# Predictive 4D Generation with Latent State Machines

**Yue Gao**[1,*] **Chen Geng**[1,*] **Hanzhe Hu**[2] **Fangyun Wei**[3]
**Jianfeng Xiang**[3] **Jiaolong Yang**[3] **Jiajun Wu**[1]

[1]Stanford University [2]Carnegie Mellon University [3]Microsoft Research




Figure 1: Given a single image of a flower at any growth stage, our model generates the full 4D growth sequence from sprout to full bloom. Our method also recovers past growth states from the same image. By composing the generated object states, we obtain high-fidelity long-horizon 4D dynamic scenes, as shown in the animated results on our project page.

## Abstract

What is the right representation for 4D (3D + time) generation of graphics assets conditioned on their current states? In this setting, an ideal 4D representation should support flexible geometric structure, empower long-horizon modeling, and utilize training data efficiently. We study this problem by considering a sufficiently challenging setting to stress test those properties: learning the generative distribution of the entire 4D lifespan of a plant, conditioned on a snapshot of its growth. We curate a large-scale dataset and benchmark for this setting, on which most existing 4D representations and accompanying generative models fail to deliver promising results. We analyze the pitfalls of existing representations and point out that the capacity of existing models is bounded by their representations: they are mostly augmenting a temporal dimension to an existing geometric representation, without a semantically-meaningful low-dimensional state representation. Instead, we argue for a framework for designing 4D representations by separately considering 3D state representations and temporal state transitions. We propose an instantiation of this framework using latent spaces of 3D foundation models and a transformer-based state transition model, effectively forming a finite state machine on a latent 3D space. Therefore, we name this instantiation a *latent state machine* and compare it against existing generative models on the proposed benchmark. Our evaluation

*Equal contribution

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

shows that our model achieves state-of-the-art performance in expressiveness, data efficiency, and rendering quality, surpassing existing methods by a large margin. Project page: `https://yuegao.me/lsm`.

# 1 Introduction

Looking at a bud just sprouting from a pot in Fig. 1, how would its 3D appearance evolve, grow, and wither? Such curiosity stems from humans' desire to understand our dynamic natural world, and it fuels the need for machines to model the generative distribution of a temporally evolving graphics asset of an object conditioned on a single observed snapshot, a task we define as *predictive 4D generation* in this paper.

Until recently, this task was intractable for both the machine learning and graphics communities, primarily because long-horizon 4D data is scarce. Yet, with the emergence of large-scale 4D datasets [1–3] and the viability of distillation from 2D foundation models [4–6], recent work [7, 8] attempts to tackle adjacent tasks with similar structures. However, most of these are limited to relatively contrived settings, such as a fixed geometric topology, or generation of a short temporal window with a limited number of frames.

Assuming abundant 4D data, is it possible to tackle predictive 4D generation with minimal assumptions, and to generate 4D graphics assets of arbitrary geometric structure and temporal horizon, conditioned on a single snapshot? This paper studies the problem by considering a potentially extreme case of such settings: generating the lifespan of a flower given only one snapshot as input. This setting poses unique challenges for model design, because flower growth involves flexible topology changes over a long horizon. This setting also captures common traits of natural dynamic processes, because challenging natural dynamics are both topology-varying (*e.g.*, candle burning) and long-horizon (*e.g.*, animal evolution).

To meet the data requirements for the chosen natural phenomenon and to make it feasible to benchmark different 4D generative pipelines, we build a scalable 4D data generation pipeline. Specifically, we manually curate procedural models of the flower growth process and design a data generation pipeline that produces a benchmark of 120k rollouts for generative models to learn from.

Our study on this benchmark shows that existing 4D generative models do not scale well to this challenging setting. We observe that the failure stems from how these pipelines define *4D representations*. Most existing work considers a 4D representation as a 3D geometric representation augmented by a temporal dimension, yielding representations like deformable meshes [9–11], 3DGS or NeRF with deformation fields [6, 12, 13], and temporally evolving implicit fields [4, 5]. This "temporal-augmenting" choice also implicitly forces models to generate fixed-length 4D sequences with a limited number of frames, rather than long-horizon dynamic rollouts. Furthermore, it models the generative dynamics directly in the high-dimensional geometric space, wasting representation capacity on generative modeling.

Motivated by this observation, we argue for a representation-centric design principle in 4D generative modeling. Inspired by the literature on learning-based dynamic models in robotics [14] and control [15], we argue that a 4D representation in generative modeling should be defined as the combination of *3D state representation* and *temporal state transitions*, where the first term defines a state space in which dynamic modeling occurs, and the latter term defines the predictive model of the next state given the current state. Under this umbrella, most existing work parameterizes the 3D state representation using a raw, uncompressed geometric representation, and uses a fixed array of deformation fields to model state transitions.

Instead, we give another instantiation under this framework. We argue that the state space of a 4D representation should be compact, semantic, and structured. We therefore leverage the semantic features of a 3D foundation model [16] to instantiate our state representation. We design our state transitions with minimal inductive bias in mind, and we therefore adopt a transformer-based architecture to model the state transition process from data. This design echoes the term "finite state machine" from the theory-of-computation literature, and we therefore name our model the *Latent State Machine* (LSM).

Our model achieves state-of-the-art performance on the proposed benchmark. With a state design that does not assume topology invariance, it captures the drastic structural changes of the flower

Table 1: Categorization of 4D generation methods by *3D state representation* and *temporal state transition*. Existing methods augment a raw 3D representation (mesh or 3D Gaussians) with a temporal mechanism, locking topology to the initial reconstruction and the rollout to a short horizon. LSM decouples the two: the state is the structured latent of a 3D foundation model, and a transformer-based transition predicts the next state, enabling topology changes and arbitrary-horizon rollouts.

| Method | State Representation | Temporal Modeling | Topology Change | Long Horizon |
|---|---|---|---|---|
| Free4D [17] | Explicit 3D Gaussians | Video diffusion models | ✗ | ✗ |
| DreamGaussian4D [18] | Explicit 3D Gaussians | Video diffusion models | ✗ | ✗ |
| AnimateAnyMesh [7] | Explicit mesh vertices | Vertex displacement | ✗ | ✗ |
| ODE-GS [19] | Explicit 3D Gaussians | Neural ODE | ✗ | ✓ |
| GaussianPrediction [20] | Explicit 3D Gaussians | Keypoint-GCN motion | ✗ | ✓ |
| LSM (Ours) | Implicit structured 3D latent | Latent state transition | ✓ | ✓ |

growth process. It is scalable and efficient, owing to its semantically rich latent state representation. It predicts rollouts of hundreds of frames, significantly outperforming prior work in 4D generation.

## 2 Related Work

**Geometry-Inherited 4D Representations.** The choice of representation is the first step in a generative modeling problem. While 3D representation is well-studied and has converged into a small set of geometric primitives such as meshes, radiance fields [21], and Gaussian primitives [22], the study of 4D representations remains relatively underexplored. Most existing works approach 4D generation by simply augmenting an existing 3D representation with a temporal dimension, yielding deformable meshes [9–11], radiance fields with a deformation field [13, 23], 3D Gaussian primitives with deformation [6, 12, 18, 24–35], or temporally changing implicit fields [4, 5, 36–39]. While effective in specific settings, these methods treat the temporal dimension as interchangeable with the three spatial dimensions, leading to wasted representational power for dynamic modeling in a high-dimensional geometric space. Moreover, by augmenting an existing geometric representation, these methods often assume that the underlying 3D object's topology remains fixed, which prevents them from modeling complex dynamic phenomena. Instead, we propose to model 4D processes with a learned state representation and transition function, addressing the issues above.

**Low-Dimensional 4D Representations.** More broadly, ours is not the first work to formulate 4D generation in a low-dimensional space. In domains with strong category priors, it is in fact the default choice. Human motion generation, for example, has long worked in the pose space of a parametric body skeleton [40–42]. The same is true of face [43–49] and animal [50] modeling. However, these methods all assume a category-specific template, and thus do not scale to the category-agnostic setting. Instead, we learn a state representation in the category-agnostic setting, which extends this approach to general 4D modeling. Recent work [51–54] has also attempted to learn a state-based representation from data, but focuses on the representation-learning side, not generative modeling.

**4D Generation.** Our work falls under the broad umbrella of 4D generation. Most existing work tackle the problem from two different perspectives. One dominant line of work [4–6, 9–13, 18, 23–26, 28–36, 38, 39] splits the pipeline into 2D video generation followed by reconstruction, naturally limiting themselves to modeling fixed frame length 4D sequences. While effective in specific settings, they suffer from slow optimization or struggle with occlusion. Our work belongs to another emerging paradigm of directly modeling the 4D content distribution [7, 8, 55, 56], and achieves the best balance between expressivity, generation speed, and scalability. SS4D [57] also builds on a 3D-native latent, but treats time as an axis of a structured spacetime latent, so the sequence length is fixed and generation is conditioned on an input video. The concurrent 4DSTAR [58] is also autoregressive, but it reconstructs 4D from monocular observations with a fixed set of deformable Gaussians and predicts only forward in time. We instead keep the latent purely 3D and model time as a learned transition, decoding a new 3D state at each step in both directions. To our knowledge, our method is the first to use the generative latent space of a 3D foundation model as the state space of a learned dynamical system.

**Learning-Based Dynamic Models.** In machine-learning terminology, 4D generation can be viewed as learning a *dynamic model* for graphics assets. We therefore draw on prior work in this area.

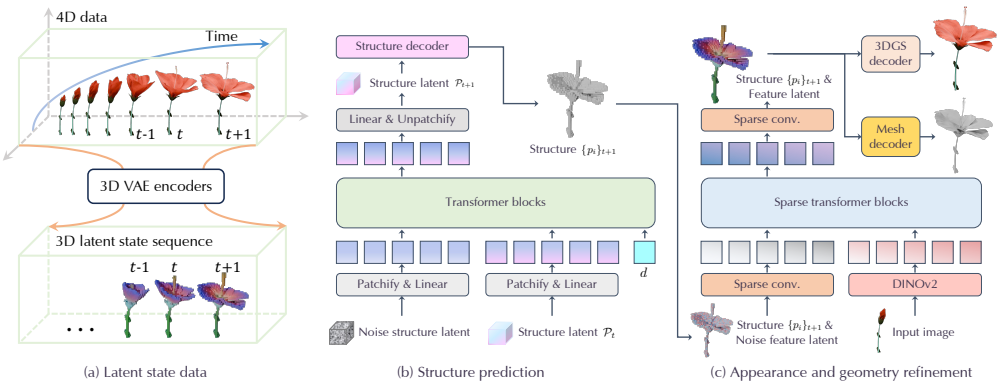



Figure 2: **Method overview. (a)** TRELLIS 3D VAE encoders map a 4D sequence to structured latent states. **(b)** A *state-transition* flow transformer denoises the next sparse-structure latent $\mathcal{P}_{t+1}$ from $\mathcal{P}_t$, with a learnable direction embedding $d$ for forward or backward rollout; a structure decoder yields the voxel set $\{p_i\}_{t+1}$. **(c)** A sparse flow transformer predicts per-voxel features conditioned on DINOv2 image features, and decoders produce meshes and 3D Gaussians [22].

We organize this prior work along two axes: how the state is defined, and how the state evolves. For state definition, some methods operate directly in the raw geometric space [59–61], while others [14, 15, 62–64] first learn a compressed representation. Our method falls in the second category, using representations from pretrained 3D foundation models as the state. For state transition, prior work uses end-to-end models [14, 59–61, 65–69], physics-based structures [70–75], or neural ODEs [76–79]. We assume no physical priors and learn the state transition end-to-end. Most work on learning-based dynamic models pays little attention to rendering fidelity, whereas we extend dynamic modeling into the 4D generation of graphics assets, where rendering quality is a primary concern.

# 3 Methodology

## 3.1 Task

This paper studies *predictive 4D generation*, an extended variant of 4D generation that explicitly conditions on one given state of an object and predicts a long-horizon 4D trajectory that maintains consistency with this input. Formally, given an input state of an object $o$ (*e.g.*, a flower) parameterized by its RGB image observation $\mathbf{I}$ at timestamp $t_0$, our goal is to generate a 4D trajectory $\mathcal{O}$:

$$\mathcal{O} = \{O_t \mid t \in [t_{\min}, t_{\max}]\}, \tag{1}$$

where each $O_t$ represents a renderable graphics asset encompassing the complete 3D geometry and appearance of the object at time $t$. We do not assume any shared geometric structures among timestamps, and $t_{\min}$ and $t_{\max}$ can be given arbitrarily, yielding a varying total length of trajectory.

This task setting generalizes 4D dynamic process modeling and captures several recurring challenges of natural dynamic phenomena. First, it imposes no fixed frame count, requiring models to handle 4D sequences of arbitrary length. Moreover, it does not assume a shared geometric structure (*e.g.*, shared topology), since natural phenomena such as flower growth and candle burning routinely add and remove structural elements. Finally, it requires methods to capture large-magnitude motion, a long-standing challenge for the 4D modeling community.

## 3.2 Dataset and Benchmark

To evaluate existing methods in this task setting, we need a benchmark with enough training data. However, 4D data is much scarcer than other data sources [3], and existing datasets [1, 2] mostly have a limited temporal horizon and topological variation.

We therefore select plant growth, a natural long-horizon phenomenon with many publicly available procedural-generation assets. We construct five kinds of flower-growth animations in Blender (rose, lily, hibiscus, daisy, and dahlia) and randomize the parameters of the procedural programs, obtaining

a dataset of 120k 4D dynamic rollouts of high-quality graphics assets (Sec. A). This dataset and benchmark provide abundant data for generative models to learn from, and have sufficient complexity to test the upper bound of these models.

### 3.3 4D Representation and Our Instantiation

We evaluate existing 4D generative methods [7, 17–20] on this benchmark. As shown in Sec. 4, none of these methods successfully handle the challenges of our setting.

Analyzing their model design, we trace these failures to representation design, the first step of any generative modeling problem. Most existing methods treat 4D representations as temporally augmented 3D representations, omitting the low-dimensional and semantic nature of the spatial content.

Instead, we decompose 4D representations into two components: the 3D state representation and the temporal state transition. Formally, a 3D state representation is a tuple $(P, E, D)$, where $P$ is a latent state space, $\mathcal{P}_t \in P$ is the implicit state at frame $t$, the encoder $E$ maps an explicit geometric state to its latent code $\mathcal{P}_t = E(\mathcal{O}_t)$, and the decoder $D$ inverts this map as $\mathcal{O}_t = D(\mathcal{P}_t)$.

Temporal state transitions form a bidirectional dynamic process over the latent state space. Specifically, this process comprises two prediction functions, $\mathcal{F}_{\text{fwd}}$ and $\mathcal{F}_{\text{bwd}}$. The forward function $\mathcal{F}_{\text{fwd}}$ models the future state $\mathcal{P}_{t+n} \sim \mathcal{F}_{\text{fwd}}(\cdot \mid \mathcal{P}_t, \cdots, \mathcal{P}_{t+n-1})$. The backward function $\mathcal{F}_{\text{bwd}}$ models the previous state $\mathcal{P}_{t-n} \sim \mathcal{F}_{\text{bwd}}(\cdot \mid \mathcal{P}_t, \cdots, \mathcal{P}_{t-n+1})$.

Under this formulation, existing representations decompose into these two components, as illustrated in Tab. 1. For example, the deformable Gaussian Splatting representation [18] takes Gaussian primitives directly as the state space (so $E$ and $D$ are identity mappings) and models state transitions through a fixed array of deterministic deformations. However, this geometry-inherited instantiation has limited expressiveness for topology and fails on long-horizon generation. As a result, it performs poorly on our benchmark.

We propose an alternative instantiation of this formulation. As illustrated in Fig. 2, we train a variational autoencoder [80] to encode the 3D state space into a semantically structured latent space (Sec. 3.4) and model temporal state transitions with a scalable, flexible diffusion process (Sec. 3.5). This design forms a finite state machine over an implicit latent space, so we name our model *Latent State Machine* (LSM).

### 3.4 Latent State Representation

The 3D state representation in an ideal 4D representation needs to be flexible, compact, and semantically meaningful. Recent 3D foundation generative models [16] are trained on a large amount of data and therefore contain a semantically meaningful space of 3D geometric representations. Inspired by this observation, we base our state representation on the pre-trained latent representation of TRELLIS [16] and show that such a space provides a competitive design for the 3D state representation in predictive 4D generation.

Following TRELLIS, our encoder $E$ encodes a 3D object $\mathcal{O}$ into $\mathcal{P}$:

$$\mathcal{P} = \{(p_i, z_i)\}_{i=1}^{L}, P = \mathbb{R}^{(3+C)\times L}, \tag{2}$$

where $p_i \in \mathbb{R}^3$ denotes the spatial position of the $i$-th active voxel, and $z_i \in \mathbb{R}^C$ is its corresponding latent feature with dimension $C$. Here, $L$ is the total number of active voxels.

During decoding, our decoder $D$ decodes $\mathcal{P}$ into meshes for graphics rendering. We finetune from the pretrained weights of TRELLIS on the provided training data in our benchmark to obtain $E$ and $D$.

### 3.5 Latent State Transition

To capture expressive dynamics over long horizons, we model the state transition as an autoregressive process. Without loss of generality, we describe the forward process $\mathcal{F}_{\text{fwd}}$ in this section.

Formally, $\mathcal{F}_{\text{fwd}}$ models the state transition as a conditional generation problem. We instantiate it as a one-step transition model conditioned on the previous state $\mathcal{P}_{t+n-1}$ and on the input frame $\mathcal{P}_t$ (the latter motivated below as a "global anchor" against drift). We use a conditional flow transformer

Table 2: Quantitative comparison of reconstruction metrics on our collected dataset across different methods. "CD" denotes Chamfer distance.

| | Geometry | | Appearance | | |
|---|---|---|---|---|---|
| | Voxel IoU ↑ | Mesh CD ↓ | SSIM ↑ | PSNR ↑ | LPIPS ↓ |
| AnimateAnyMesh [7] | - | 0.03267 | 0.8453 | 20.85 | 0.1233 |
| Free4D [17] | 0.0365 | 0.09543 | 0.7576 | 16.21 | 0.1657 |
| ODE-GS [19] | 0.0965 | 0.08457 | 0.8601 | 21.07 | 0.0911 |
| GaussianPrediction [20] | 0.1169 | 0.08261 | 0.8525 | 21.85 | 0.0821 |
| LSM (Ours) | **0.8987** | **0.01412** | **0.9553** | **31.13** | **0.0407** |

Table 3: Ablation studies of the main components. "GA" denotes the Global Anchor. "DF" stands for diffusion forcing [82].

| | Geometry | | Appearance | | |
|---|---|---|---|---|---|
| | Voxel IoU ↑ | Mesh CD ↓ | SSIM ↑ | PSNR ↑ | LPIPS ↓ |
| w/o DF | 0.1963 | 0.07618 | 0.7961 | 14.28 | 0.2843 |
| w/o GA | 0.6964 | 0.01907 | 0.8275 | 26.43 | 0.1291 |
| LSM (Full) | **0.8987** | **0.01412** | **0.9553** | **31.13** | **0.0407** |

structurally similar to TRELLIS [16] to model this distribution. To specify the temporal direction (forward or backward), we add a learnable embedding as an extra condition. We train our models with classical flow-matching objectives [81].

**Diffusion forcing for long-horizon prediction.** We notice that directly using the above-mentioned model will yield drifting trajectories when predicting long-horizon trajectories. Inspired by literature in video modeling, we adopt Diffusion forcing [82] during training to mitigate this problem. Specifically, we apply another noise level to $\mathcal{P}_{t+n-1}$, while we denoise $\mathcal{P}_{t+n}$.

**Global anchor.** We notice that modeling $\mathcal{F}_{\text{fwd}}$ as a purely Markovian process $\mathcal{P}_{t+n} \sim \mathcal{F}_{\text{fwd}}(\cdot \mid \mathcal{P}_{t+n-1})$ will lead to drifting in the generated textures in long-horizon sequences. To fix this, we propose to add the input frame $\mathcal{P}_t$ to the conditioning, a technique we name "global anchor". Our ablation study shows the importance of this design.

# 4 Evaluations

## 4.1 Experiment Setup

**Datasets.** As described in Sec. 3.2, we construct our dataset to contain five types of plant growth process. The training split contains 120k rollouts of diverse plant types. The test split is generated by the same procedural pipeline and contain $450$ unseen rollouts ($\sim$ 42k frames). Test rollouts retain the full lifespan length of their species and are never observed during training of any component. Please refer to Sec. A for more details.

**Baselines.** We compare against representative 4D generation methods that differ in their *state representation* and *temporal state transition* mechanism (Tab. 1). Free4D [17] and DreamGaussian4D (DG4D) [18] drive per-asset dynamic 3D Gaussians with a generated video, and AnimateAnyMesh [7] predicts an animated mesh from a single static input. ODE-GS [19] and GaussianPrediction [20] are dynamics-based baselines built on dynamic 3DGS [22]: ODE-GS extrapolates per-Gaussian state with neural ODEs, while GaussianPrediction predicts keypoint motion with GCNs. We train all models on the training set of our benchmark (Sec. A).

**Metrics.** We compare how the outputs of each method align with the desired trajectory as specified in the dataset both visually and geometrically. For *geometry quality*, we report Chamfer Distance (CD) on the extracted mesh and Voxel IoU on the predicted sparse structure. For 3DGS-based methods, we compute both geometry metrics by treating 3DGS centers as geometry points. For *appearance quality*, we report PSNR, SSIM [83], and LPIPS [84] on multi-view renders.

We report results across both forward (growth) and reverse (de-growth) settings in Tab. 2 and Fig. 3. LSM outperforms baselines on every metric compared, demonstrating the effectiveness of our design.

To analyze how design choices of different representation affect the performance of the generative pipeline, we ask the following two questions along the two design axes: whether the state representation should be explicit or implicit, and how the temporal transition should be modeled. We detail those two questions in the following sections.

## 4.2 3D State Representation: Explicit or Implicit?

Based on our results in Tab. 2 and Fig. 3, we analyze different design choices of 3D state representation for different methods. The dominant design in existing 4D generative models is to use geometry-

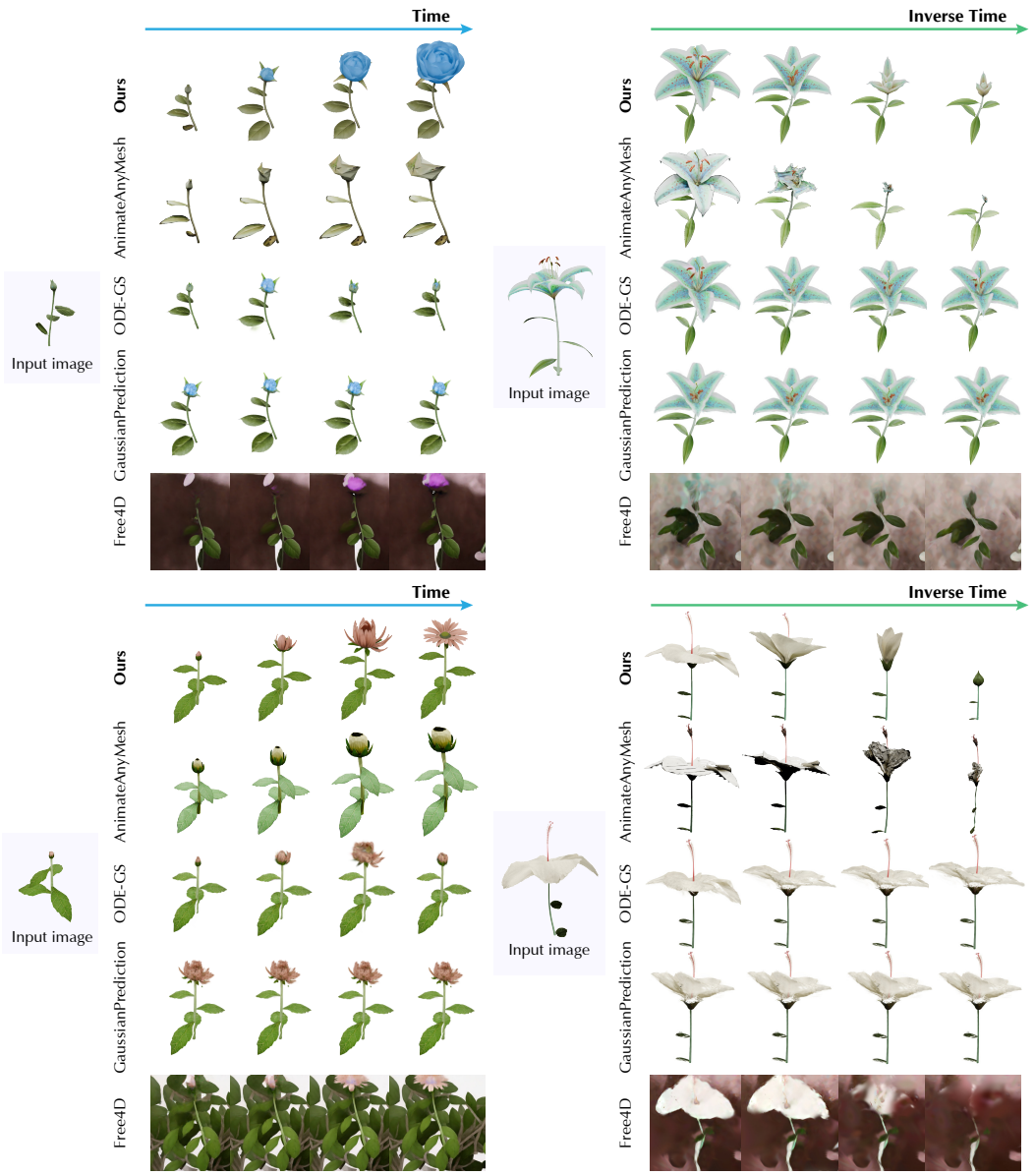



Figure 3: **Qualitative comparison of different 4D representations for 4D generation.** Existing methods struggle to handle topology changes and long temporal horizon, while our model achieves the best generation fidelity.

inherited representations for state representation, such as explicitly deformed meshes or Gaussian Splattings, and they have different failure modes.

**Failure mode of deformable meshes.** Methods like AnimateAnyMesh deform a mesh with a fixed topology, so they cannot insert or remove primitives when plant growth adds new petals, leaves, and stem segments.

**Failure modes of deformable GS.** ODE-GS and GaussianPrediction freeze the set of Gaussian primitives at initialization and predict only their state evolution. ODE-GS degrades significantly outside the observed time window, and GaussianPrediction produces near-canonical outputs across all frames (Sec. C.3).

In contrast, LSM decodes a fresh latent state at every step, so the geometry evolves with time. This decoupling of state representation from state transition is what enables our model to grow new petals,

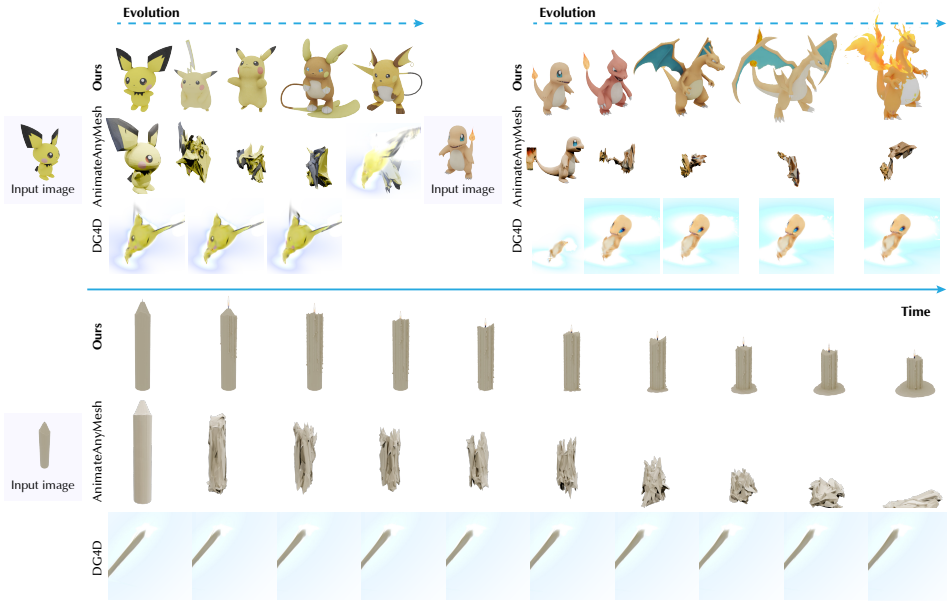



Figure 4: **Generalizing beyond plants**. Our problem setup and representation is general and can apply to other complicated dynamic phenomena like animal evolution or candle burning.

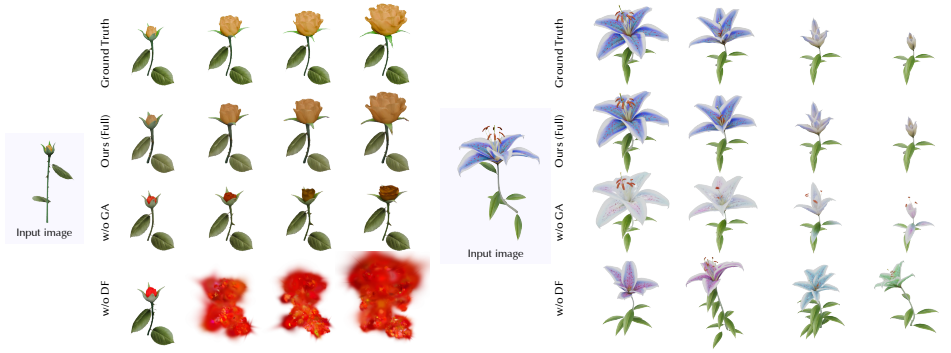



Figure 5: **Ablation studies of different design choices.** "GA" denotes the Global Anchor. "DF" stands for diffusion forcing [82].

drop old leaves, and otherwise change topology over a 120-frame rollout while preserving identity from the input image.

### 4.3 Temporal State Transition: Fixed-frame modeling or Latent State Transition?

The second axis is how temporal state transition is modeled. Video-diffusion-driven 4D pipelines lift a generated 2D video into a per-asset 4D representation; their rollout length is bounded by the I2V model's training horizon, and any 2D hallucination is fixed into the lifted 3D content. LSM instead models time directly in the structured 3D latent space as a Markovian state transition, so rollout length is bounded only by the autoregressive horizon, not by the training length of any single 2D video model.

As a result, Free4D and DreamGaussian4D are bounded by the rollout horizon of the I2V model, and Free4D struggles to recover plausible geometry (Sec. C.3). LSM instead models the temporal transition with autoregressive generation, yielding long-horizon sequences without drifting.

**Video-to-4D methods and long-horizon rollouts.** LSM also outperforms the video-to-4D reconstruction methods SV4D [13], SV4D 2.0 [23], and GVFDiffusion [8] across all metrics (Sec. C.4). On sequences of up to 360 frames, performance degrades gradually but does not collapse, and models trained on only part of each lifespan extrapolate beyond their training horizon (Sec. C.5).

Table 4: Quantitative results on candle burning and animal evolution.

| | Candle burning | | | Animal evolution | | |
|---|---|---|---|---|---|---|
| | IoU ↑ | CD ↓ | PSNR ↑ | IoU ↑ | CD ↓ | PSNR ↑ |
| DG4D [18] | 0.6238 | 0.1352 | 24.15 | 0.6874 | 0.154 | 22.38 |
| AnimateAnyMesh [7] | 0.164 | 0.1125 | 23.52 | 0.158 | 0.1273 | 21.75 |
| LSM (Ours) | **0.9108** | **0.0129** | **33.28** | **0.9464** | **0.0093** | **34.31** |

Table 5: Generalization beyond the procedural training distribution (forward / backward).

| | IoU ↑ | CD ↓ | PSNR ↑ |
|---|---|---|---|
| *Non-procedural datasets* | | | |
| DyMesh | 0.892 / 0.892 | 0.0020 / 0.0079 | 33.5 / 33.1 |
| ObjXL4D | 0.988 / 0.987 | 0.0062 / 0.0067 | 34.7 / 34.9 |
| In-distribution | 0.909 / 0.905 | 0.0140 / 0.0139 | 31.7 / 31.7 |
| *Beyond the training range* | | | |
| Petal count | 0.900 / 0.899 | 0.0146 / 0.0148 | 31.2 / 31.3 |
| Size scale | 0.901 / 0.901 | 0.0141 / 0.0144 | 31.0 / 31.0 |
| Color/material | 0.909 / 0.904 | 0.0140 / 0.0141 | 31.6 / 31.6 |
| All parameters | 0.898 / 0.894 | 0.0152 / 0.0149 | 30.9 / 30.8 |

### 4.4 Generalization

Our problem formulation and training recipe are universal and generalizable, and can be applied to diverse domains without domain-specific redesign. We show such results of fitting different 4D representations to other phenomena in Fig. 4, which comprises the burning lifespan of a candle and the evolutionary stages of animals. Other baselines exhibit similar issues as in the plant growth experiment. As shown in Tab. 4, LSM substantially outperforms both baselines on all metrics.

With the same recipe, LSM also performs well on the non-procedural datasets DyMesh [7] and ObjXL4D [2, 30], so the formulation is not specific to our procedural benchmark. It also remains competitive on procedural parameters outside the training ranges (Tab. 5). Evaluated zero-shot on sunflower, a held-out sixth species, the model achieves an IoU of $0.6295$ and a Chamfer distance of $0.0914$. A single model trained jointly on all five flower species and both temporal directions performs comparably to the per-species models (Sec. C.6).

### 4.5 Real Image to 4D

Although our model is trained on synthetic data, it can generalize to real images successfully. As shown in Figure 6, our method produces plausible 4D growth sequences from real inputs that are not seen during training. This demonstrates that the model learns transferable representations of plant structure and dynamics rather than overfitting to the synthetic domain.

### 4.6 Ablation Studies

We analyze the contributions of two key components in our framework: (1) **Diffusion Forcing (DF)** [82], used to train the latent state machine, and (2) **Global Anchor (GA)**, which enforces consistent structure-latent generation across the entire growth sequence.

**Effect of diffusion forcing.** As shown in Figure 5 and Table 3, removing diffusion forcing leads to severe degradation in both geometric and appearance quality (e.g., IoU $0.1963$ vs. $0.8987$, PSNR $14.28$ vs. $31.13$), indicating the critical role of DF in stable autoregressive generation. DF mitigates




Figure 6: Qualitative results on two real samples.

Table 6: Ablation study of transition design variants on rose.

| | Forward | | Backward | |
|---|---|---|---|---|
| | IoU ↑ | PSNR ↑ | IoU ↑ | PSNR ↑ |
| Direction-specific models | 0.9106 | 32.47 | 0.9153 | 32.80 |
| ControlNet-style branch | 0.7858 | 28.45 | 0.7631 | 28.62 |
| Residual propagation | 0.6108 | 27.13 | 0.6196 | 26.96 |
| 2-state history cache | 0.8877 | 30.60 | 0.8814 | 31.32 |
| 4-state history cache | 0.8906 | 30.11 | 0.8905 | 31.79 |
| Raw occupancy-grid state | 0.6906 | 24.11 | 0.6146 | 24.58 |
| LSM (Ours) | **0.9198** | **32.98** | **0.9191** | **32.58** |

Table 7: Analysis of contribution of TRELLIS pretraining.

| | From scratch | | Full model | |
|---|---|---|---|---|
| Species, direction | IoU ↑ | PSNR ↑ | IoU ↑ | PSNR ↑ |
| Rose, forward | 0.9043 | 31.82 | 0.9106 | 32.47 |
| Rose, backward | 0.9025 | 30.48 | 0.9153 | 32.80 |
| Lily, forward | 0.9015 | 30.01 | 0.9139 | 30.17 |
| Lily, backward | 0.8965 | 30.03 | 0.9073 | 31.58 |

the training–inference mismatch (i.e., exposure bias) by aligning the conditioning distribution during training with that at inference time. Without DF, the model is trained on clean latent states but must rely on its own predictions during autoregressive generation, leading to error accumulation; the latent state progressively drifts and eventually collapses after a few steps. Because our latent is sparse and two-stage, we adapt diffusion forcing with independent noise levels on the conditioning state in both stages (Sec. A).

**Effect of global anchor.** Removing GA results in noticeable inconsistencies across frames, particularly in appearance, where color drift and identity mismatch are observed. This is reflected quantitatively by degraded perceptual quality (LPIPS $0.1291$ vs. $0.0407$). Without GA, small feature errors accumulate and the appearance drifts; anchoring every step to the input frame bounds this drift. With GA, a global conditioning signal stabilizes structure-latent generation across time, enabling consistent object identity and appearance throughout the growth sequence.

**Transition design and pretraining.** Standard alternatives, including transition strategies inspired by 4DSTAR [58], all perform worse under matched training on rose (Tab. 6), so the transition formulation is a nontrivial technical component of the method. The model trained from scratch without TRELLIS weights retains strong performance (Tab. 7), and the shared bidirectional model performs comparably to two direction-specific models (Sec. C.7).

## 5 Conclusion

Existing 4D generative models are competitive on a limited class of problems but fall well short of faithfully modeling dynamic 4D worlds. We broaden the problem setting and contribute a large-scale 4D long-horizon trajectory dataset and benchmark, on which existing models struggle.

Representation design is central to 4D generation in this setting. We propose a design principle that decouples state representation from state transition for 4D generative models. Under this principle, our instantiation of Latent State Machine (LSM) shows promising results on these challenging cases.

## Acknowledgments and Disclosure of Funding

This paper is in part funded by the Microsoft Research Faculty Fellowship, with computing equipment in part supported by the NVIDIA Academic Grant Program. We also acknowledge support in compute credits from the AMD University Program for AI & HPC Cluster and the Google Cloud Credit Grants (#312170619047) from the Stanford Institute for Human-Centered AI (HAI).

## References


[1] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. *arXiv preprint arXiv:2212.08051*, 2022.

[2] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-XL: A universe of 10M+ 3D objects. *arXiv preprint arXiv:2307.05663*, 2023.

[3] Yufan Deng, Yuhao Zhang, Chen Geng, Shangzhe Wu, and Jiajun Wu. Anymate: A dataset and baselines for learning 3d object rigging. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–10, 2025.

[4] Sherwin Bahmani, Ivan Skorokhodov, Victor Rong, Gordon Wetzstein, Leonidas Guibas, Peter Wonka, Sergey Tulyakov, Jeong Joon Park, Andrea Tagliasacchi, and David B Lindell. 4d-fy: Text-to-4d generation using hybrid score distillation sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7996–8006, 2024.

[5] Chen Geng, Yunzhi Zhang, Shangzhe Wu, and Jiajun Wu. Birth and death of a rose. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26102–26113, 2025.

[6] Yanzhe Lyu, Chen Geng, Karthik Dharmarajan, Yunzhi Zhang, Hadi Alzayer, Shangzhe Wu, and Jiajun Wu. Choreographing a world of dynamic objects. *arXiv preprint arXiv:2601.04194*, 2026.

[7] Zijie Wu, Chaohui Yu, Fan Wang, and Xiang Bai. Animateanymesh: A feed-forward 4d foundation model for text-driven universal mesh animation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13557–13568, 2025.

[8] Bowen Zhang, Sicheng Xu, Chuxin Wang, Jiaolong Yang, Feng Zhao, Dong Chen, and Baining Guo. Gaussian variation field diffusion for high-fidelity video-to-4d synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12502–12513, 2025.

[9] Chen Hongyuan, Chen Xingyu, Youjia Zhang, Xu Zexiang, and Chen Anpei. Motion 3-to-4: 3d motion reconstruction for 4d synthesis. *arXiv preprint arXiv:2601.14253*, 2026.

[10] Jianqi Chen, Biao Zhang, Xiangjun Tang, and Peter Wonka. V2m4: 4d mesh animation reconstruction from a single monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11643–11653, October 2025.

[11] Lukas Uzolas, Elmar Eisemann, and Petr Kellnhofer. Motiondreamer: Exploring semantic video diffusion features for zero-shot 3d mesh animation. In *2025 International Conference on 3D Vision (3DV)*, pages 893–904. IEEE, 2025.

[12] Ziqiao Ma, Xuweiyi Chen, Shoubin Yu, Sai Bi, Kai Zhang, Chen Ziwen, Sihan Xu, Jianing Yang, Zexiang Xu, Kalyan Sunkavalli, et al. 4d-lrm: Large space-time reconstruction model from and to any view at any time. *arXiv preprint arXiv:2506.18890*, 2025.

[13] Yiming Xie, Chun-Han Yao, Vikram Voleti, Huaizu Jiang, and Varun Jampani. Sv4d: Dynamic 3d content generation with multi-frame and multi-view consistency. *arXiv preprint arXiv:2407.17470*, 2024.

[14] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. 2022.

[15] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in neural information processing systems*, 28, 2015.

[16] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21469–21480, 2025.

[17] Tianqi Liu, Zihao Huang, Zhaoxi Chen, Guangcong Wang, Shoukang Hu, Liao Shen, Huiqiang Sun, Zhiguo Cao, Wei Li, and Ziwei Liu. Free4d: Tuning-free 4d scene generation with spatial-temporal consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 25571–25582, October 2025.

[18] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. Dreamgaussian4d: Generative 4d gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023.

[19] Daniel Wang, Patrick Rim, Tian Tian, Dong Lao, Alex Wong, and Ganesh Sundaramoorthi. Ode-gs: Latent odes for dynamic scene extrapolation with 3d gaussian splatting. *arXiv preprint arXiv:2506.05480*, 2025.

[20] Boming Zhao, Yuan Li, Ziyu Sun, Lin Zeng, Yujun Shen, Rui Ma, Yinda Zhang, Hujun Bao, and Zhaopeng Cui. Gaussianprediction: Dynamic 3d gaussian prediction for motion extrapolation and free view synthesis. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–12, 2024.

[21] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *2003.08934*, 2020.

[22] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.

[23] Chun-Han Yao, Yiming Xie, Vikram Voleti, Huaizu Jiang, and Varun Jampani. Sv4d 2.0: Enhancing spatio-temporal consistency in multi-view video diffusion for high-quality 4d generation. pages 13248–13258, 2025.

[24] Linzhan Mou, Jiahui Lei, Chen Wang, Lingjie Liu, and Kostas Daniilidis. Dimo: Diverse 3d motion generation for arbitrary objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14357–14368, 2025.

[25] Ting-Hsuan Liao, Haowen Liu, Yiran Xu, Songwei Ge, Gengshan Yang, and Jia-Bin Huang. Pad3r: Pose-aware dynamic 3d reconstruction from casual videos. *SIGGRAPH ASIA*, 2025.

[26] Qi Sun, Can Wang, Jiaxiang Shang, Wensen Feng, and Jing Liao. Animus3d: Text-driven 3d animation via motion score distillation. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–11, 2025.

[27] Bowen Zhang, Sicheng Xu, Chuxin Wang, Jiaolong Yang, Feng Zhao, Dong Chen, and Baining Guo. Gaussian variation field diffusion for high-fidelity video-to-4d synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12502–12513, October 2025.

[28] Quankai Gao, Qiangeng Xu, Zhe Cao, Ben Mildenhall, Wenchao Ma, Le Chen, Danhang Tang, and Ulrich Neumann. Gaussianflow: Splatting gaussian dynamics for 4d content creation. *arXiv preprint arXiv:2403.12365*, 2024.

[29] Zhiqi Li, Yiming Chen, and Peidong Liu. Dreammesh4d: Video-to-4d generation with sparse-controlled gaussian-mesh hybrid representation. *Advances in Neural Information Processing Systems*, 37:21377–21400, 2024.

[30] Hanwen Liang, Yuyang Yin, Dejia Xu, Hanxue Liang, Zhangyang Wang, Konstantinos N Plataniotis, Yao Zhao, and Yunchao Wei. Diffusion4d: Fast spatial-temporal consistent 4d generation via video diffusion models. *arXiv preprint arXiv:2405.16645*, 2024.

[31] Jiawei Ren, Cheng Xie, Ashkan Mirzaei, Karsten Kreis, Ziwei Liu, Antonio Torralba, Sanja Fidler, Seung Wook Kim, Huan Ling, et al. L4gm: Large 4d gaussian reconstruction model. *Advances in Neural Information Processing Systems*, 37:56828–56858, 2024.

[32] Qi Sun, Zhiyang Guo, Ziyu Wan, Jing Nathan Yan, Shengming Yin, Wengang Zhou, Jing Liao, and Houqiang Li. Eg4d: Explicit generation of 4d object without score distillation. *arXiv preprint arXiv:2405.18132*, 2024.

[33] Yikai Wang, Xinzhou Wang, Zilong Chen, Zhengyi Wang, Fuchun Sun, and Jun Zhu. Vidu4d: Single generated video to high-fidelity 4d reconstruction with dynamic gaussian surfels. *Advances in Neural Information Processing Systems*, 37:131316–131343, 2024.

[34] Zeyu Yang, Zijie Pan, Chun Gu, and Li Zhang. Diffusion$^2$: Dynamic 3d content generation via score composition of video and multi-view diffusion models. *arXiv preprint arXiv:2404.02148*, 2024.

[35] Heng Yu, Chaoyang Wang, Peiye Zhuang, Willi Menapace, Aliaksandr Siarohin, Junli Cao, László Jeni, Sergey Tulyakov, and Hsin-Ying Lee. 4real: Towards photorealistic 4d scene generation via video diffusion models. *Advances in Neural Information Processing Systems*, 37: 45256–45280, 2024.

[36] Yu-Jie Yuan, Leif Kobbelt, Jiwen Liu, Yuan Zhang, Pengfei Wan, Yu-Kun Lai, and Lin Gao. 4dynamic: Text-to-4d generation with hybrid priors. *arXiv preprint arXiv:2407.12684*, 2024.

[37] Haiyu Zhang, Xinyuan Chen, Yaohui Wang, Xihui Liu, Yunhong Wang, and Yu Qiao. 4diffusion: Multi-view video diffusion model for 4d generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=SFk7AMpyhx`.

[38] Yanqin Jiang, Li Zhang, Jin Gao, Weimin Hu, and Yao Yao. Consistent4d: Consistent 360 {\deg} dynamic object generation from monocular video. *arXiv preprint arXiv:2311.02848*, 2023.

[39] Yuyang Zhao, Zhiwen Yan, Enze Xie, Lanqing Hong, Zhenguo Li, and Gim Hee Lee. Animate124: Animating one image to 4d dynamic scene. *arXiv preprint arXiv:2311.14603*, 2023.

[40] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 851–866. 2023.

[41] Mathis Petrovich, Michael J. Black, and Gül Varol. Action-conditioned 3d human motion synthesis with transformer vae, 2021.

[42] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10975–10985, 2019.

[43] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 157–164. 2023.

[44] Dafei Qin, Jun Saito, Noam Aigerman, Thibault Groueix, and Taku Komura. Neural face rigging for animating and retargeting facial meshes in the wild. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023.

[45] Tal Daniel and Aviv Tamar. Unsupervised image representation learning with deep latent particles. *arXiv preprint arXiv:2205.15821*, 2022.

[46] Bernhard Egger, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, et al. 3d morphable face models—past, present, and future. *ACM Transactions on Graphics (ToG)*, 39(5):1–38, 2020.

[47] Stephen W Bailey, Dalton Omens, Paul Dilorenzo, and James F O'Brien. Fast and deep facial deformations. *ACM Transactions on Graphics (TOG)*, 39(4):94–1, 2020.

[48] James Booth, Anastasios Roussos, Stefanos Zafeiriou, Allan Ponniah, and David Dunaway. A 3d morphable model learnt from 10,000 faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5543–5552, 2016.

[49] Volker Blanz and Thomas Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Transactions on pattern analysis and machine intelligence*, 25(9):1063–1074, 2003.

[50] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[51] Guangzhao He, Chen Geng, Shangzhe Wu, and Jiajun Wu. Category-agnostic neural object rigging. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22078–22088, 2025.

[52] Chen Geng, Hong-Xing Yu, Sida Peng, Xiaowei Zhou, and Jiajun Wu. Neural polynomial gabor fields for macro motion analysis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dTlKCQuuxP.

[53] Shangzhe Wu, Ruining Li, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. Magicpony: Learning articulated 3d animals in the wild. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8792–8802, 2023.

[54] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. In *CVPR*, 2022.

[55] Zijie Wu, Chaohui Yu, Fan Wang, and Xiang Bai. Animateanymesh++: A flexible 4d foundation model for high-fidelity text-driven mesh animation, 2026. URL https://arxiv.org/abs/2604.26917.

[56] Jiraphon Yenphraphai, Ashkan Mirzaei, Jianqi Chen, Jiaxu Zou, Sergey Tulyakov, Raymond A Yeh, Peter Wonka, and Chaoyang Wang. Shapegen4d: Towards high quality 4d shape generation from videos. *arXiv preprint arXiv:2510.06208*, 2025.

[57] Zhibing Li, Mengchen Zhang, Tong Wu, Jing Tan, Jiaqi Wang, and Dahua Lin. Ss4d: Native 4d generative model via structured spacetime latents. *ACM Trans. Graph.*, 44(6), December 2025. ISSN 0730-0301. doi: 10.1145/3763302. URL https://doi.org/10.1145/3763302.

[58] Liying Yang, Jialun Liu, Jiakui Hu, Chenhao Guan, Haibin Huang, Fangqiu Yi, Chi Zhang, and Yanyan Liang. Spatial-temporal state propagation autoregressive model for 4d object generation. *arXiv preprint arXiv:2602.18830*, 2026.

[59] Chenchang Li, Zihao Ai, Tong Wu, Xiaosa Li, Wenbo Ding, and Huazhe Xu. Deformnet: Latent space modeling and dynamics prediction for deformable object manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14770–14776. IEEE, 2024.

[60] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International conference on learning representations*, 2020.

[61] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[62] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[63] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[64] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[65] Bokui Shen, Zhenyu Jiang, Christopher Choy, Silvio Savarese, Leonidas J Guibas, Anima Anandkumar, and Yuke Zhu. Action-conditional implicit visual dynamics for deformable object manipulation. *The International Journal of Robotics Research*, 43(4):437–455, 2024.

[66] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018.

[67] Connor Schenck and Dieter Fox. Spnets: Differentiable fluid dynamics for deep neural networks. 2018.

[68] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[69] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[70] Ravinder Bhattoo, Sayan Ranu, and NM Krishnan. Learning articulated rigid body dynamics with lagrangian graph neural network. *Advances in Neural Information Processing Systems*, 35: 29789–29800, 2022.

[71] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

[72] Marc Finzi, Ke Alexander Wang, and Andrew G Wilson. Simplifying hamiltonian and lagrangian neural networks via explicit constraints. *Advances in neural information processing systems*, 33:13880–13889, 2020.

[73] Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019.

[74] Himangi Mittal, Peiye Zhuang, Hsin-Ying Lee, and Shubham Tulsiani. Uniphy: Learning a unified constitutive model for inverse physics simulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16208–16218, 2025.

[75] Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*, pages 23279–23300. PMLR, 2023.

[76] Honglin Chen, Rundi Wu, Eitan Grinspun, Changxi Zheng, and Peter Yichen Chen. Implicit neural spatial representations for time-dependent pdes. In *International Conference on Machine Learning*, pages 5162–5177. PMLR, 2023.

[77] Yichen Li, Peter Yichen Chen, Tao Du, and Wojciech Matusik. Learning preconditioners for conjugate gradient pde solvers. In *International Conference on Machine Learning*, pages 19425–19439. PMLR, 2023.

[78] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[79] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 2019.

[80] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

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

[82] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

[83] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TIP*, 2004.

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

[85] 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. *arXiv preprint arXiv:2512.14692*, 2025.

[86] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[87] 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.

[88] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

# A Datasets and Implementation Details

**Implementation details.** We build LSM on TRELLIS [16, 85]. The state machine operates on the $16\times16\times16\times8$ structure latent and the $64\times64\times64$ active-voxel grid of TRELLIS' two-stage VAE. The state-transition flow and the global anchor flow are fully finetuned for 280k and 100k iterations, respectively; the image-to-structure flow, the structure-to-SLat flow, the SLat VAE encoder, and the (3DGS, mesh) SLat decoders are LoRA-finetuned for 10k iterations each. We use AdamW [86] with learning rate $10^{-4}$ throughout.

**Diffusion forcing on the two-stage latent.** The latent is sparse and two-stage (active-voxel structure plus per-voxel features), so a small error at one step changes which voxels are active and the next step compounds it through both stages. Diffusion forcing therefore has to be adapted rather than applied directly: we add independent noise levels on the conditioning state in both stages, so inference-time self-conditioning matches training.

**Dataset details.** We construct a procedural flower-growth benchmark covering five species rendered in Blender: rose, lily, hibiscus, daisy, and dahlia. Each rollout has 120 frames for rose, 100 for lily and dahlia, and 80 for daisy and hibiscus. We scale up the dataset by randomizing the geometry-node parameters of each procedural model. As far as we know, plant growth is the only natural long-horizon dynamic phenomenon that has many procedural-generation assets publicly available online.

The training split contains 50k procedurally-generated rollouts of rose, 20k each of lily, dahlia, and daisy, and 10k of hibiscus, summing to 120k rollouts. The test split is generated by the same procedural pipeline with disjoint random seeds: 50 rose rollouts (120 frames each), 100 lily and 100 dahlia rollouts (100 frames each), and 100 daisy and 100 hibiscus rollouts (80 frames each), totaling 450 unseen rollouts ($\sim$42k frames).

For each frame we render 15 views to extract the DINOv2 [87] dense 3D features. For each active voxel grid in the sparse structure, we have the mean unproject image features from DINOv2.

**Baseline implementation details.** For Free4D [17] and DG4D [18], we use their public codebases with Wan2.2-I2V [88] finetuned on our flower videos as the driving signal. For AnimateAnyMesh [7], we follow the official code and finetune on our data, uniformly downsampling each mesh to 4096 vertices. ODE-GS [19] and GaussianPrediction's [20] original setup observes most of the time window and forecasts a short suffix; we modify it to observe the first half ($0 \le t < 0.5$) with 14 training views and forecast the second half ($0.5 \le t \le 1.0$). ODE-GS and GaussianPrediction require multi-frame observations to fit scene-specific dynamics and cannot operate from a single image. All methods are evaluated on the same held-out second half. We tried to compare with SS4D [57]: only its inference code and VAE decoder weights are public, so we could not finetune it on our data; its pretrained model shows a clear domain gap on our scenes, and its short training clips leave the sequences covering only part of the lifespan.

# B Computational Resources

Table 8: Computational resources per pipeline stage.

| Stage | Time |
|---|---|
| Dataset rendering (5×8 TITAN RTX + 3×8 2080Ti + 2×8 L40S; all 5 species) | 60 hrs |
| TRELLIS VAEs + image-conditioned flow LoRA finetuning (4× RTX 6000 Blackwell; per flower, 10k iter. each) | 10 hrs |
| State-transition flow training (4× RTX 6000 Blackwell; per flower, 280k iter.) | 140 hrs |
| Global anchor flow training (4× RTX 6000 Blackwell; per flower, 100k iter.) | 50 hrs |

Table 8 reports the GPU hours for each stage of our pipeline. Dataset rendering is a one-time cost that covers all five flower species; LSM training (LoRA finetuning of the pretrained TRELLIS components, full finetuning the state-transition and the global anchor flow matching model) is performed independently per flower.

Table 9: Comparison under the observe-half-then-forecast protocol. ODE-GS and GaussianPrediction observe the first half of each trajectory; LSM receives only the boundary frame. All methods are evaluated on the same held-out second half.

| Method | Input | IoU ↑ | CD ↓ | PSNR ↑ |
|---|---|---|---|---|
| ODE-GS [19] | Half trajectory, 14 views | 0.1038 | 0.08314 | 22.31 |
| GaussianPrediction [20] | Half trajectory, 14 views | 0.1219 | 0.08056 | 22.64 |
| LSM (Ours) | Single boundary image | **0.9199** | **0.01503** | **31.76** |

Table 10: Distributional spread of the generated rollouts. We sample several times and report population statistics per species: bloom size (active-voxel count), color (circular hue and saturation statistics of the flower region), stem thickness, and orientation tilt.

| Species | Size mean $\pm$ std | Hue mean / std (deg) | Saturation | Stem size | Tilt (deg) |
|---|---|---|---|---|---|
| Dahlia | $4195.5 \pm 957.1$ | 70.7 / 129.2 | 0.328 | $0.022 \pm 0.005$ | $1.6 \pm 1.5$ |
| Daisy | $1916.6 \pm 420.0$ | 86.7 / 97.4 | 0.375 | $0.032 \pm 0.010$ | $10.5 \pm 3.5$ |
| Hibiscus | $3249.8 \pm 369.0$ | 351.6 / 36.8 | 0.416 | $0.017 \pm 0.006$ | $7.2 \pm 2.5$ |
| Lily | $2161.3 \pm 348.1$ | 334.1 / 86.8 | 0.224 | $0.026 \pm 0.005$ | $22.7 \pm 5.4$ |
| Rose | $1699.4 \pm 495.9$ | 356.4 / 83.9 | 0.597 | $0.014 \pm 0.002$ | $7.7 \pm 2.6$ |
| Sunflower | $1978.2 \pm 217.9$ | 51.7 / 14.0 | 0.653 | - | - |

# C Additional Results

## C.1 Observe-Half-Then-Forecast Protocol

The conditioning asymmetry favors the baselines: ODE-GS and GaussianPrediction observe the first half of each trajectory with 14 views, while LSM receives only the boundary frame. Under this protocol, LSM still surpasses both baselines significantly (Tab. 9).

## C.2 Distributional Evaluation

We sample several times using our generative model and quantify the distributional spread of the generated rollouts with population statistics per species: bloom size (active-voxel count), color (circular hue and saturation statistics of the flower region), stem thickness, and orientation tilt. The results in Tab. 10 show that our generation has a commendable diversity.

## C.3 Failure Modes of the Baselines

*Deformable meshes.* Deformable meshes used in AnimateAnyMesh assume a fixed topology throughout the generation, while real natural phenomena like plant growth involves the appearance of new petals, leaves, and stem segments, which requires inserting and removing primitives — an operation such representation cannot express. AnimateAnyMesh's reliance on a small set of mesh control nodes further restricts its ability to capture complex local geometry changes.

*Deformable GS.* The deformable GS representations used in ODE-GS and GaussianPrediction both freeze the set of Gaussian primitives at initialization and predict only their state evolution. ODE-GS produces reasonable reconstructions *within* the observed time window, where its 3DGS representation behaves as a per-frame fit, but its predictions degrade significantly outside this window — an extrapolation failure that compounds with the topology-fixedness of the underlying representation. GaussianPrediction collapses further: it fails to capture temporal dynamics at all and produces near-canonical outputs across all frames, indicating that its keypoint-GCN motion model is unable to drive the kind of large-magnitude topological change present in flower growth.

*Video-diffusion-driven pipelines.* Free4D struggles to recover plausible geometry in its final 4DGS optimization process, especially on textureless renderings, and its rollout horizon is bounded by the underlying I2V model. DreamGaussian4D shares similar issue in rollout horizon because of the video-lifting design, with limited topology variation due to its deformation-based state representation. LSM instead models the temporal transition with autoregressive generation, yielding long-horizon sequences without drifting.

Table 11: Comparison with video-to-4D methods.

| Method | Input | Geometry | | Appearance | | |
|---|---|---|---|---|---|---|
| | | IoU ↑ | CD ↓ | SSIM ↑ | PSNR ↑ | LPIPS ↓ |
| SV4D [13] | GT-rendered video | 0.0103 | 0.11141 | 0.8894 | 21.57 | 0.2107 |
| | Wan 2.2 video | 0.0109 | 0.10215 | 0.8796 | 19.69 | 0.2510 |
| SV4D 2.0 [23] | GT-rendered video | 0.0110 | 0.11893 | 0.8877 | 21.74 | 0.1642 |
| | Wan 2.2 video | 0.0130 | 0.11315 | 0.8937 | 20.96 | 0.1977 |
| GVFDiffusion [8] | GT-rendered video | 0.0328 | 0.07755 | 0.8982 | 22.04 | 0.2444 |
| | Wan 2.2 video | 0.0203 | 0.10094 | 0.9030 | 20.32 | 0.2387 |
| LSM (Ours) | Single image | **0.9069** | **0.01397** | **0.9603** | **31.71** | **0.0408** |

Table 12: Voxel IoU along longer rollouts.

| Rollout frame | Forward | | Backward | |
|---|---|---|---|---|
| | Rose | Lily | Rose | Lily |
| 10 | 0.9606 | 0.9477 | 0.9588 | 0.9537 |
| 100 | 0.9344 | 0.9195 | 0.9331 | 0.9183 |
| 160 | 0.9138 | 0.8970 | 0.9266 | 0.8814 |
| 220 | 0.9026 | 0.8647 | 0.9158 | 0.8570 |
| 280 | 0.8674 | 0.8202 | 0.8781 | 0.8176 |
| 360 | 0.7922 | - | 0.7831 | - |

Table 13: Voxel IoU beyond the training horizon.

| Rollout offset | Training horizon | | |
|---|---|---|---|
| | First $1/4$ | First $1/2$ | Full |
| 10 | 0.9551 | 0.9548 | 0.9532 |
| 30 | 0.9431 | 0.9429 | 0.9416 |
| 60 | 0.8608 | 0.9171 | 0.9167 |
| 90 | 0.7934 | 0.8235 | 0.8997 |
| 120 | 0.7438 | 0.7648 | 0.8886 |

### C.4 Comparison with Video-to-4D Methods

We further compare with SV4D [13], SV4D 2.0 [23], and GVFDiffusion [8]. These methods operate under a different setup: they take videos as input and perform 4D reconstruction. We therefore evaluate them using (1) videos generated by the same fine-tuned Wan 2.2 model used for Free4D and DG4D, and (2) ground-truth rendered videos. As shown in Tab. 11, LSM outperforms all baselines across all metrics.

### C.5 Long-Horizon Rollouts

The lifespan endpoints are fixed states of each phenomenon, so sequences cannot be extended beyond them. To obtain quantitative ground truth for longer sequences, we re-render the same lifespans at a finer temporal resolution, producing sequences of up to $360$ frames, and retrain the models under otherwise identical settings. We report voxel IoU along the rollout in Tab. 12. Performance degrades gradually with rollout length but does not collapse.

We further train models on only the first $1/4$ or $1/2$ of each lifespan and roll them out over the full lifespan. Every offset beyond the prefix boundary therefore lies outside the training horizon, with ground truth available at each frame. As shown in Tab. 13, the model can extrapolate beyond its training horizon, although performance gradually degrades with longer rollouts.

### C.6 Generalization

Our generality claim is on the problem formulation and training recipe, not a single trained model (Sec. 4.4). Each trained model instantiates this common recipe, and its performance is limited by the amount and coverage of training data. We use procedural generation because it is currently the only way to obtain long-horizon, topology-changing 4D sequences with full 3D ground truth. The method only takes rendered sequences as input, and our results also cover settings beyond procedural flower growth.

*Non-plant domains.* We evaluate candle burning and animal evolution under the same protocol as Tab. 2. As shown in Tab. 4, LSM substantially outperforms both baselines on all metrics, showing that the proposed formulation also applies to non-plant dynamic processes.

*Non-procedural datasets.* We train and evaluate on DyMesh [7] and ObjXL4D [2, 30] with the same recipe (Tab. 5, top). These results show that the proposed formulation is not specific to our procedural flower benchmark. *OOD procedural parameters.* We generate test sequences with procedural

Table 14: Unified model versus per-species models.

| | Forward | | | Backward | | |
|---|---|---|---|---|---|---|
| | IoU ↑ | CD ↓ | PSNR ↑ | IoU ↑ | CD ↓ | PSNR ↑ |
| Unified | 0.9085 | 0.01394 | 31.41 | 0.9095 | 0.01434 | 31.19 |
| Per-species models | 0.9093 | 0.01403 | 31.67 | 0.9046 | 0.01391 | 31.74 |

Table 15: Shared bidirectional model versus two direction-specific models.

| | Forward | | Backward | |
|---|---|---|---|---|
| | IoU ↑ | PSNR ↑ | IoU ↑ | PSNR ↑ |
| LSM (Ours) | 0.9093 | 31.67 | 0.9046 | 31.74 |
| Direction-specific models | 0.9059 | 31.46 | 0.9057 | 31.69 |

parameters outside the training ranges (Tab. 5, bottom). The performance of our model remains competitive under these parameter shifts.

*Held-out species.* We create a sixth species, sunflower, using the same procedural pipeline and evaluate the model zero-shot, without training on sunflower sequences. The model achieves an IoU of $0.6295$ and a Chamfer distance of $0.0914$, demonstrating transfer to an unseen species.

*Unified model.* For Fig. 4, we train a separate model for each category or phenomenon. We also train a single unified model jointly on all five flower species and both temporal directions. Its performance is comparable to that of the per-species models (Tab. 14). These results show that LSM can model multiple species and both temporal directions within a single checkpoint, without species-specific architectures.

*Real images.* We apply the model to real images without finetuning. Real photos have no 4D ground truth, so we measure agreement between two independently trained models on the same real inputs: $0.985$ mesh IoU. These results support a capability that, to our knowledge, no existing method provides: train on procedural or reconstructed 4D data, then predict the complete bidirectional lifespan of a real object from a single photo.

### C.7 Transition Design, TRELLIS Pretraining, and Shared Bidirectional Transition

*Transition design.* Since 4DSTAR [58] is a preprint and its official implementation is not yet available, we implement several transition strategies inspired by 4DSTAR within our framework: a residual propagation transition (static content plus per-step offsets) and 2-state and 4-state history caches. Our design achieves the best performance across all variants. Our transition design is specific to this setting rather than a standard choice in prior dynamic modeling work.

*Contribution of TRELLIS pretraining.* To isolate the contribution of TRELLIS pretraining, we retrain the model from scratch without initializing from pretrained TRELLIS weights. The model trained from scratch retains strong performance (Tab. 7), showing that the gains are not solely due to TRELLIS pretraining and that our formulation itself is effective.

*Shared bidirectional transition.* Forward and backward refer to temporal direction, i.e., from $t-1$ to $t$ or from $t$ to $t-1$, rather than to the semantic type of evolution. Petals falling off, for example, is still forward dynamics because it occurs as time progresses in the physical world. Thus, both directions describe transitions between nearby states of the same physical process, differing only in temporal direction. The direction embedding is not merely a fixed shift added to the latent representation. Instead, it serves as a conditioning input to the transition transformer, allowing the shared model to represent two distinct conditional distributions. It selects the temporal direction in the same way that a class label conditions a class-conditional diffusion model. To evaluate whether this shared model has sufficient capacity, we also train two direction-specific models and compare them with our bidirectional model. The shared bidirectional model performs comparably to the two direction-specific models (Tab. 15), showing that it has sufficient capacity to model both directions.

Table 16: Inference time and peak VRAM on a single RTX PRO 6000 Blackwell GPU. All baselines are rerun on the same GPU.

| | Time ↓ | Peak VRAM ↓ |
|---|---|---|
| LSM, per AR step (structure + SLat flows, 25 + 25 steps) | 2.45 s | 9.6 GB |
| LSM, decode per frame (gaussian + mesh) | 0.05 s | 12.6 GB |
| LSM, 120-frame rollout | 4.9 min | 9.6 GB |
| Free4D [17], per scene (10k iterations) | 14.1 min | 12.4 GB |
| DG4D [18], per scene (2000 iterations) | 20.8 min | 30.3 GB |
| ODE-GS [19], per scene | 28.1 min | 3.8 GB |
| GaussianPrediction [20], per scene | 15.6 min | 8.8 GB |

Table 17: Parameters and checkpoint size of each component of LSM.

| | Parameters ↓ | Checkpoint size ↓ |
|---|---|---|
| SS VAE encoder | 59.3M | 119 MB |
| SS VAE decoder | 73.7M | 148 MB |
| SLat VAE encoder | 85.8M | 173 MB |
| SLat VAE Gaussian Splatting decoder | 85.4M | 171 MB |
| SLat VAE Mesh decoder | 90.9M | 182 MB |
| SS state transition flow (bidirectional) | 559.7M | 1120 MB |
| SLat global-anchor transition flow | 600.4M | 1201 MB |
| Total | 1555.2M | 3114 MB |

### C.8 Efficiency

Our efficiency claims are comparative. *Inference-time computation.* As an amortized model, LSM is more efficient than the optimization-based baselines. All measurements were obtained on a single RTX PRO 6000 Blackwell GPU. We reran all baselines on the same GPU to measure peak memory usage. As shown in Tab. 16, our method generates the full sequence faster than all baselines while maintaining moderate peak memory usage.

*Data efficiency.* Even after finetuning on the same rollouts, a 14B-parameter pretrained video model achieves about 10 dB lower PSNR than LSM. In contrast, LSM uses no video pretraining and learns the dynamics solely from these rollouts. The parameters and checkpoint size of each component are listed in Tab. 17.

## D Broader Impacts

Our model can lower the cost of producing dynamic visual content for film, games, education, and scientific visualization, where current pipelines rely on expensive capture or manual authoring. Learned models of natural growth and decay may also offer differentiable proxies for slow biological and physical processes that are hard to observe directly, with potential use in biology, agriculture, and ecology research.

Several risks also exist. Generated rollouts may be mistaken for real measurements if used outside the trained domain, so provenance disclosure is important. Models trained on procedural data inherit the biases of the underlying programs and will under-represent species, conditions, and dynamics absent from the training set. Finally, training large 4D models carries energy costs that should be weighed against the application scope.

## E Limitations

Our method relies on the assumption of extensive training data. Therefore, for natural phenomena that currently lack sufficient data, our model may fail to train successfully. One way to mitigate this is to distill from video generative models [6], which we leave for future work. This paper does not aim to solve the broader problem of 4D data scarcity. Instead, we show that the proposed setting remains challenging for existing methods even when sufficient training data are available. This is a

forward-looking problem, as advances in 4D reconstruction and procedural generation are likely to make such data increasingly available. Our work also partly addresses data availability. We show that procedural generation provides an efficient way to produce large-scale 4D data and may be further scaled through agentic pipelines. We also contribute a large-scale benchmark. Moreover, our state transition model is purely learned from data; therefore, there might be non-conserved energy and momentum in some of the rollouts. We will incorporate physically inspired models [71] into our dynamic model to improve its physical plausibility in future work.