\hat{v}_j = v_\theta(Z_k^\alpha, t_k; 0) + g[v_\theta(Z_k^\alpha, t_k; c_j) - v_\theta(Z_k^\alpha, t_k; 0)], \tag{7}$$

for $j \in \{s, t\}$, and then form $v_\alpha = (1-\alpha)\hat{v}_s + \alpha\hat{v}_t$. Following the implementation's custom rectified flow scheduler and velocity convention, the actual Euler update is $Z_{k+1}^\alpha = Z_k^\alpha + (\sigma_k - \sigma_{k+1})v_\alpha$, with $\sigma_k = 1 - k/K$, $t_k = 1000\sigma_k$, and $K = 50$. Each trajectory starts from a Gaussian latent of shape 2048×64. We recreate the device-specific random generator with seed 42 for every α, yielding identical initial noise elementwise across morphing weights; subsequent sampling updates introduce no additional noise. Each weight follows its own trajectory from the shared initialization. The final latent is passed to the pretrained VAE's flash decoding and geometry extraction routine with octree-depth parameter 9; this downstream stage depends on the generated latent and requires no additional flow sampling or VFI. We use CFG strength $g = 7$, FP16 model inference on CUDA, and seven uniformly spaced weights in $[0, 1]$, including both endpoints. Outputs comprise a GLB mesh per weight, copies of the original and preprocessed conditions, and a manifest recording settings and asset paths. For visualization, we render 120-frame orbit videos at 512×512 resolution and 30 fps, together with a side-by-side comparison.

TRELLIS.2. We also implement velocity field interpolation in the pretrained TRELLIS.2-4B pipeline without additional training. Given a source–target image pair, we independently apply the

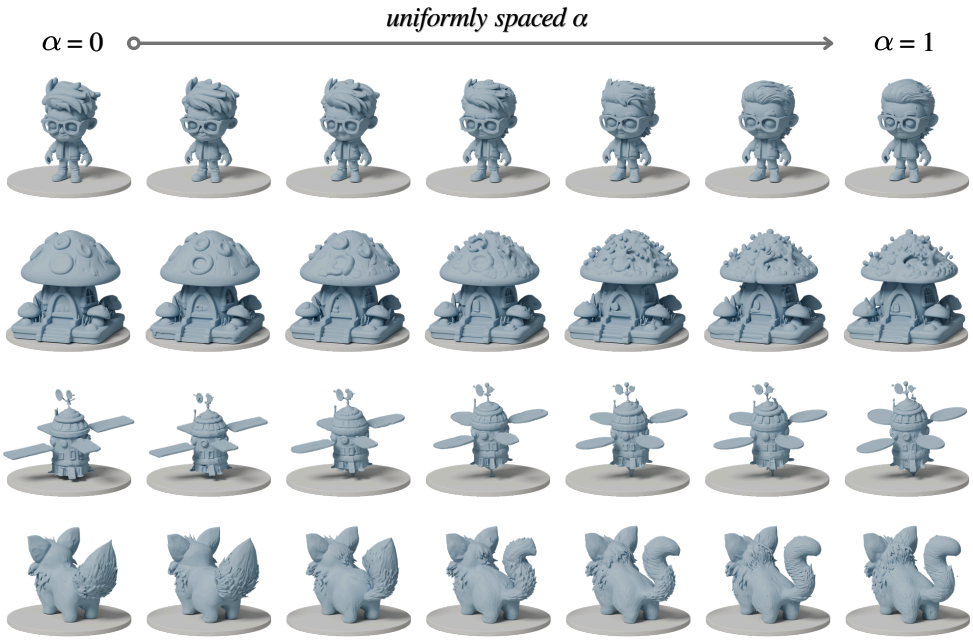

Figure 6: FlowMorpher on TripoSG



Figure 7: FlowMorpher on TRELLIS.2.

native foreground extraction, square cropping, and alpha compositing procedure, and obtain separate DINOv3 (Siméoni et al., 2025) image conditions at the resolutions required by each stage; CFG uses an all-zero negative condition. VFI is applied during sparse-structure, shape structured-latent, and texture structured-latent sampling. In the 1024-cascade configuration, the sampled sparse structure defines the support for the 512-resolution shape stage; the resulting shape latent is decoded to

obtain an upsampled support, on which the 1024-resolution shape stage starts from fresh Gaussian noise. Texture sampling subsequently uses the final shape support and the normalized shape latent as an additional concatenated condition, shared by the source and target evaluations. Thus, each α generates its own structure and shape-dependent texture. At every sampling step, both image-conditioned branches are evaluated at the same current latent state and timestep. CFG and its rescaling operation are applied separately to each branch before interpolating the resulting velocities, $v_\alpha = (1-\alpha)\tilde{v}_{\text{src}} + \alpha\tilde{v}_{\text{tar}}$; the native rescaling operates on the predicted clean latent before converting it back to velocity. Sampling then follows the explicit Euler update,

$$z_{t_{k+1}} = z_{t_k} + (t_{k+1} - t_k)v_\alpha(z_{t_k}, t_k), \tag{8}$$

where $u_k = 1 - k/12$ and $t_k = ru_k/(1 + (r-1)u_k)$. We retain the checkpoint's sampling settings: 12 updates per sampling pass, including both shape passes; timestep rescaling factors $r = 5$ for sparse structure and $r = 3$ for shape and texture; and CFG strength 7.5 with rescaling weights 0.7 and 0.5 for structure and shape, respectively, active over $t \in [0.6, 1]$. Texture uses CFG strength 1 and no rescaling. Each stage is initialized with standard Gaussian noise, and the PyTorch random seed is reset to 42 before sampling each α. This reproduces the initial dense structure noise across morphing weights, but does not establish coordinate-wise noise correspondence in subsequent sparse stages, whose supports and random-number consumption can vary with α. We sample seven morphings, $\alpha \in \left\{\frac{k}{6} \mid k = 0, \ldots, 6\right\}$, and decode the resulting shape and texture latents using the pretrained decoders. Outputs comprise textured GLB meshes, exported with remeshing, a one-million-face decimation target, and 4096^2 texture maps, together with 512×512, 120-frame orbit videos at 30 fps and a comparison video.

A.4 Analysis of Failure Cases



Figure 8: Failure case visualizations

Beyond the quantitative and qualitative results above, we observe failure cases in extreme settings where the source and target belong to distant categories and the overall semantic change is large (Figure 8). In these cases, FlowMorpher still produces visible changes between adjacent morphing weights. For example, when morphing a humanoid dragon into a furry fox, the dragon's color shifts progressively toward the fox's red fur. However, these gradual changes are confined to local attributes, and the sequence still contains an abrupt transition in overall geometry and appearance. Similarly, when morphing a humanoid robot into space station, the appearance changes gradually across adjacent weights, yet an abrupt jump occurs at intermediate weights.

We attribute these failures to the limited coverage of the pretrained 3D generative prior. When the source and target differ substantially in both geometry and appearance, plausible intermediate assets may lie far outside the data distribution learned by the model. In that case, the interpolated velocity field has no well-supported region of intermediate shapes to follow, and the sampling trajectory is pulled toward one of the two endpoint modes. This observation is consistent with the scope of our analysis (Appendix B.6). Latent continuity in α does not guarantee perceptual smoothness. In addition, the support selected by occupancy thresholding in the SS stage can change discontinuously, which yields abrupt changes in the decoded geometry even when the underlying latent varies continuously. For image pairs with such large semantic gaps, style or appearance transfer may be more suitable than generating intermediate assets from a 3D prior that does not cover them.

A.5 Additional Qualitative Results

Here, we present more qualitative comparisons including color transition in Figure 9, shape transition in Figure 10, and semantic transition in Figure 11. Additional results are shown in Figure 12.

In Figure 9, FlowMorpher is the only method whose sequence shows a visibly gradual color transition. Interp3D comes closest, but its sequence still changes abruptly at intermediate morphing weights. The remaining baselines exhibit the failure modes illustrated in Figure 2: FreeMorph and DiffMorpher introduce geometry and texture artifacts, VoxHammer fails to produce a gradual transition. FlowMorpher best preserves thin structures and high-frequency geometric details, consistent with the portal example in Figure 5. The helicopter example in Figure 10 shows this most clearly: under FlowMorpher, the rotor blades thicken progressively as α increases. Interp3D controls the transition through feature correspondences. Because the blades have highly similar features, the matches become ambiguous, and the method cannot reliably distinguish one blade from another. We attribute the smoother result of FlowMorpher to the fact that it requires no explicit correspondences: every trajectory starts from shared Gaussian noise, and both conditional velocities are evaluated at the same evolving state, so all blades evolve under the same interpolated dynamics. The inversion-based method explicitly partitions the asset into preserved and morphed regions using 3D masks. However, features from the morphed region can leak across the mask boundary and affect the surrounding voxels. FlowMorpher also captures the evolution of small structures under larger semantic changes. In Figure 11, the hair first deforms gradually toward the shape of the horns, and the horns then grow out at larger morphing weights, so the horns emerge progressively. In Figure 12, we present additional results spanning humanoids, creatures, buildings, and objects to illustrate FlowMorpher's performance across diverse categories.

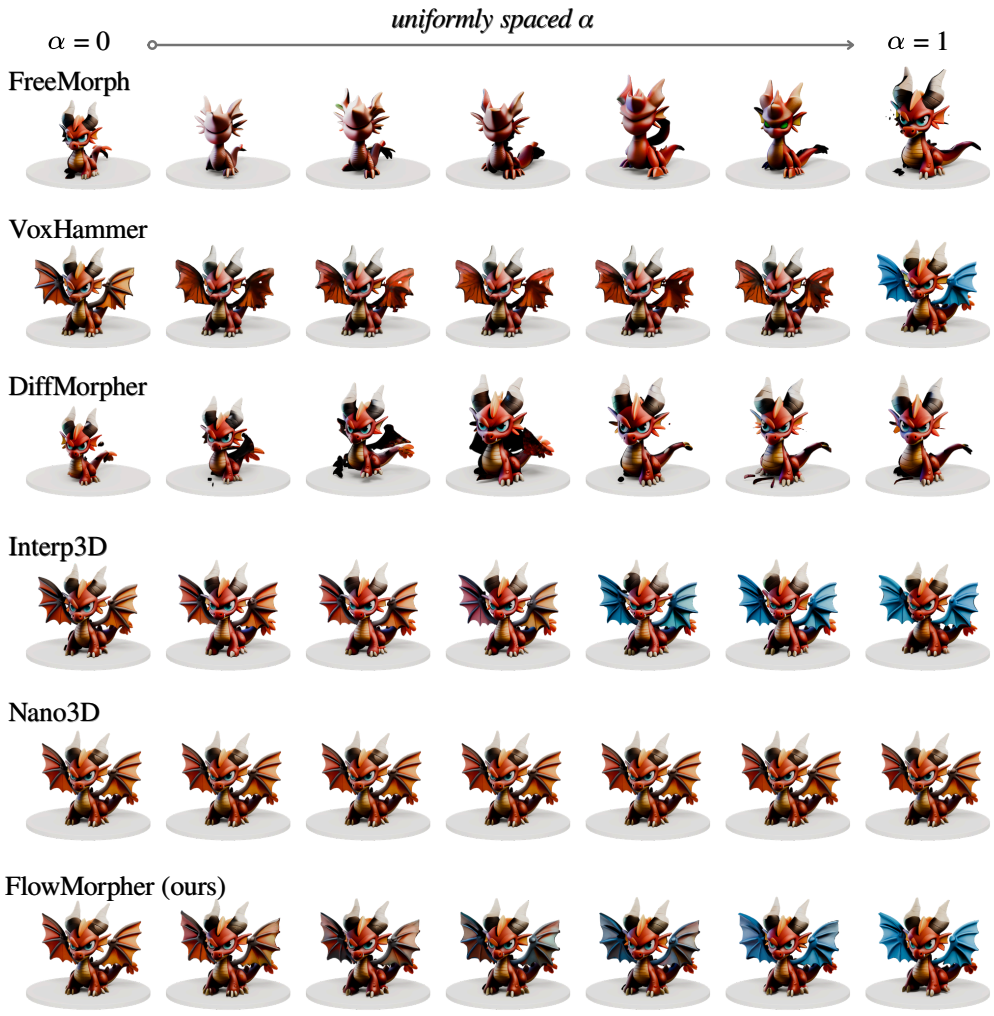

Figure 9: Qualitative comparison of color transitions.

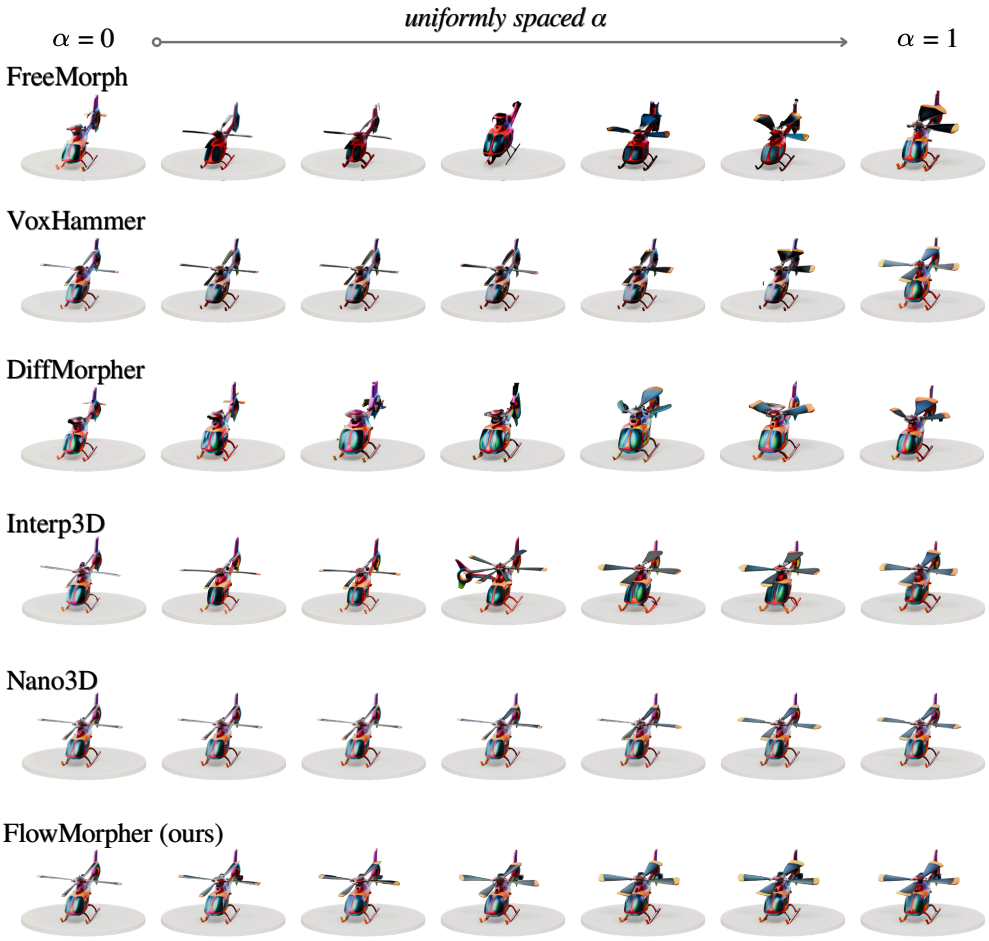

Figure 10: Qualitative comparison of shape transition.

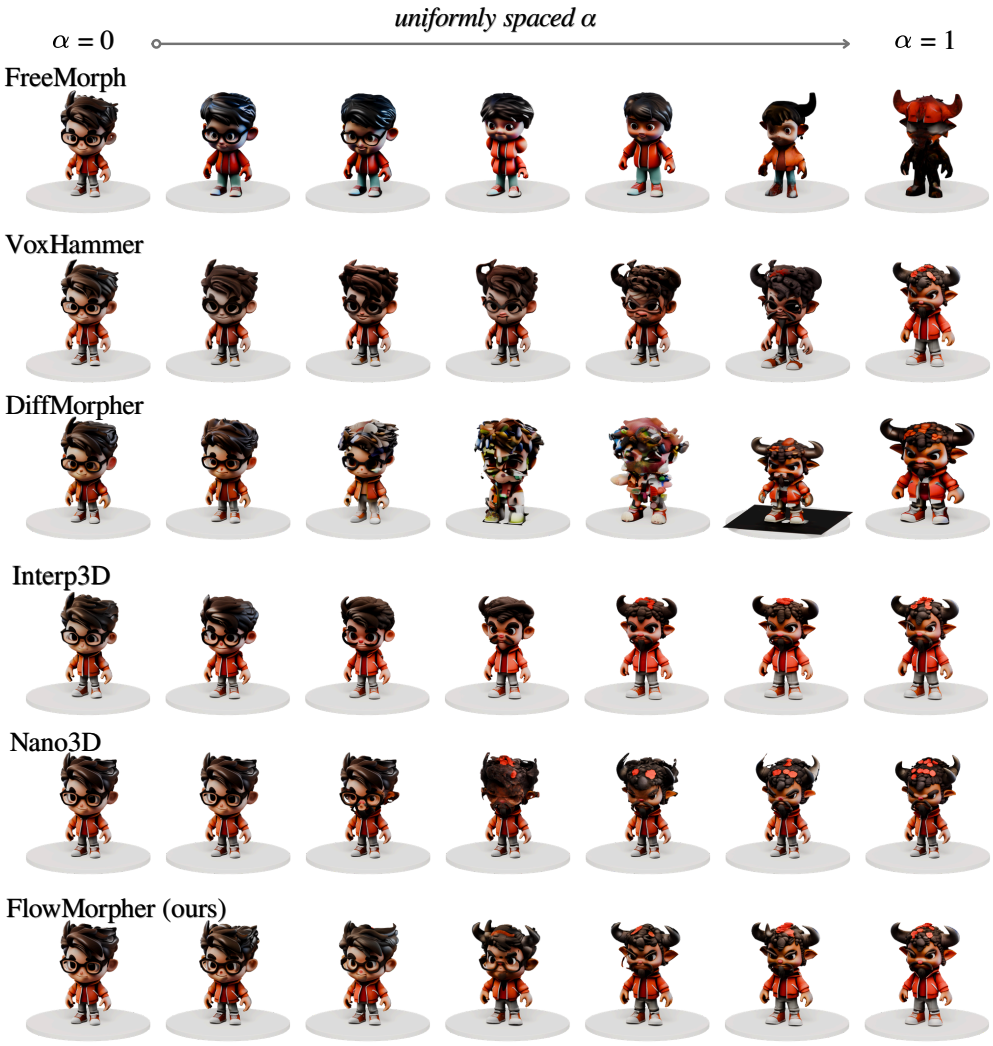

Figure 11: Qualitative comparison of semantic transition.



Figure 12: Additional qualitative results of FlowMorpher.

B PROPERTIES OF VELOCITY FIELD INTERPOLATION

We establish recovery of the model-generated endpoints, equivalence under shared affine CFG, and Lipschitz continuity of the terminal latent with respect to the morphing weight. The continuity analysis concerns a fixed continuous latent space and includes a direct bound for deterministic Euler sampling. For each fixed weight α, sampling integrates a separate trajectory in t; the morphing sequence consists of the terminal outputs across α.

B.1 SETTING AND ASSUMPTIONS

Fix a pair of conditions, the pretrained network, and one realization of the Gaussian initial noise $Z_1 \sim \mathcal{N}(0, I)$. Latents lie in $\mathbb{R}^d$, with Euclidean norm $\|\cdot\|$. We use the convention $t = 1$ for noise and $t = 0$ for data, and integrate $\mathrm{d}Z_t = v(Z_t, t)\,\mathrm{d}t$ from 1 to 0. Sampling fields supplied with a different sign or time parameterization must first be expressed in this convention, including the chain-rule factor for a time change. The constants below may depend on the fixed conditions, noise realization, sampling settings, and common region, but are uniform in α.

Let $v^c(z, t) \in \mathbb{R}^d$ denote the complete deterministic velocity field used for standard generation under condition $c \in \{\mathrm{src}, \mathrm{tar}\}$, including any per-condition guidance or rescaling. The fields are required to be defined and regular on the common region specified below. We reserve u^c for the unguided network prediction; v^c retains its meaning throughout the appendix. Both fields are fixed independently of α. For $\alpha \in [0, 1]$, define

$$\begin{aligned} v^\alpha(z, t) &:= (1 - \alpha)v^{\mathrm{src}}(z, t) + \alpha v^{\mathrm{tar}}(z, t), \\ \delta(z, t) &:= v^{\mathrm{tar}}(z, t) - v^{\mathrm{src}}(z, t). \end{aligned} \tag{9}$$

Thus, at the current sampling state,

$$v_t^\alpha = v^\alpha(Z_t^\alpha, t) = v^{\mathrm{src}}(Z_t^\alpha, t) + \alpha\big[v^{\mathrm{tar}}(Z_t^\alpha, t) - v^{\mathrm{src}}(Z_t^\alpha, t)\big]. \tag{10}$$

The symbol δ denotes a difference evaluated at the same state and time; it is distinct from a paired-query velocity difference in an editing baseline. The results analyze interpolation of the complete fields v^{src} and v^{tar}, including the per-condition guidance and rescaling used in our TRELLIS.2 implementation (Remark 2).

Continuous and discrete outputs. The continuous trajectory is defined by

$$Z_t^\alpha = Z_1 - \int_t^1 v^\alpha(Z_\tau^\alpha, \tau)\,\mathrm{d}\tau, \qquad X^\alpha := Z_0^\alpha. \tag{11}$$

For Euler sampling, fix a grid $1 = t_0 > t_1 > \cdots > t_N = 0$, independent of α, and let $h_k := t_k - t_{k+1} > 0$. Then $\sum_{k=0}^{N-1} h_k = 1$, and

$$\begin{aligned} \bar{Z}_{t_{k+1}}^\alpha &= \bar{Z}_{t_k}^\alpha - h_k v^\alpha(\bar{Z}_{t_k}^\alpha, t_k), \\ \bar{Z}_{t_0}^\alpha &= Z_1, \qquad \bar{X}^\alpha := \bar{Z}_{t_N}^\alpha. \end{aligned} \tag{12}$$

Standard generation under condition c uses the same definitions with v^α replaced by v^c; its outputs are X^c and $\bar{X}^c$, respectively. Continuous and discrete outputs are distinguished throughout. For Euler sampling, the main-text notation $Z_{t_k}^\alpha$ and X^α corresponds to $\bar{Z}_{t_k}^\alpha$ and $\bar{X}^\alpha$ here, respectively; the same convention applies to the source and target endpoints.

Assumption 1 (Regularity and common containment)**.** *There is a compact set $\Omega \subset \mathbb{R}^d$ such that the following hold for $c \in \{\mathrm{src}, \mathrm{tar}\}$.*

(i) There is a common finite partition $0 = s_0 < \cdots < s_m = 1$ such that v^c is jointly continuous on $\Omega \times (s_{i-1}, s_i)$ for every i.

(ii) There is a finite $B \geq 0$ such that $\|v^c(z, t)\| \leq B$ on $\Omega \times [0, 1]$.

(iii) There is a finite $L_c \geq 0$ such that

$$\|v^c(z, t) - v^c(z', t)\| \leq L_c \|z - z'\| \quad \text{for all } z, z' \in \Omega,\ t \in [0, 1].$$

(iv) For the continuous statements, a solution of Eq. 11 exists on $[0,1]$ *for every* $\alpha \in [0,1]$*, and every such solution remains in* Ω*. For the discrete statements, all Euler iterates under consideration remain in* Ω*. These containment requirements are assumed separately for the continuous and discrete trajectories.*

Define the uniform bounds

$$L := \max(L_{\mathrm{src}}, L_{\mathrm{tar}}), \qquad M := \sup_{(z,t)\in\Omega\times[0,1]} \|\delta(z,t)\| \le 2B. \tag{13}$$

Remark 1 (Interpretation of the assumptions). Existence and containment in (iv) do not follow from bounds imposed only on Ω. A sufficient condition for containment of existing solutions and Euler iterates is that the fields are defined on all of $\mathbb{R}^d$ and satisfy a global velocity bound $\|v^c(z,t)\| \le B$: convexity gives $\|v^\alpha\| \le B$, and integration or summation gives $\|Z_t^\alpha - Z_1\| \le B(1-t)$ and $\left\|\bar{Z}_{t_k}^\alpha - Z_1\right\| \le B\sum_{j<k} h_j$. Both therefore lie in the closed ball of radius B about Z_1. No such global bound is asserted for a trained network; the analysis is conditional on the stated assumptions. The partition in (i) allows finite guidance switches. The Euler results need bounds and containment only at their query times, with no continuity assumption in t.

B.2 Basic estimates

Lemma 1 (Bounds for the interpolated field)**.** *Under Assumption 1, for* $z, z' \in \Omega$*,* $t \in [0,1]$*, and* $\alpha, \alpha' \in [0,1]$*,*

$$\|v^\alpha(z,t) - v^\alpha(z',t)\| \le L\,\|z - z'\|\,, \tag{14}$$

$$v^\alpha(z,t) - v^{\alpha'}(z,t) = (\alpha - \alpha')\delta(z,t), \tag{15}$$

$$\|v^\alpha(z,t)\| \le B.$$

In particular, the norm in Eq. 15 is at most $M|\alpha - \alpha'|$*.*

Proof. The first bound follows by the triangle inequality with constant $(1-\alpha)L_{\mathrm{src}} + \alpha L_{\mathrm{tar}} \le L$. The identity follows by subtracting $v^{\mathrm{src}} + \alpha\delta$ and $v^{\mathrm{src}} + \alpha'\delta$. The last bound follows from convexity. □

Lemma 2 (Backward Grönwall estimate)**.** *Let* $u : [0,1] \to [0,\infty)$ *be continuous, let* b *be nonnegative and integrable, and let* $c, L \ge 0$*. If* $u(t) \le c + \int_t^1 [Lu(\tau) + b(\tau)]\,\mathrm{d}\tau$*, then*

$$u(t) \le ce^{L(1-t)} + \int_t^1 e^{L(\tau-t)} b(\tau)\,\mathrm{d}\tau. \tag{16}$$

Proof. Set $U(t) := c + \int_t^1 [Lu(\tau) + b(\tau)]\,\mathrm{d}\tau$. Since u is continuous and b is integrable, U is absolutely continuous. Moreover, $u \le U$, $U(1) = c$, and $U' = -Lu - b \ge -LU - b$ almost everywhere. Integrating $(e^{Lt}U(t))' \ge -e^{Lt}b(t)$ from t to 1 and rearranging gives Eq. 16. □

B.3 Endpoint recovery

Proposition 1 (Recovery of model-generated endpoints)**.** *Under Assumption 1, the continuous outputs satisfy* $X^0 = X^{\mathrm{src}}$ *and* $X^1 = X^{\mathrm{tar}}$*. For any shared Euler grid with well-defined iterates,* $\bar{X}^0 = \bar{X}^{\mathrm{src}}$ *and* $\bar{X}^1 = \bar{X}^{\mathrm{tar}}$*.*

Proof. At $\alpha = 0$, $v^0 = v^{\mathrm{src}}$ pointwise. Both continuous trajectories lie in Ω by Assumption 1(iv), so the Lipschitz bound in (iii) applies to their states. Their separation is continuous, and their initializations agree; subtracting their integral equations therefore gives

$$\left\|Z_t^0 - Z_t^{\mathrm{src}}\right\| \le \int_t^1 L_{\mathrm{src}} \left\|Z_\tau^0 - Z_\tau^{\mathrm{src}}\right\| \mathrm{d}\tau.$$

Lemma 2, with $c = 0$ and $b = 0$, makes this separation zero for every t. For Euler, the initial states agree and equality at step k implies equality at step $k+1$, because the update fields and step sizes agree. Induction yields equality of the discrete endpoints without regularity assumptions on the fields. The argument at $\alpha = 1$ is identical with $v^1 = v^{\mathrm{tar}}$. □

These identities recover the corresponding model-generated samples under matched initialization and sampling settings; they do not assert reconstruction of arbitrary supplied 3D assets. They require neither straight trajectories nor constant velocities. The discrete argument also applies to any deterministic solver whose complete update rule and initial internal state agree with those of the reference generation. For a multi-stage pipeline, endpoint equality propagates through the stages when every corresponding initialization, sampling setting, and deterministic intermediate operation matches. This propagation requires equality of intermediate inputs, not continuity of the intermediate operations.

Floating-point implementation. The identities are algebraic statements in exact arithmetic. Bitwise equality additionally requires the same arithmetic and evaluation context, including batching and kernel choices. Evaluating source, target, and null predictions in a different batch arrangement can introduce rounding differences even with the same seed. The continuity bounds below likewise describe real-arithmetic fields and update rules.

B.4 CLASSIFIER-FREE GUIDANCE

Let u^c denote the unguided conditional velocity and let $u^\varnothing$ denote the unguided prediction under a shared null or negative condition. Define $u^\alpha := (1-\alpha)u^{\mathrm{src}} + \alpha u^{\mathrm{tar}}$. In the special case of affine CFG without nonlinear rescaling, the complete sampling field is

$$v^c(z,t) = u^\varnothing(z,t) + \omega_t\big[u^c(z,t) - u^\varnothing(z,t)\big], \tag{17}$$

where ω_t is a common guidance schedule. This covers the conventions $u^\varnothing + s(u^c - u^\varnothing)$ with $\omega_t = s$ and $(1+s)u^c - su^\varnothing$ with $\omega_t = 1+s$. Guidance is off when $\omega_t = 1$, which can be imposed outside a fixed guidance interval.

Proposition 2 (Equivalence under shared affine guidance)**.** *If both conditions use the same $u^\varnothing$ and schedule ω_t in Eq. 17, then, at every common state z and time t,*

$$\begin{aligned} v^\alpha(z,t) &= (1-\alpha)v^{\mathrm{src}}(z,t) + \alpha v^{\mathrm{tar}}(z,t) \\ &= u^\varnothing(z,t) + \omega_t\big[u^\alpha(z,t) - u^\varnothing(z,t)\big] \end{aligned} \tag{18}$$

for $\alpha \in [0,1]$.

Proof. Substituting $v^c = (1-\omega_t)u^\varnothing + \omega_t u^c$ into the convex combination gives $(1-\omega_t)u^\varnothing + \omega_t[(1-\alpha)u^{\mathrm{src}} + \alpha u^{\mathrm{tar}}]$, since the interpolation weights sum to one. □

This is a pointwise identity and therefore holds at every continuous or discrete solver query. The null prediction must use the same negative prompt or null embedding, as well as the same state and time, for both conditions. Here v^c is the actual affine-guided field, so Proposition 1 applies to the corresponding guided generations, with regularity and containment imposed on these complete fields for its continuous statement.

Remark 2 (Guidance rescaling in TRELLIS.2). Our TRELLIS.2 implementation applies CFG and any enabled rescaling separately under each condition, then interpolates the returned velocity predictions. Let $\mathcal{R}(z,t;q,n)$ denote this processing rule, including prediction conversions, for conditional prediction q and shared null prediction n. The same schedules and rescaling rule are used for both conditions, with all auxiliary conditioning fixed. Thus, the complete fields and their interpolation are

$$\begin{aligned} v^c(z,t) &= \mathcal{R}\big(z,t;u^c(z,t),u^\varnothing(z,t)\big), \\ v^\alpha(z,t) &= (1-\alpha)\mathcal{R}\big(z,t;u^{\mathrm{src}}(z,t),u^\varnothing(z,t)\big) \\ &\quad + \alpha\mathcal{R}\big(z,t;u^{\mathrm{tar}}(z,t),u^\varnothing(z,t)\big). \end{aligned} \tag{19}$$

Although rescaling is nonlinear in its conditional input, Eq. 19 is affine in α at each fixed state and time. It therefore has exactly the form of Eq. 9. Proposition 1 gives endpoint recovery, and Theorem 1 and Proposition 3 give continuity under Assumption 1 for these processed fields and their trajectories. The constants L and M in Eq. 13 must bound the processed fields; the affine CFG bounds in Corollary 1 do not automatically apply when rescaling is enabled. Without rescaling, Proposition 2 applies under its shared-null and guidance-schedule conditions.

Rescaling uses the ratio of the standard deviations of the conditional and guided clean predictions, blends the rescaled and guided clean predictions, and converts the result back to a velocity. On a fixed tensor layout, sufficient conditions for Assumption 1(i)–(iii) are bounded raw fields, uniformly Lipschitz dependence on the state, bounded guidance and rescale schedules, and bounded, uniformly Lipschitz prediction conversions on the relevant domains. All components must have the stated piecewise continuity in time, with every processing switch included in a common finite partition. Additionally, every guided-clean standard deviation used as a denominator must be bounded below by $\eta > 0$ for both conditions on Ω wherever rescaling is active. Standard deviation is Lipschitz on a fixed finite-dimensional layout; on bounded inputs, the positive denominator bound makes the required ratios and products bounded and Lipschitz as well. Existence and common containment remain separate assumptions. Conversion back to a velocity can involve inverse noise-coefficient factors, so its Lipschitz bound may be large, or unbounded without a positive noise floor. For a fixed Euler grid, the required bounds concern only the query times $t_0, \dots, t_{N-1} > 0$, but their constants may still be large.

B.5 Continuity in α

Theorem 1 (Lipschitz continuity of the terminal latent)**.** *Under Assumption 1 and shared initialization, for all $\alpha, \alpha' \in [0,1]$ and $t \in [0,1]$,*

$$\left\| Z_t^{\alpha} - Z_t^{\alpha'} \right\| \le |\alpha - \alpha'| \int_t^1 e^{L(\tau - t)} \left\| \delta(Z_\tau^{\alpha'}, \tau) \right\| \, \mathrm{d}\tau. \tag{20}$$

In particular,

$$\left\| X^{\alpha} - X^{\alpha'} \right\| \le K|\alpha - \alpha'|, \qquad K := M \int_0^1 e^{L\tau} \, \mathrm{d}\tau = \begin{cases} M(e^L - 1)/L, & L > 0, \\ M, & L = 0. \end{cases} \tag{21}$$

The constant K is independent of α and α'.

Proof. Let $e(t) := \left\| Z_t^{\alpha} - Z_t^{\alpha'} \right\|$. The trajectories are absolutely continuous by Eq. 11 and boundedness, hence e is continuous as required by Lemma 2. Shared initialization cancels the initial-state term when subtracting the two integral equations. Adding and subtracting $v^{\alpha}(Z_\tau^{\alpha'}, \tau)$ gives

$$\begin{aligned} &v^{\alpha}(Z_\tau^{\alpha}, \tau) - v^{\alpha'}(Z_\tau^{\alpha'}, \tau) \\ &\quad = \left[v^{\alpha}(Z_\tau^{\alpha}, \tau) - v^{\alpha}(Z_\tau^{\alpha'}, \tau) \right] + (\alpha - \alpha')\delta(Z_\tau^{\alpha'}, \tau). \end{aligned}$$

Lemma 1 and common containment therefore imply

$$e(t) \le \int_t^1 \left[L e(\tau) + |\alpha - \alpha'| \left\| \delta(Z_\tau^{\alpha'}, \tau) \right\| \right] \mathrm{d}\tau.$$

The forcing term is bounded and integrable by Assumption 1(i)–(ii). Lemma 2 with $c = 0$ gives Eq. 20. At $t = 0$, replacing the trajectory-dependent difference by its uniform upper bound M gives Eq. 21. The trajectory integral may depend on α'; the use of the supremum over the common $\Omega \times [0,1]$ makes K uniform in both weights. □

Proposition 3 (Continuity of Euler outputs)**.** *Suppose the common bounds and discrete containment in Assumption 1(ii)–(iv) hold at the grid times. For the shared, possibly nonuniform grid,*

$$\begin{aligned} \left\| \bar{X}^{\alpha} - \bar{X}^{\alpha'} \right\| &\le |\alpha - \alpha'| \sum_{k=0}^{N-1} h_k \left\| \delta(\bar{Z}_{t_k}^{\alpha'}, t_k) \right\| \prod_{j=k+1}^{N-1} (1 + h_j L) \\ &\le K_{\mathcal{T}} |\alpha - \alpha'| \le K |\alpha - \alpha'|, \end{aligned} \tag{22}$$

where an empty product equals one, K is defined in Eq. 21, and

$$K_{\mathcal{T}} := \begin{cases} \dfrac{M}{L} \left[\displaystyle\prod_{k=0}^{N-1} (1 + h_k L) - 1 \right], & L > 0, \\ M, & L = 0. \end{cases} \tag{23}$$

The upper bound K is uniform over any family of grids sharing valid common bounds L, M and containment region Ω.

Proof. Set $e_k := \left\| \bar{Z}^{\alpha}_{t_k} - \bar{Z}^{\alpha'}_{t_k} \right\|$ and $d_k := \left\| \delta(\bar{Z}^{\alpha'}_{t_k}, t_k) \right\|$. The same decomposition as above gives

$$e_0 = 0, \qquad e_{k+1} \le (1 + h_k L) e_k + h_k |\alpha - \alpha'| d_k.$$

Unrolling proves the first inequality in Eq. 22. Using $d_k \le M$, the resulting sum telescopes for $L > 0$:

$$M \sum_{k=0}^{N-1} h_k \prod_{j=k+1}^{N-1} (1 + h_j L) = \frac{M}{L} \left[\prod_{k=0}^{N-1} (1 + h_k L) - 1 \right].$$

Since $\prod_k (1+h_k L) \le \exp(L \sum_k h_k) = e^L$, this is at most K. For $L = 0$, the sum is $M \sum_k h_k = M$. This argument analyzes the discrete updates directly and requires no convergence claim for Euler as the grid is refined. □

Corollary 1 (Continuity under affine CFG)**.** *Suppose the raw fields $u^{\mathrm{src}}, u^{\mathrm{tar}}, u^{\varnothing}$ obey the regularity conditions in Assumption 1(i)–(iii) on a common Ω. Let B_u bound both conditional fields, let $B_{u,\varnothing}$ bound the null field, and let their state Lipschitz constants be $L_{u,\mathrm{src}}, L_{u,\mathrm{tar}}, L_{u,\varnothing}$. Define $L_u := \max(L_{u,\mathrm{src}}, L_{u,\mathrm{tar}})$ and $M_u := \sup_{\Omega \times [0,1]} \|u^{\mathrm{tar}} - u^{\mathrm{src}}\|$. Let ω_t be bounded and continuous on each interval of a common finite partition, and assume the relevant guided solutions and iterates satisfy the containment and existence requirements in (iv). Then Theorem 1 and Proposition 3 apply to the complete affine-guided fields v^c of Eq. 17, with the admissible bounds*

$$\begin{aligned} \tilde{L} &:= \sup_{t \in [0,1]} \big(|1 - \omega_t| L_{u,\varnothing} + |\omega_t| L_u \big), \\ \tilde{M} &:= \left(\sup_{t \in [0,1]} |\omega_t| \right) M_u, \\ \tilde{K} &:= \tilde{M} \int_0^1 e^{\tilde{L}\tau} \, \mathrm{d}\tau = \begin{cases} \tilde{M}(e^{\tilde{L}} - 1)/\tilde{L}, & \tilde{L} > 0, \\ \tilde{M}, & \tilde{L} = 0. \end{cases} \end{aligned} \tag{24}$$

Proof. The affine combination in Eq. 17 is piecewise continuous and bounded by $\tilde{B} := \sup_t (|1 - \omega_t| B_{u,\varnothing} + |\omega_t| B_u)$. Its state Lipschitz constant is at most $\tilde{L}$. The shared null terms cancel in the difference, giving $\delta = v^{\mathrm{tar}} - v^{\mathrm{src}} = \omega_t (u^{\mathrm{tar}} - u^{\mathrm{src}})$, whose norm is at most $\tilde{M}$. The continuous and discrete arguments therefore apply with these guided-field bounds. □

B.6 Interpretation and scope

Remark 3 (Initialization and interpretation of the bound). *Shared initialization.* If initialization varies with α and the corresponding trajectories satisfy the same regularity and common-containment conditions, the same argument gives

$$\left\| X^{\alpha} - X^{\alpha'} \right\| \le e^L \left\| Z^{\alpha}_1 - Z^{\alpha'}_1 \right\| + K |\alpha - \alpha'|. \tag{25}$$

An analogous bound holds for Euler, since the initial separation is amplified by at most $\prod_k (1 + h_k L) \le e^L$. Shared initialization removes the initial-state term entirely, isolating the effect of the interpolated dynamics. It is sufficient but not necessary for continuity: continuous initialization makes that term vanish as $\alpha' \to \alpha$; a K_0-Lipschitz initialization gives terminal constant $e^L K_0 + K$. Independent noise resampling generally provides no such continuity guarantee.

Meaning of the bound. The result establishes a finite latent-space Lipschitz constant under the assumptions. The network bounds may be unknown or large, especially under guidance or rescaling, so K is not a calibrated estimate of change at a chosen weight spacing. Latent continuity does not guarantee perceptual smoothness, monotone semantic change, or preservation of a prescribed data distribution.

Continuous latents and discrete 3D representations. All continuity comparisons require a fixed latent space, fixed auxiliary conditioning, and two complete sampling fields that are independent of α. In TRELLIS, Theorem 1 and Proposition 3 apply directly to the continuous SS latent before occupancy thresholding, and to SLAT features only for a fixed support with a consistent initialization. Even

equal numbers of active voxels do not identify the same latent space and fields if their coordinates differ, and resetting a random seed does not by itself identify noise vectors on different supports. Section B.7 derives consequences for support selection, decoded attributes, and extracted meshes under additional stated conditions. The same restrictions apply to the structure-dependent stages of TRELLIS.2; for TripoSG, the analysis covers the fixed-dimensional continuous latent, and extracted meshes only under the fixed-branch conditions of Section B.7.

Sampling scope. The proofs cover the stated deterministic ODE and fixed-grid Euler updates, including guidance intervals represented by a common time schedule. A sampler that injects additional stochastic noise is outside these statements and requires a separate analysis specifying how that noise is coupled across weights. Remark 6 gives such an analysis for samplers with shared additive noise.

B.7 Continuity of decoded 3D assets

The latent bounds of Theorem 1 and Proposition 3 are standard stability estimates, and we do not claim novelty for the Grönwall argument. We establish its consequences for occupancy selection, decoded attributes, and extracted geometry under explicit additional conditions. Throughout this subsection, $\bar{X}_s^\alpha$ denotes the Euler output of stage s, consistently with Proposition 3; X_s^α denotes its exact ODE counterpart. Each stage uses a fixed grid shared across weights, but different stages may use different grids. An ODE extension is stated only where the corresponding continuous-time assumptions are imposed. All velocity fields are the complete sampling fields, including per-condition guidance and rescaling when used.

What the VFI construction controls. Consider two absolutely continuous trajectories with initial states Z_1, Z_1', driven by w, w' in a common region, where w is uniformly L-Lipschitz in its state and the velocity differences below are integrable. Adding and subtracting $w(Z_\tau', \tau)$ in their integral equations gives

$$\|Z_t - Z_t'\| \le \|Z_1 - Z_1'\| + \int_t^1 \left[L\|Z_\tau - Z_\tau'\| + \|(w - w')(Z_\tau', \tau)\|\right] \mathrm{d}\tau.$$

The backward Grönwall estimate in Lemma 2 therefore yields

$$\|Z_0 - Z_0'\| \le e^L \|Z_1 - Z_1'\| + \int_0^1 e^{L\tau} \|(w - w')(Z_\tau', \tau)\| \,\mathrm{d}\tau. \tag{26}$$

For VFI, shared initialization removes the first term, and same-state interpolation gives

$$v^\alpha(z,t) - v^{\alpha'}(z,t) = (\alpha - \alpha')\big[v^{\mathrm{tar}}(z,t) - v^{\mathrm{src}}(z,t)\big].$$

The forcing bound is $M|\alpha - \alpha'|$, with state Lipschitz bound $L = \max(L_{\mathrm{src}}, L_{\mathrm{tar}})$. These constants concern the two original-condition fields on the common region containing the morphing trajectories; they are not bounds measured only along the two reference trajectories. The discrete counterpart follows from the Euler recurrence in Proposition 3. This explains how the sampling construction enters the estimate, without asserting a comparative smoothness guarantee over other interpolation rules.

Two-stage setting. Let $\mathcal{G} \subset \mathbb{Z}^3$ be a fixed finite voxel grid and define

$$\ell^\alpha := \mathcal{D}_{\mathrm{SS}}(\bar{X}_{\mathrm{SS}}^\alpha), \qquad \mathcal{P}^\alpha := \{\boldsymbol{p} \in \mathcal{G} : \ell^\alpha(\boldsymbol{p}) > \gamma\}, \tag{27}$$

where $\mathcal{D}_{\mathrm{SS}}$ produces occupancy logits and γ is a fixed threshold. Write

$$\begin{aligned} m(\alpha) &:= \min_{\boldsymbol{p} \in \mathcal{G}} |\ell^\alpha(\boldsymbol{p}) - \gamma|, \\ n_\alpha(r) &:= \big|\{\boldsymbol{p} \in \mathcal{G} : |\ell^\alpha(\boldsymbol{p}) - \gamma| \le r\}\big|, \qquad r \ge 0. \end{aligned} \tag{28}$$

For a fixed support $\mathcal{P}$, SLAT states lie in $\mathbb{R}^{\mathcal{P} \times C}$ with a fixed coordinate order. Let $v_{\mathcal{P}}^c(z,t)$, $c \in \{\mathrm{src}, \mathrm{tar}\}$, be the complete SLAT fields with all other conditioning fixed, and set $v_{\mathcal{P}}^\alpha = (1-\alpha)v_{\mathcal{P}}^{\mathrm{src}} + \alpha v_{\mathcal{P}}^{\mathrm{tar}}$. Fix a coordinate-indexed field $\epsilon_{\mathrm{SLAT}}(\boldsymbol{p})$ and let $E_{\mathcal{P}} := (\epsilon_{\mathrm{SLAT}}(\boldsymbol{p}))_{\boldsymbol{p} \in \mathcal{P}}$. This specifies the noise coupling analyzed below; coordinate-wise coupling is not implied merely by resetting a random seed.

Let $\mathcal{A}_{\mathcal{P}} : \mathbb{R}^{\mathcal{P}\times C} \to \mathbb{R}^{q(\mathcal{P})}$ map terminal features, through denormalization and decoding, to an attribute vector with a fixed indexing for that support. Define $a^{\alpha} := \mathcal{A}_{\mathcal{P}^{\alpha}}(\bar{X}^{\alpha}_{\mathrm{SLAT}})$. For Gaussian representations, the vector may include centers, scales, rotations, opacities, and color coefficients. For mesh extraction, it includes the required scalar-field values, deformations, extraction weights, and any color attributes, before discrete extraction decisions. Any decoder operation that changes this output layout must also be fixed or analyzed separately. The Lipschitz assumptions on these maps are explicit conditions; they are not inferred from compactness alone.

Corollary 2 (Support stability and decoded attributes)**.** *Suppose the Euler assumptions of Proposition 3 hold for the SS stage, with terminal constant K_{SS}. Suppose $\mathcal{D}_{\mathrm{SS}}$ is L_D-Lipschitz from the Euclidean norm to the max-norm on a set containing all $\bar{X}^{\alpha}_{\mathrm{SS}}$. Set $\kappa := L_D K_{\mathrm{SS}}$. Then the following hold.*

(i) For all $\alpha, \alpha' \in [0,1]$,

$$\begin{aligned} \mathcal{P}^{\alpha} \triangle \mathcal{P}^{\alpha'} &\subseteq \{\boldsymbol{p} : |\ell^{\alpha}(\boldsymbol{p}) - \gamma| \le \kappa|\alpha - \alpha'|\}, \\ |\mathcal{P}^{\alpha} \triangle \mathcal{P}^{\alpha'}| &\le n_{\alpha}(\kappa|\alpha - \alpha'|). \end{aligned} \tag{29}$$

(ii) The margin m is κ-Lipschitz, and $\mathcal{P}^{\alpha'} = \mathcal{P}^{\alpha}$ whenever $\kappa|\alpha' - \alpha| < m(\alpha)$. The support is constant on each connected component of the relatively open set $\{\alpha \in [0,1] : m(\alpha) > 0\}$. A support difference between two weights requires a logit to equal γ somewhere in the closed interval between them.

(iii) Let $I \subseteq [0,1]$ be an interval with $\mathcal{P}^{\alpha} \equiv \mathcal{P}$. Assume that the SLAT initial state is $E_{\mathcal{P}}$ for every $\alpha \in I$, that the two complete SLAT fields and all auxiliary conditioning are independent of α, and that the Euler regularity and common-containment assumptions hold uniformly on I, giving a constant $K_{\mathcal{P}}$. If $\mathcal{A}_{\mathcal{P}}$ is $L_{\mathcal{A},\mathcal{P}}$-Lipschitz on a set containing the corresponding terminal latents, then

$$\begin{aligned} \|\bar{X}^{\alpha}_{\mathrm{SLAT}} - \bar{X}^{\alpha'}_{\mathrm{SLAT}}\| &\le K_{\mathcal{P}}|\alpha - \alpha'|, \\ \|a^{\alpha} - a^{\alpha'}\| &\le L_{\mathcal{A},\mathcal{P}} K_{\mathcal{P}}|\alpha - \alpha'|, \qquad \alpha, \alpha' \in I. \end{aligned} \tag{30}$$

The same statements hold for ODE outputs in place of Euler outputs when the respective existence, regularity, and common-containment assumptions of Theorem 1 hold for both stages.

Proof. By Proposition 3 and decoder Lipschitz continuity,

$$\|\ell^{\alpha} - \ell^{\alpha'}\|_{\infty} \le L_D \|\bar{X}^{\alpha}_{\mathrm{SS}} - \bar{X}^{\alpha'}_{\mathrm{SS}}\| \le \kappa|\alpha - \alpha'|.$$

If a voxel belongs to exactly one of the two supports, its logit is greater than γ at one weight and at most γ at the other. Consequently, $|\ell^{\alpha}(\boldsymbol{p}) - \gamma| \le |\ell^{\alpha}(\boldsymbol{p}) - \ell^{\alpha'}(\boldsymbol{p})|$, which proves (i), including equality at the threshold. For (ii), each function $|\ell^{\alpha}(\boldsymbol{p}) - \gamma|$ is κ-Lipschitz, and their finite minimum has the same Lipschitz bound. If $\kappa|\alpha' - \alpha| < m(\alpha)$, (i) leaves no voxel that can change membership. This proves local constancy on $\{m > 0\}$, hence constancy on each of its connected components. The final claim follows from the intermediate value theorem applied to a logit whose membership changes. For (iii), both trajectories have the same latent space, initialization, fields, and grid. The proof of Proposition 3 therefore applies to each pair of weights in I. Composition with $\mathcal{A}_{\mathcal{P}}$ gives the second inequality. Replacing Proposition 3 with Theorem 1 gives the stated ODE version under its separate assumptions. □

The support bound identifies voxels by their logit margins, not by spatial proximity, and does not imply that few voxels change in practice. If every occurring support admits one common region and finite constants covering all its trajectories, including across disconnected weight intervals, then the maximum of $L_{\mathcal{A},\mathcal{P}} K_{\mathcal{P}}$ over the finitely many occurring supports is finite. This additional uniformity condition is needed to obtain one constant for all fixed-support intervals; interval-specific bounds alone do not establish it.

Extracted geometry and color. For a mesh, a fixed SS support does not determine every discrete extraction decision. Let b specify a fixed extraction branch, including scalar-field sign decisions, triangulation choices, vertex ordering, and any discrete choices affecting attribute correspondence. On that branch, write $H_{\mathcal{P},b}(a) = Y = (V, \Gamma)$, where V lists vertex positions and Γ lists the associated color attributes; the face array F_b is fixed. Colors may instead be represented by coefficients in a fixed texture parameterization, provided their conversion is included in $H_{\mathcal{P},b}$ and satisfies the same regularity condition.

Proposition 4 (Continuity of extracted geometry on a fixed branch)**.** *Under Corollary 2(iii), let $J \subseteq I$ be an interval on which the extraction branch b is fixed. Suppose $H_{\mathcal{P},b}$ is $L_{H,\mathcal{P},b}$-Lipschitz on a set containing $\{a^\alpha : \alpha \in J\}$. Then, for $\alpha, \alpha' \in J$,*

$$\|Y^\alpha - Y^{\alpha'}\| \le C_{\mathcal{P},b}|\alpha - \alpha'|, \qquad C_{\mathcal{P},b} := L_{H,\mathcal{P},b} L_{\mathcal{A},\mathcal{P}} K_{\mathcal{P}}. \tag{31}$$

For the nonempty surfaces S^α formed by the triangles of (V^α, F_b), their Hausdorff distance satisfies

$$d_H(S^\alpha, S^{\alpha'}) \le \max_i \|V_i^\alpha - V_i^{\alpha'}\| \le C_{\mathcal{P},b}|\alpha - \alpha'|. \tag{32}$$

When Γ contains vertex colors, the same constant bounds their linear interpolants at points with corresponding face indices and barycentric coordinates.

Proof. The first inequality follows by composing the Lipschitz bound for $H_{\mathcal{P},b}$ with Eq. 30. For any point $x = \sum_{r=1}^3 \lambda_r V_{i_r}^\alpha$ on a face, with $\lambda_r \ge 0$ and $\sum_r \lambda_r = 1$, the point $x' = \sum_r \lambda_r V_{i_r}^{\alpha'}$ lies on the corresponding face of the other mesh. Therefore

$$\|x - x'\| \le \sum_{r=1}^3 \lambda_r \|V_{i_r}^\alpha - V_{i_r}^{\alpha'}\| \le \max_i \|V_i^\alpha - V_i^{\alpha'}\|.$$

Taking the supremum over x bounds one directed distance. Interchanging the two meshes bounds the other and proves Eq. 32. When vertex colors are included in Y, the identical barycentric argument applies to them, since their maximum difference is at most $\|Y^\alpha - Y^{\alpha'}\|$. □

The extraction assumption requires more than fixed occupancy or fixed scalar-field signs. For example, edge interpolation contains ratios such as $s_i/(s_i - s_j)$; a uniform bound holds, for instance, when the relevant denominators stay away from zero; otherwise a separate argument must control removable degeneracies. Indeed, if $d = s_i - s_j$, $d' = s_i' - s_j'$, $|d|, |d'| \ge \eta > 0$, and $|s_i|, |s_i'| \le B_s$, then

$$\begin{aligned} \left| \frac{s_i}{d} - \frac{s_i'}{d'} \right| &\le \frac{|s_i - s_i'|}{\eta} + \frac{B_s |d - d'|}{\eta^2} \\ &\le \frac{|s_i - s_i'|}{\eta} + \frac{B_s}{\eta^2}\big(|s_i - s_i'| + |s_j - s_j'|\big). \end{aligned} \tag{33}$$

Combining this estimate with bounded, Lipschitz endpoint positions controls the corresponding interpolated vertex positions. More generally, fixed-branch arithmetic with bounded inputs and uniformly nonzero denominators yields finite Lipschitz constants when all constituent operations are Lipschitz on the relevant domains. These conditions must cover every normalization and extraction operation actually used. Equation 31 establishes geometry and color continuity under those conditions; it does not cover changing texture correspondences, visibility changes, or rendered-image continuity.

Assumption 2 (Analytic SS Euler maps)**.** *For the chosen fixed Euler grid, the complete SS fields $z \mapsto v^c(z, t_k)$ are real-analytic on $\mathbb{R}^{d_{\mathrm{SS}}}$ at every query time for $c \in \{\mathrm{src}, \mathrm{tar}\}$, and $\mathcal{D}_{\mathrm{SS}}$ is real-analytic on that space. The initialization is fixed independently of α.*

Proposition 5 (Finitely many SS support events)**.** *Under Assumption 2, there is a finite set $\mathcal{Z} \subset [0, 1]$ such that $\mathcal{P}^\alpha$ is constant on each connected component of $[0, 1] \setminus \mathcal{Z}$.*

Proof. Each Euler update is real-analytic in the pair (α, z). By induction over the finite grid, the terminal SS latent and every decoded logit are real-analytic functions of α on $\mathbb{R}$. For each voxel, the logit minus γ is either identically zero or has only isolated zeros. In the first case the voxel is inactive at every weight under the strict threshold rule. In the second case its zeros in $[0, 1]$ are finite: otherwise compactness would give an accumulation point inside the analytic domain, contradicting the identity theorem. Take $\mathcal{Z}$ to be the union of these finite zero sets over all nonidentically zero logits minus γ. Outside $\mathcal{Z}$, each such function has constant sign on each connected component, proving the result. □

It is sufficient to have analytic extension of all logit paths to an open neighborhood of $[0, 1]$; analyticity only on an open interval with uncontrolled boundary behavior is insufficient. This proposition is an additional fixed-grid Euler result, not a consequence of Lipschitz continuity and not an assertion that every backbone satisfies Assumption 2. It concerns SS support events only. It does not bound mesh connectivity events caused by scalar-field sign changes or other extraction decisions within a fixed support.

An additional bound across supports. The following result concerns decoders that admit a compatible per-voxel attribute layout $\mathcal{A}_{\mathcal{P}}(z) \in \mathbb{R}^{\mathcal{P}\times m_a}$ with the same block dimension m_a for all supports. This is an additional restriction on the attribute representation; fixed-support continuity alone does not provide it. For $\mathcal{Q} \subseteq \mathcal{P}$, the restriction $z|_{\mathcal{Q}}$ retains the rows at those coordinates.

Assumption 3 (Uniform bounds across supports)**.** *Let $\mathfrak{P}$ be the finite set of occurring supports and let*

$$\Omega_{\mathcal{P}} := \{z \in \mathbb{R}^{\mathcal{P}\times C} : \|z_{\boldsymbol{p}}\| \le R \text{ for every } \boldsymbol{p} \in \mathcal{P}\}$$

for some finite R. All relevant Euler iterates on each support lie in $\Omega_{\mathcal{P}}$, and the complete fields are bounded there, uniformly L_{S}-Lipschitz in the state, and satisfy $\|v^{\mathrm{tar}}_{\mathcal{P}}(z,t) - v^{\mathrm{src}}_{\mathcal{P}}(z,t)\| \le M_{\mathrm{S}}$ at every shared SLAT query time, with constants valid for all $\mathcal{P} \in \mathfrak{P}$. Each $\mathcal{A}_{\mathcal{P}}$ is $L_{\mathcal{A}}$-Lipschitz on $\Omega_{\mathcal{P}}$ and obeys $\|\mathcal{A}_{\mathcal{P}}(z)_{\boldsymbol{p}}\| \le A_{\max}$. There are finite $\beta, \beta_{\mathcal{A}} \ge 0$ such that, for every $\mathcal{P}, \mathcal{P}' \in \mathfrak{P}$, $\mathcal{Q} = \mathcal{P} \cap \mathcal{P}'$, and $z \in \Omega_{\mathcal{P}}, z' \in \Omega_{\mathcal{P}'}$ agreeing on $\mathcal{Q}$,

$$\begin{aligned} \|v^{c}_{\mathcal{P}}(z,t)|_{\mathcal{Q}} - v^{c}_{\mathcal{P}'}(z',t)|_{\mathcal{Q}}\| &\le \beta|\mathcal{P}\triangle\mathcal{P}'|, \\ \|\mathcal{A}_{\mathcal{P}}(z)|_{\mathcal{Q}} - \mathcal{A}_{\mathcal{P}'}(z')|_{\mathcal{Q}}\| &\le \beta_{\mathcal{A}}|\mathcal{P}\triangle\mathcal{P}'|, \end{aligned} \tag{34}$$

for both conditions and every SLAT query time. For an ODE version, all velocity bounds, including Eq. 34, must hold for every $t \in [0, 1]$, and the fields must satisfy the continuous-time regularity, existence, and containment requirements of Assumption 1 on these regions.

These are worst-case stability assumptions on the complete fields and decoders. They do not establish spatial locality of attention or a small effect from adding one voxel; without estimates of β and $\beta_{\mathcal{A}}$, those interpretations are not justified.

Proposition 6 (Decoded attribute fields across support changes)**.** *Assume the SS conditions of Corollary 2, Assumption 3, and coordinate-indexed SLAT initialization $E_{\mathcal{P}}$. For two weights, set*

$$\mathcal{Q} := \mathcal{P}^{\alpha} \cap \mathcal{P}^{\alpha'}, \qquad \Delta := |\mathcal{P}^{\alpha}\triangle\mathcal{P}^{\alpha'}| \le n_{\alpha}(\kappa|\alpha - \alpha'|).$$

Define

$$\Phi(L) := \int_0^1 e^{L\tau}\,\mathrm{d}\tau = \begin{cases} (e^L - 1)/L, & L > 0, \\ 1, & L = 0, \end{cases} \quad K_{\mathrm{S}} := M_{\mathrm{S}}\Phi(L_{\mathrm{S}}), \quad K_{\beta} := \beta\Phi(L_{\mathrm{S}}).$$

Then

$$\begin{aligned} \|\bar{X}^{\alpha}_{\mathrm{SLAT}}|_{\mathcal{Q}} - \bar{X}^{\alpha'}_{\mathrm{SLAT}}|_{\mathcal{Q}}\| &\le K_{\mathrm{S}}|\alpha - \alpha'| + K_{\beta}\Delta, \\ \|a^{\alpha}|_{\mathcal{Q}} - a^{\alpha'}|_{\mathcal{Q}}\| &\le L_{\mathcal{A}}K_{\mathrm{S}}|\alpha - \alpha'| + (L_{\mathcal{A}}K_{\beta} + \beta_{\mathcal{A}})\Delta. \end{aligned} \tag{35}$$

For the zero-padded vectors $a^{\alpha}_{\mathrm{ext}} \in \mathbb{R}^{\mathcal{G}\times m_a}$,

$$\begin{aligned} \|a^{\alpha}_{\mathrm{ext}} - a^{\alpha'}_{\mathrm{ext}}\| \le{}& L_{\mathcal{A}}K_{\mathrm{S}}|\alpha - \alpha'| \\ &+ (L_{\mathcal{A}}K_{\beta} + \beta_{\mathcal{A}})\Delta + A_{\max}\sqrt{\Delta}. \end{aligned} \tag{36}$$

In particular, this attribute field is continuous at every weight with $m(\alpha) > 0$, and

$$\begin{aligned} \limsup_{\alpha'\to\alpha} \|a^{\alpha}_{\mathrm{ext}} - a^{\alpha'}_{\mathrm{ext}}\| \le{}& (L_{\mathcal{A}}K_{\beta} + \beta_{\mathcal{A}})n_{\alpha}(0) \\ &+ A_{\max}\sqrt{n_{\alpha}(0)}. \end{aligned} \tag{37}$$

The same bounds hold for ODE outputs under the continuous-time version of Assumption 3 and the corresponding SS assumptions.

Proof. Write $\mathcal{P} = \mathcal{P}^{\alpha}$ and $\mathcal{P}' = \mathcal{P}^{\alpha'}$. At an Euler query, denote the two states by z, z' and their separation on shared coordinates by $e = \|z|_{\mathcal{Q}} - z'|_{\mathcal{Q}}\|$. Construct $\tilde{z} \in \mathbb{R}^{\mathcal{P}' \times C}$ using z on $\mathcal{Q}$ and z' on $\mathcal{P}' \setminus \mathcal{Q}$. It belongs to $\Omega_{\mathcal{P}'}$ because this region is a product of per-voxel balls, and $\|\tilde{z} - z'\| = e$. Adding and subtracting two intermediate predictions gives

$$\begin{aligned}
&\|v^{\alpha}_{\mathcal{P}}(z,t)|_{\mathcal{Q}} - v^{\alpha'}_{\mathcal{P}'}(z',t)|_{\mathcal{Q}}\| \\
&\quad \le \|v^{\alpha}_{\mathcal{P}}(z,t)|_{\mathcal{Q}} - v^{\alpha}_{\mathcal{P}'}(\tilde{z},t)|_{\mathcal{Q}}\| \\
&\qquad + \|v^{\alpha}_{\mathcal{P}'}(\tilde{z},t) - v^{\alpha}_{\mathcal{P}'}(z',t)\| \\
&\qquad + \|v^{\alpha}_{\mathcal{P}'}(z',t) - v^{\alpha'}_{\mathcal{P}'}(z',t)\| \\
&\quad \le \beta\Delta + L_{\mathrm{S}} e + M_{\mathrm{S}}|\alpha - \alpha'|.
\end{aligned}$$

The first term uses Eq. 34 for each condition and convexity; the remaining terms use the within-support bounds. Shared coordinates receive identical initial noise, so the Euler separations satisfy

$$e_0 = 0, \qquad e_{k+1} \le (1 + h_k L_{\mathrm{S}}) e_k + h_k \big[\beta\Delta + M_{\mathrm{S}}|\alpha - \alpha'|\big].$$

Unrolling and using the product bound from Proposition 3 yields $e_N \le \Phi(L_{\mathrm{S}})[\beta\Delta + M_{\mathrm{S}}|\alpha - \alpha'|]$. For continuous trajectories, the same construction is valid at every integration time under the continuous-time assumptions. Their shared-coordinate separation is continuous and satisfies

$$e(t) \le \int_t^1 \big[L_{\mathrm{S}} e(\tau) + \beta\Delta + M_{\mathrm{S}}|\alpha - \alpha'|\big]\, \mathrm{d}\tau;$$

Lemma 2 gives the same terminal bound.

Apply the identical mixed-state construction to the terminal latents and use the decoder part of Eq. 34. The shared attribute separation is at most $\beta_{\mathcal{A}}\Delta + L_{\mathcal{A}} e_N$, proving Eq. 35. The squared norm of the zero-padded difference splits into the shared rows and the Δ rows present in only one support. Each latter row contributes at most $A_{\max}^2$. Thus its norm is at most the shared-row norm plus $A_{\max}\sqrt{\Delta}$, proving Eq. 36. Finally, the finite grid implies that $n_{\alpha}(r) = n_{\alpha}(0)$ for all sufficiently small $r \ge 0$. Combining this fact with Eq. 29 and letting $\alpha' \to \alpha$ proves Eq. 37. If $m(\alpha) > 0$, then $n_{\alpha}(0) = 0$. □

This proposition controls a specified attribute-vector norm, not Hausdorff distance or rendered appearance across support changes. Coordinate-indexed noise is a sufficient coupling for its proof. With other initializations, an additional term proportional to the initial discrepancy on shared coordinates must be retained; mere reseeding does not make that discrepancy zero.

Remark 4 (Support events, mesh events, and topology). SS occupancy threshold contacts and mesh extraction decisions are distinct events. Corollary 2 constrains the former, while Proposition 4 applies only where the latter remain on one regular branch. Mesh connectivity can change even on a fixed SS support. If the endpoint assets have different topological invariants, some topological change is necessary along any path connecting them, but such a change does not by itself imply a discontinuity in a geometric distance such as Hausdorff distance. We do not guarantee topology preservation, nor do we identify all mesh topology events with SS threshold contacts.

The constants in these results are not calibrated and may be large. The conclusions provide conditional links from sampling to geometry and color, and bounds on selected discrete effects; they do not establish perceptual smoothness or guarantee small jumps at extraction boundaries.

B.8 Endpoint recovery in the two-stage pipeline

Endpoint recovery compares a morphing endpoint with a model generation under the corresponding original condition and matched sampling inputs. Fix the SS initialization ϵ_{SS}. For $c \in \{\mathrm{src}, \mathrm{tar}\}$, let the reference generation $\mathcal{O}^c$ use that initialization, SLAT noise $E_{\mathcal{P}^c}$ on its own support, and the same complete update fields, grids, guidance and rescaling settings, thresholding, coordinate ordering, denormalization, and decoders as the morphing endpoint. All evaluations must be well defined, and every additional stage input must agree between the endpoint run and its reference.

Proposition 7 (Recovery of matched two-stage reference generations)**.** *Under the matched-input conditions above and deterministic sampling and postprocessing, the Euler outputs satisfy, in exact*

arithmetic,

$$\begin{aligned} \bar{X}^0_{\mathrm{SS}} &= \bar{X}^{\mathrm{src}}_{\mathrm{SS}}, \qquad \mathcal{P}^0 = \mathcal{P}^{\mathrm{src}}, \\ \bar{X}^0_{\mathrm{SLAT}} &= \bar{X}^{\mathrm{src}}_{\mathrm{SLAT}}, \quad \mathcal{O}^0 = \mathcal{O}^{\mathrm{src}}, \end{aligned} \tag{38}$$

and the corresponding identities hold at $\alpha = 1$ for the target. For exact ODE sampling, the same conclusion holds when the stagewise solutions exist and are unique, as ensured by the corresponding assumptions of Proposition 1.

Moreover, suppose $\epsilon_{\mathrm{SS}} \sim \mathcal{N}(0, I)$ and the coordinate-indexed vectors $\epsilon_{\mathrm{SLAT}}(\boldsymbol{p}) \sim \mathcal{N}(0, I_C)$ are independent across coordinates and independent of ϵ_{SS}. For each condition, this reference generation has the same marginal distribution as the otherwise identical pipeline that samples an independent standard Gaussian SLAT matrix conditional on its support.

Proof. At $\alpha = 0$, the complete SS interpolation reduces pointwise to the source field. The initial SS states and Euler updates agree, so induction over the shared grid gives $\bar{X}^0_{\mathrm{SS}} = \bar{X}^{\mathrm{src}}_{\mathrm{SS}}$. The deterministic SS decoder and threshold then give $\mathcal{P}^0 = \mathcal{P}^{\mathrm{src}}$. Since the coordinate order and coordinate-indexed noise agree, the SLAT initial tensors coincide. All other SLAT inputs agree by assumption, and its complete interpolation also reduces to the source field. A second induction gives $\bar{X}^0_{\mathrm{SLAT}} = \bar{X}^{\mathrm{src}}_{\mathrm{SLAT}}$. Deterministic denormalization and decoding produce equal final assets. The target argument is identical. For ODE sampling, uniqueness replaces the induction in each stage. Only equality of intermediate inputs is required; their dependence on α need not be continuous.

For the distributional statement, the support is determined by the SS noise and the fixed condition, and is independent of the SLAT noise field. Conditional on ϵ_{SS}, selecting distinct coordinates of that field yields

$$E_{\mathcal{P}^c} \mid \epsilon_{\mathrm{SS}} \sim \mathcal{N}(0, I_{|\mathcal{P}^c|C}).$$

This is the same conditional law as an independently sampled SLAT noise matrix of that size. Applying the same deterministic subsequent maps proves equality of the output distributions. □

Equality in distribution does not identify the output of an unmodified implementation under the same integer seed: coordinate indexing and sequential matrix sampling can assign different values to the same coordinate. The exact endpoint identities concern matched realized noise tensors. Coordinate indexing is sufficient but not necessary for them; matching each endpoint to its reference is enough, regardless of how noises at intermediate weights are coupled. The source and target supports may differ. The same stagewise induction extends to more than two stages when each corresponding input, complete update rule, and postprocessing operation matches.

TRELLIS.2 and floating-point implementations. TRELLIS.2 is used only for qualitative demonstrations, and no quantitative result depends on it. For the seed-reset TRELLIS.2 adaptation, endpoint recovery remains conditional on matching each endpoint run to its reference run's stagewise random inputs and deterministic operations. Resetting a seed alone does not prove this condition: random-number consumption, tensor shapes, coordinate order, auxiliary inputs, and sampler settings must also match. Such matching is a property of the sampler implementation rather than of the pretrained model: whenever a sampler accepts its stage initializations as inputs, coordinate-keyed noise can be supplied without retraining. The absence of coordinate-wise noise correspondence between different intermediate supports does not itself invalidate the endpoint argument, but it prevents invoking the cross-support continuity result without an additional coupling argument. The identities are exact-arithmetic statements; bitwise equality additionally requires the same arithmetic and evaluation context, including batching and kernel choices. This analysis does not substitute for checking these implementation conditions.

Remark 5 (Model-generated endpoints and supplied assets). The endpoint identities refer to generations conditioned on the two input images, including when those images depict supplied 3D assets. They do not imply exact recovery of those external assets. Such recovery is a separate reconstruction problem, potentially involving inversion or optimization, and is also limited by the decoder's representational range. DINO-I measures endpoint semantic alignment rather than exact geometric reconstruction.

Remark 6 (Shared additive noise in a fixed latent space). Fix a latent space, a finite grid, and a shared initial state. Consider the discrete sampler

$$\bar{Z}^{\alpha}_{t_{k+1}} = c_k \bar{Z}^{\alpha}_{t_k} - g_k v^{\alpha}(\bar{Z}^{\alpha}_{t_k}, t_k) + \sigma_k \xi_k, \tag{39}$$

where the finite scalar coefficients c_k, g_k, σ_k are fixed independently of the state, weight, and condition, and the realized vectors ξ_k are shared across weights and with the corresponding reference generations. Euler sampling has $c_k = 1$, $g_k = h_k$, and $\sigma_k = 0$. Conditional on this noise sequence, identical endpoint fields and initial states give identical source or target runs by induction.

Suppose all iterates for the fixed realization remain in a common region where the complete fields have uniform state Lipschitz constant L and difference bound M. Writing $e_k = \|\bar{Z}^{\alpha}_{t_k} - \bar{Z}^{\alpha'}_{t_k}\|$ and $d_k = \|v^{\mathrm{tar}}(\bar{Z}^{\alpha'}_{t_k}, t_k) - v^{\mathrm{src}}(\bar{Z}^{\alpha'}_{t_k}, t_k)\|$, subtraction of the updates cancels the shared additive noise and gives

$$e_0 = 0, \qquad e_{k+1} \le (|c_k| + |g_k|L)e_k + |g_k|\,|\alpha - \alpha'|d_k.$$

Unrolling and using $d_k \le M$ yields the pathwise bound

$$\|\bar{X}^{\alpha} - \bar{X}^{\alpha'}\| \le M|\alpha - \alpha'| \sum_{k=0}^{N-1} |g_k| \prod_{j=k+1}^{N-1} (|c_j| + |g_j|L), \tag{40}$$

where an empty product is one and N here denotes the number of sampling steps. The Euler choice reduces to the bound in Proposition 3. Independent noise draws introduce an extra discrepancy and provide no general pathwise continuity guarantee. This result applies to the stated discrete additive-noise family; shared randomness alone does not establish analogous bounds for arbitrary stochastic samplers, state-dependent diffusion, adaptive branching, or changing latent supports.