ActionMesh: Animated 3D Mesh Generation with Temporal 3D Diffusion

Remy Sabathier[1,3] David Novotny[2] Niloy J. Mitra[3] Tom Monnier[1]
[1]Meta Reality Labs [2]SpAItial [3]University College London

https://remysabathier.github.io/actionmesh/

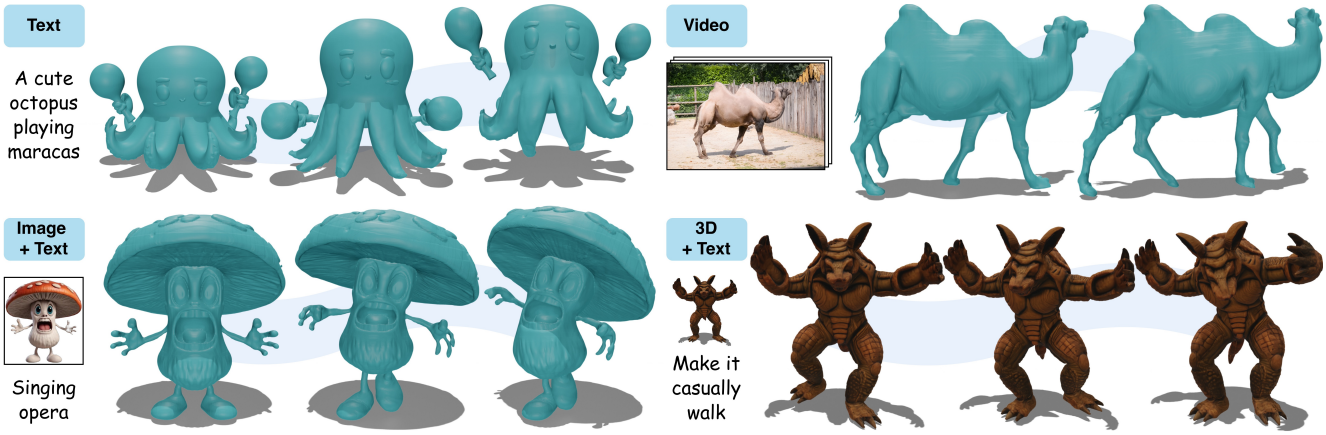

Figure 1. **ActionMesh.** Our model generates 3D meshes 'in action' from a wide range of inputs such as a text prompt, a video, an image + an animation text prompt, or a 3D mesh + an animation text prompt. Unlike previous approaches, our method is not only **fast**, but also **rig-free** and **topology consistent**. These properties are convenient in practice, *e.g.*, they allow the seamless animation of complex 3D shapes like an octopus with maracas (top left) or the automatic transfer of the mesh texture throughout the animation (bottom right).

Abstract

Generating animated 3D objects is at the heart of many applications, yet most advanced works are typically difficult to apply in practice because of their limited setup, their long runtime, or their limited quality. We introduce ActionMesh, a generative model that predicts production-ready 3D meshes 'in action' in a feed-forward manner. Drawing inspiration from early video models, our key insight is to modify existing 3D diffusion models to include a temporal axis, resulting in a framework we dubbed 'temporal 3D diffusion'. Specifically, we first adapt the 3D diffusion stage to generate a sequence of synchronized latents representing time-varying and independent 3D shapes. Second, we design a temporal 3D autoencoder that translates a sequence of independent shapes into the corresponding deformations of a pre-defined reference shape, allowing us to build an animation. Combining these two components, ActionMesh generates animated 3D meshes from different inputs like a monocular video, a text description, or even a 3D mesh with a text prompt describing its animation. Besides, compared to previous approaches, our method is fast and produces results that are rig-free and topology consistent, hence enabling rapid iteration and seamless applications like texturing and retargeting. We evaluate our model on standard video-to-4D benchmarks (Consistent4D, Objaverse) and report state-of-the-art performances on both geometric accuracy and temporal consistency, demonstrating that our model can deliver animated 3D meshes with unprecedented speed and quality.

1. Introduction

The ability to automatically produce animated 3D objects from simple user inputs is a core computer vision problem that holds high promise for any 3D-content application, like video games, animated movies and commercials, or augmented/virtual reality. However, despite recent progress, most works share three main limitations. First, they are specific to limited setups with a predefined input modality (*e.g.*, a video) and predefined object categories (*e.g.*, bipeds or rig-able objects). Second, they often rely on long (30-45 minutes) optimization loops, which are slow and prone to local minima. Third, the overall output quality does not meet the production criteria.

In this paper, we introduce *ActionMesh*, a feed-forward generative model that is simple, scalable, and computes

production-ready 3D meshes 'in action' from diverse inputs. At its core, ActionMesh is a new video-to-4D model that predicts an animated 3D mesh given a video as input. Drawing inspiration from early video models, we extend existing 3D diffusion models with a temporal axis and separate the 3D generation from the animation prediction. Our model runs in two stages. First, we derive a *temporal 3D diffusion model* from a pretrained 3D latent diffusion model, which produces a sequence of synchronized latents representing time-varying but independent 3D meshes. Second, we design a *temporal 3D autoencoder*, which converts a generic sequence of 3D meshes into the deformations of a chosen reference mesh, yielding an animation with constant topology. Importantly, for both stages, we build upon models with strong 3D priors to balance for the lack of 3D animated data.

Compared to existing approaches, ActionMesh is fast (3 minutes for 16-frame video) and produces meshes that are rig-free and topology consistent. These properties are convenient in practice since (i) it allows the animation of complex 3D shapes where rigging is unknown or difficult, and (ii) it automatically preserves the mesh attributes, like a texture, throughout the animation. In addition, thanks to its design, we show that our model can be extended to other generative tasks like text-to-4D, {image+text}-to-4D or {3D+text}-to-animation, as well as related applications like retargeting/motion transfer, as illustrated in Figure 1 and Figure 4. We evaluate our video-to-4D model qualitatively on the Consistent4D benchmark [14] and quantitatively on a newly introduced Objaverse [8] benchmark. Comparing against state-of-the-art methods, including DreamMesh4D [18], LIM [31], V2M4 [4], and the concurrent work ShapeGen4D [50], we show that ActionMesh consistently outperforms competitors on both geometric accuracy and correspondence quality while achieving a speed-up of roughly $10\times$. Code and pretrained weights are available on our webpage: **https://remysabathier.github.io/actionmesh/**.

Summary. Our main contributions are three-fold:

- A fast feed-forward model called ActionMesh that generates animated 3D meshes from diverse inputs with an unprecedented speed and quality.
- A temporal 3D diffusion model that produces synchronized shape latents by minimally extending pretrained 3D diffusion backbones.
- A temporal 3D autoencoder that assembles independent shapes into a single topology-consistent animation via the deformation of a reference mesh.

2. Related Work

3D foundational models. The seminal work of Zhang *et al*. [54] introduces 3DShape2VecSet, a neural field supported by a set of latent vectors (a vecset), designed for scalable 3D encoding of meshes and point clouds, and explicitly targeting generative diffusion. Trellis [45] generalizes this idea to structured 3D latents, scaling to large models that can decode into multiple 3D representations such as radiance fields, Gaussian splats, and meshes. Building on vecset-style latents, Craftsman [16] performs 3D-native diffusion over latent sets coupled with a geometry refiner, enabling high-quality mesh synthesis and editing; Dora-VAE [5] improves VAE sampling on similar representations with a strong emphasis on sharp edges and high-frequency details. CLAY [56] further advances controllable 3D generation with a large 3D-native latent DiT for geometry and a dedicated diffusion model for materials. In parallel, LRM [12] and follow-up works like [34, 38, 52] demonstrate that large transformers trained on massive image collections can reconstruct high-fidelity 3D NeRFs or Gaussians from a single view. Among recent image-to-3D mesh generators, Hunyuan3D [39] predicts geometry with a flow-based DiT and texture with a separate model, generating high-resolution assets from text or images; TripoSG [17] uses a large rectified-flow transformer for high-fidelity mesh reconstruction. While image-to-3D models provide powerful frame-level priors, they process each instance independently and do not explicitly model temporal correspondences, making it difficult to produce an animated mesh with a constant topology across frames.

Video-to-4D via optimization. Targeting dynamic 4D content, a common strategy first synthesizes multi-view or multi-frame videos from a monocular sequence and then optimizes a 4D representation. SV4D and SV4D 2.0 [46, 48] unify multi-frame, multi-view video diffusion to supervise dynamic NeRFs, significantly improving spatio-temporal consistency but still requiring per-scene optimization. CAT4D [43] similarly converts a monocular video into multi-view sequences and then optimizes deformable 3D Gaussians, offering strong reconstructions and controllable camera and time, but without vertex-to-vertex correspondences on a single topology. Related approaches [14, 28, 41, 44, 51, 53, 55] also rely on diffusion or generative priors to supervise dynamic Gaussians or NeRFs, typically with temporal regularizers or sparse controls and a subsequent optimization or training stage for each scene. DreamMesh4D [18] predicts meshes via a mesh–Gaussian hybrid with sparsely controlled deformation followed by optimization. V2M4 [4] enforces topology and texture consistency through a multi-stage pipeline that includes camera search, reposing, pairwise registration, and global texture optimization. LIM [31] learns to interpolate a 3D implicit field over time and extracts a UV-textured mesh sequence with consistent topology through a test-time optimization. These methods achieve high-quality 4D reconstructions but rely on post-optimization or per-scene training.

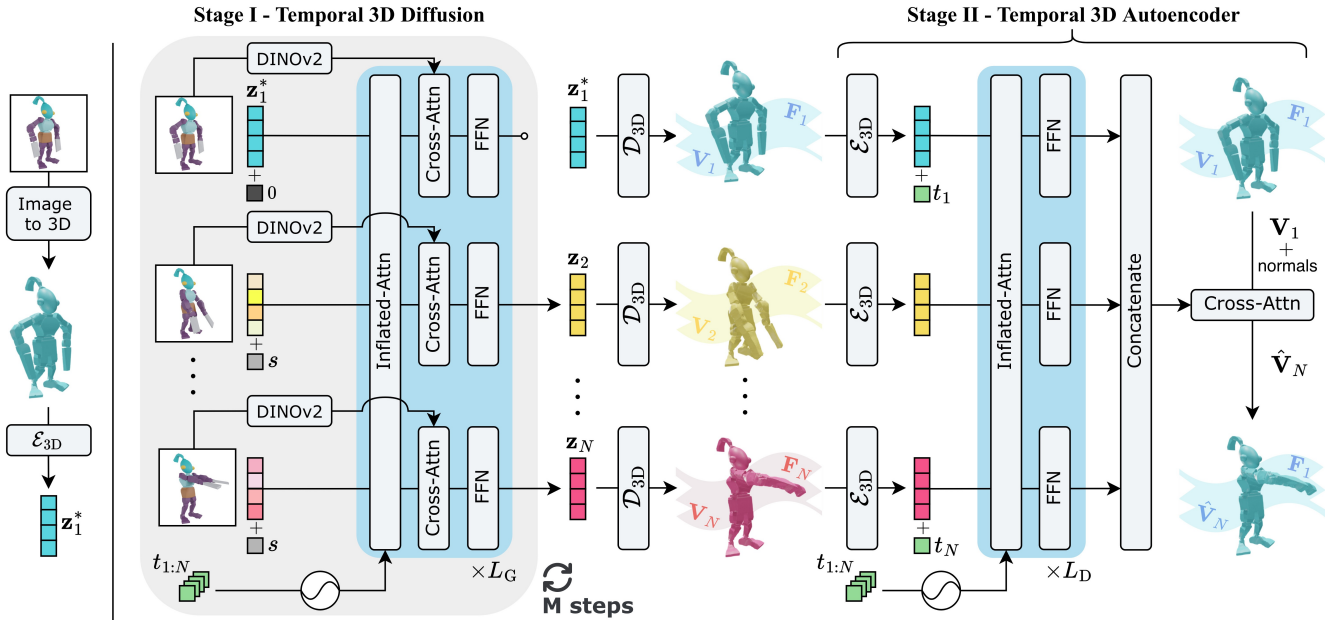

Figure 2. **Overview.** Given an input video, our model generates an animated 3D mesh in two stages. **(Stage I)** After computing a reference mesh latent $\mathbf{z}_1^*$ with an off-the-shelf image-to-3D model, we use our temporal 3D diffusion model to produce from $\mathbf{z}_1^*$ and the video, a sequence of time-varying but independent 3D meshes. **(Stage II)** Our temporal 3D autoencoder takes theses shapes as input and predicts, for each shape in the sequence, a deformation field of the reference mesh vertices, thus yielding an animated 3D mesh with consistent topology.

Feed-forward 4D reconstruction. A complementary line of work reconstructs or generates 4D content in a single forward pass, avoiding test-time optimization. Motion2VecSets [2] proposes a 4D diffusion model that denoises compressed latent sets for dynamic surface reconstruction from point-cloud sequences. L4GM [29] predicts a sequence of 3D Gaussian splats from a monocular video in one pass and then upsamples them temporally via a learned interpolation model, but still operates entirely in Gaussian space rather than producing an animated mesh with fixed topology. Similarly, 4DGT [47] learns a transformer that directly predicts 4D Gaussians from real monocular videos, again operating in Gaussian space instead of explicit meshes. These existing feed-forward methods either produce Gaussians or neural fields instead of a mesh with constant topology. Concurrently, ShapeGen4D [50] extends a pre-trained 3D generative model by introducing temporal attention, enabling feed-forward prediction of video-conditioned 4D meshes. Unlike our approach, it does not explicitly enforce a single, globally consistent mesh topology across the sequence.

Animation-ready 4D assets. Several works focus on making existing 3D assets *animation-ready* by predicting rigs, skinning weights, or deformation fields, instead of reconstructing geometry from raw videos. Make-It-Animatable [10] predicts skeletons, skinning weights, and pose rectification for meshes. MagicArticulate [36] similarly generates articulation-ready rigs and deformations for a wide range of shapes. RigAnything [21] extends auto-rigging to diverse object categories via an autoregressive transformer that predicts hierarchical skeletons and skinning weights. For Gaussian-based assets, RigGS [49] recovers skeletons and skinning directly from videos, enabling articulated motion in the space of 3D Gaussians. DriveAnyMesh [33] deforms an existing mesh given a monocular video, using latent 4D diffusion over point-trajectory sequences. These methods operate on meshes or 3D Gaussians provided a priori, and focus on rigging and animation, rather than reconstructing an animated mesh from a single input video.

3. Method

Our goal is to generate animated 3D meshes from various user inputs. Instead of building input-specific models, we cast multiple generative tasks into the core *video-to-4D* problem, whose goal is to generate an animated 3D mesh given a single video. To address this, we harness pretrained 3D generators to balance for the lack of 3D animated data. Specifically, we build upon the 3D latent diffusion framework of 3DShape2VecSet [54] (Section 3.1) and propose minimal modifications leading to our two-stage model called *ActionMesh*. First, we introduce a temporal 3D diffusion model, predicting a sequence of time-varying but independent 3D meshes (Section 3.2). Second, we present a temporal 3D autoencoder, converting a sequence of independent 3D shapes into an animated mesh with a constant topology (Section 3.3). Finally, we discuss how to apply our model to other scenarios in Section 3.4. Figure 2 shows an overview of ActionMesh.

Terminology and problem setting. We use '3D mesh' to describe a triangular 3D mesh denoted by $\mathcal{M} = (\mathbf{V}, \mathbf{F})$, where $\mathbf{V} \in \mathbb{R}^{N_\text{v} \times 3}$ are the vertex positions and $\mathbf{F} \in \{1, ..., N_\text{V}\}^{N_\text{f} \times 3}$ captures the face connectivity. We use '4D mesh' to describe a sequence of time-varying but *independent* 3D meshes that do not share the same topology, which we denote by $\{(\mathbf{V}_k, \mathbf{F}_k)\}_{k=1}^N$. Finally, we call 'animated 3D mesh' a sequence of 3D meshes sharing the *same* topology and denoted by $\{(\mathbf{V}_k, \mathbf{F})\}_{k=1}^N$. Note that an animated 3D mesh is a particular form of a 4D mesh.

Let $\{\mathbf{I}_k\}_{k=1}^N$ be a video, where $\mathbf{I}_k \in \mathbb{R}^{H \times W \times 3}$ is an RGB frame depicting an object in motion at framestep t_k. Our objective is to predict an animated 3D mesh corresponding to the input video. Specifically, we aim at predicting a reference mesh $\mathcal{M} = (\mathbf{V}, \mathbf{F})$ as well as its vertex position updates $\mathbf{V}_k$ for each framestep t_k, such that $\mathbf{V}_k$ represents the new vertices of $\mathcal{M}$ matching the motion depicted by $\mathbf{I}_k$.

3.1. Background: 3DShape2VecSet

Following the paradigm of latent diffusion models [30], *3DShape2VecSet* learns a 3D diffusion model comprising two stages: (i) a variational autoencoder (VAE) made of an encoder $\mathcal{E}_\text{3D}$ and a decoder $\mathcal{D}_\text{3D}$ to encode 3D shapes into a compact latent space, and (ii) a diffusion model $\mathcal{G}_\text{3D}$ predicting latents conditioned on an input image. Specifically, given a dense set of points $\mathbf{P}$ sampled on the surface, a sparse set of query vectors $\mathbf{Q}$ (called vector set or *VecSet*, which is either learned or subsampled from $\mathbf{P}$) is encoded by a cross-attention layer using $\mathbf{P}$ as context. Then, the query vectors are passed to several self-attention layers to produce the low-dimensional latent $\mathbf{z}$. The decoder $\mathcal{D}_\text{3D}$ takes the latent $\mathbf{z}$, processes it with several self-attention layers and uses the output as context to a cross-attention layer producing the occupancy (or signed distance) of a query 3D point. Finally, a mesh is computed by querying the decoder on a dense grid of points and running a meshification algorithm like marching cubes [23]. Note that this architecture is reminiscent of Perceiver IO [13] used in NLP.

The generative model $\mathcal{G}_\text{3D}$ is a diffusion transformer [26], which takes the form of a decoder-only transformer [40] where each block has an extra cross-attention layer to incorporate the conditioning signal. In follow-up works, the conditioning signal is typically an image that is described using a frozen DINOv2 [25]. Many advanced image-to-3D models build on top of VecSet, *e.g.*, CLAY [56], Craftsman [16], TripoSG [17] or Hunyuan3D [39]. In this work, we adopt TripoSG as our backbone for its state-of-the-art performance and its open-source implementation, but our formulation should be applicable to other VecSet backbones. One specificity of TripoSG is that it relies on rectified flow [20, 22] and that the flow timestep $s \in [0, 1000]$ is Fourier-embedded and concatenated as an additional token (see [17]).

Figure 3. **Image-to-3D results on video frames.** Running an image-to-3D model on each frame produces meshes exhibiting inconsistent global orientation (left) or inconsistent geometric details (right), even when using identical Gaussian noise in the denoiser.

3.2. Stage I: Temporal 3D Diffusion

Our goal in the first stage is to produce a 4D mesh, *i.e.*, a sequence of meshes without consistent topology, from a monocular video. A naive approach is to apply an off-the-shelf image-to-3D generator independently on all frames of the video. However, we found that this per-frame generation exhibits severe inconsistencies across frames, such as inconsistent 3D orientations or geometric errors that manifest as a surface flickering through time (see Figure 3). This is somehow expected since this naive approach lacks a mechanism enforcing consistency across frames.

To encourage cross-frame consistency, we introduce *temporal 3D diffusion models*. Drawing inspiration from multi-image models derived from pretrained image models [9, 11, 32, 35], we propose to augment pretrained 3D diffusion models with a temporal axis to encourage the generations to be synchronized. Specifically, we introduce two minimal changes to the original architecture, namely *inflated attention* and *masked generation*, which we describe next. Note that we use 'temporal 3D diffusion' instead of '4D diffusion' since it is more accurate and it differentiates from recent 4D diffusion works extending image/video models to handle multi-views [19, 42, 55]. Figure 2 illustrates our resulting diffusion model architecture.

Inflated attention. Given a sequence of N frames, we build a model that outputs corresponding 3D latents in a synchronized fashion. Inspired by MVDream [32] for multi-view generation, we propose to *inflate* the existing self-attention layers to allow the cross-frame synchronization of latents. There are two benefits: (i) the tokens now attend to all tokens across frames, and (ii) we leverage existing layers that are already pretrained. Specifically, let $\mathbf{X} \in \mathbb{R}^{N \times T \times D}$ be T tokens of dimension D corresponding to the N frames. Inflated self-attention ($\mathrm{infattn}$) is applied by reshaping the tensor, applying standard self-attention ($\mathrm{selfattn}$) and re-

shaping back for subsequent layers as:

$$\text{infattn}(\mathbf{X}) = \text{reshape}^{-1}(\text{selfattn}(\text{reshape}(\mathbf{X}))), \quad (1)$$

where reshape flattens the first two dimensions such that $\text{reshape}(\mathbf{X}) \in \mathbb{R}^{1\times NT\times D}$ and reshape^{-1} is the reverse operation. To reduce the $(NT)^2$ complexity overhead, we use the efficient FlashAttention2 implementation [6]. This mechanism improved the consistency across latents, yet, we still observed small jittering across consecutive frames. To mitigate this issue, we propose to inject the relative frame position information via rotary positional embedding [37] inside the inflated attention layers. We found this simple solution to generate smoother motions across frames. Note that these simple modifications not only allow us to finetune from pretrained 3D diffusion models, but also to directly reuse the pretrained 3D autoencoders.

Masked generation. The model described above generates consistent 4D output from a given video, but it does not allow us to easily start the generation from known 3D meshes, which has practical applications (see Sec. 3.4). Hence, we turn our model into a *masked generative model* [3, 15] where some 3D latents in the sequence are known and only the remaining 'masked' latents need to be generated. Note that such masked adaptation of pretrained generative models is reminiscent of multi-view models like CAT3D [9].

To do so with minimal architectural changes, we propose to maintain some noise-free 3D latents during our temporal 3D diffusion model training, similar to CAT3D [9]. Specifically, let N_S be the number of source latents and N_T be the number of target latents such that $N = N_\text{S} + N_\text{T}$. During training, given an input sequence of 3D latents $\mathbf{Z} = \{\mathbf{z}_k\}_{k=1}^N \in \mathbb{R}^{N\times T\times D}$, we randomly sample N_S latents and keep the latter noise-free before feeding the sequence to the denoiser. To inform the model about the noise-free latents, we set the flow matching step to 0, which is a more natural solution than CAT3D's binary mask injection. Furthermore, we do not apply the diffusion loss on these source latents during training. During inference, given a set of J source meshes $\{\mathcal{M}_{k_j}\}_{j=1}^J$, we first use the 3D encoder $\mathcal{E}_\text{3D}$ to compute for each $k \in \{k_j\}_{j=1}^J$ the corresponding 3D latent $\mathbf{z}_k^* = \mathcal{E}_\text{3D}(\mathcal{M}_k)$. Then, before each denoising step, we copy the clean latents into the noisy latent sequence, allowing all noised tokens to attend to the latent representations of the known meshes. Technically, our model now corresponds to a *masked* temporal 3D diffusion model.

To run inference from a single video, we first use an off-the-shelf image-to-3D generator, which can be applied to any frame $\mathbf{I}_k$ of the video. In particular, this allows the selection of a frame where the object is well visible and free of distortion and motion-blur. After recovering a 3D mesh, we can then apply our masked model described above. Note that this process is agnostic to the image-to-3D model used, therefore it would directly benefit from advances in this area. Unless stated otherwise, we use TripoSG [17] for the image-to-3D model. See the supplemental for details.

3.3. Stage II: Temporal 3D Autoencoder

Using our temporal 3D diffusion described above, we can predict a generic 4D mesh $\{(\mathbf{V}_k, \mathbf{F}_k)\}_{k=1}^N$ corresponding to the input video. However, this 4D representation is impractical because the mesh topology changes throughout the sequence, preventing downstream applications such as texturing. We thus aim at computing a 4D output corresponding to the explicit animation of a reference mesh $(\mathbf{V}, \mathbf{F})$. For this purpose, we propose to predict time-dependent vertex deformations δ_k such that $(\mathbf{V} + \delta_k, \mathbf{F})$ approximates the surface of $(\mathbf{V}_k, \mathbf{F}_k)$. While prior works rely on slow optimization algorithms to solve this task [4, 18, 31], we instead solve it with a feed-forward autoencoder taking the time-varying 3D meshes as input and outputting temporal deformation fields applied to the reference mesh. We do so by starting from a pretrained VecSet-based VAE which we modify to handle temporal 3D data and produce deformation field outputs, hence yielding a model we term *temporal 3D autoencoder*. Similar to the original VecSet VAE, such an autoencoder is also *translational*, as it translates a sequence of point clouds into a sequence of deformation fields.

Formulation. Given a sequence of N independent meshes, we first sample 3D point clouds for each mesh and pass them independently through the frozen 3D encoder $\mathcal{E}_\text{3D}$ to obtain shape latents $\mathbf{Z} = \{\mathbf{z}_k\}_{k=1}^N$. Importantly, this part is identical to the original 3D autoencoder, which is critical for keeping consistency between the latents predicted by our temporal 3D diffusion and the latents of this temporal 3D autoencoder. On the other hand, our decoder $\mathcal{D}_\text{4D}$ differs from the original $\mathcal{D}_\text{3D}$ by ingesting the entire sequence of latents $\mathbf{Z}$ and predicting for each latent the corresponding 3D deformation field of the reference mesh. Concretely, given two arbitrary framesteps t_i and t_j, the decoder first processes all tokens with self-attention layers and then outputs the displacement from t_i to t_j of a query 3D point via a final cross-attention layer. To indicate source and target, the framesteps (t_i, t_j) are Fourier-embedded, concatenated, and injected as an extra token. During training, the query points correspond to 3D points randomly sampled on the source mesh surface whereas during inference, we feed the 3D vertex positions of the reference mesh. In practice, we augment such query points with their normals and found that such local geometry helps disambiguate points that are spatially close yet topologically distant. Following insights from our temporal 3D diffusion model, we encourage cross-shape consistency by inflating the decoder's self-attention layers and encoding relative position offsets with rotary embeddings. The autoencoder is illustrated in Figure 2.

3.4. Applications

As described above, ActionMesh predicts an animated 3D mesh given a video, thus solving a **video-to-4D** problem. One of its specificities is its masked generative modeling which enables us to incorporate known 3D shapes in the generation process. This characteristic not only allows us to solve a **{3D+video}-to-animation** problem, but also to unlock several useful applications that we describe next. Figures 1 and 4 showcase some of these applications.

{3D+text}-to-animation. To predict an animation given a mesh and a text prompt describing the motion, we first render the mesh using a frontal viewpoint and a white background to get an image $\mathbf{I}_1$. Then, we use an off-the-shelf video model to animate the image $\mathbf{I}_1$ given the text description, yielding a video $\{\mathbf{I}_k\}_{k=1}^{N}$. Finally, we apply ActionMesh with the known 3D and the generated video.

{Image+text}-to-4D. To predict an animated 3D mesh from an image depicting an object and a text prompt describing the motion, we use an off-the-shelf image-to-3D model to recover a 3D mesh, and then use our {3D+text}-to-animation process.

Text-to-4D. To predict an animated 3D mesh from a single text prompt, there are two possibilities: (i) we can run a video model to generate a video from the text and use ActionMesh; or (ii) we can use an image generator to compute an image, and then run our {image+text}-to-animation process. In practice, we use the latter option.

Motion transfer / retargeting. Although our model was not explicitly trained for retargeting, we found that it can transfer motion from an input video representing an object A to a different 3D object B. Specifically, this is done by simply running our {3D+video}-to-animation process with inconsistent objects.

Animation extrapolation. Given its autoregressive modeling, ActionMesh can also extrapolate animations, for instance generating coherent animations from long video sequences. Concretely, we split the long video into chunks and use the first chunk as input to our video-to-4D component. Then, we iterate the {3D+video}-to-animation process on the ensuing chunks, by recursively inputting the 3D output corresponding to the last frame of the previous chunk.

4. Experiments

We evaluate our model on the common video-to-4D task. We first compare to the state of the art through a quantitative comparison on an in-house benchmark constructed from Objaverse [8] as well as a qualitative analysis on the standard Consistent4D benchmark [14] (Section 4.1). Then, we present additional results corresponding to other applications of our method and we conduct an ablation study of our key components (Section 4.2).

Table 1. **Quantitative results on Objaverse.** We report results from state-of-the-art prior works, namely LIM [31], DreamMesh4D [18] and V2M4 [4]. Our method outperforms competitors across all metrics, while being significantly faster.

Method	Time	CD-3D ↓	CD-4D ↓	CD-M ↓
LIM [31]	15min	0.095	0.127	0.258
DM4D [18]	35min	0.095	0.140	0.247
V2M4 [4]	35min	0.063	0.223	0.500
Ours	**3min**	**0.050**	**0.069**	**0.137**

Figure 4. **Motion transfer results**. Our model is able to accurately transfer the motion from a source video to target meshes, even if the objects are inconsistent.

4.1. Comparison to SOTA

Baselines. We compare our model to four state-of-the-art video-to-4D methods, namely LIM [31], DreamMesh4D (DM4D) [18], V2M4 [4] and ShapeGen4D (SG4D) [50]. We use the official open-source implementations for DreamMesh4D and V2M4. For LIM, the official model is not released, so we trained a re-implementation based on the same dataset as ours. For ShapeGen4D, the code is not publicly available, so we only compare on the qualitative examples from Consistent4D evaluation set. Note that ShapeGen4D work is concurrent to ours but we include their results for completeness.

Figure 5. **Qualitative comparison on Consistent4D [14].** LIM [31] and DM4D [18] tend to produce coarse geometries that lack details. V2M4 [4] and SG4D [50] recover sharper details but leave artifacts and partial drift. In contrast, our model preserves the highest geometric fidelity across frames with a strong temporal consistency. See our supplementary for additional examples.

Quantitative comparison on Objaverse. Since there is no open-source quantitative benchmark, we build our own using Objaverse [7, 8]. Specifically, we evaluate on 32 animated scenes and report three metrics that are complementary. First, we evaluate the *per-frame* 3D reconstruction quality by aligning, for each frame, the predicted mesh with ICP [1] and computing the chamfer distance between ground-truth and prediction (CD-3D). Second, the 4D reconstruction quality is evaluated by aligning the predicted mesh sequence with a global ICP applied on the first mesh, and averaging the chamfer distance (CD-4D). Third, we evaluate motion fidelity with a chamfer-like distance tailored to quantify motion (CD-M). Specifically, after aligning the mesh sequence with a global ICP, we establish nearest neighbor correspondences using the first mesh. Then, for each remaining frame, we evaluate the bidirectional distance between corresponding points. We provide evaluation details in our supplementary.

We report the scores in Table 1. Quantitatively, our method outperforms the baselines across all metrics by a significant margin. In particular, compared to the result obtained by the best prior work for each metric, our model improves CD-3D, CD-4D and CD-M by respectively 21%, 46% and 45%. In addition, it is an order of magnitude faster at inference (3min vs 15–45min for prior works).

Qualitative comparison on Consistent4D. We compare our method on videos from the standard evaluation set of Consistent4D [14] and representative examples are shown in Figure 5. We observe that LIM and DreamMesh4D exhibit reduced shape fidelity with softer details and visible artifacts. Although V2M4 and ShapeGen4D recover sharper details, they also produce artifacts and partial temporal drift. On the contrary, our model yield high-quality meshes with a better temporal coherence and a much stronger motion fidelity. It is worth noting that it does so while being significantly faster. Other examples are shown in our supplementary.

4.2. Additional results

Real-world videos. In Figure 6, we demonstrate our method's robustness by showing qualitative results on real-world videos from DAVIS [27], where a segmentation model was used to isolate the foreground object. Although our model was only trained on synthetic data, it is able to perform accurate 4D reconstructions on these challenging examples. In particular, it is able to capture large-scale motions like a jumping horse (top right) as well as more subtle movements like a bear walking (bottom right). Notably, our rig-free approach excels in handling complex, multi-part animations, as illustrated by the aquarium scene (bottom left).

Motion transfer. In Figure 4, we show our model's ability to transfer motion from a given video representing an object A to a different 3D object B, without explicitly being trained for this task. In particular, we found this process to work well when the semantic correspondences between object A and object B can be established. For example, our model enables us to seamlessly animate a 3D dragon using a casual video of a flying bird (bottom).

Figure 6. **Qualitative results on real videos from DAVIS [27].** Even in these challenging scenarios, our model produces accurate animated 3D meshes, thus demonstrating its ability to handle complex motions, multiple objects and occlusions. See supplemental for more results.

Table 2. **Ablation study.** We evaluate some key components, namely our stage I, both stages I and II, and the pretrained model we started from by using Craftsman [16] instead of TripoSG [17].

Ablation type	CD-3D↓	CD-4D↓	CD-M↓
Full model	**0.050**	**0.069**	**0.137**
w/o stage II	**0.050**	**0.069**	–
w/o stage I & II	**0.050**	0.187	–
w/ Craftsman backbone	0.072	0.117	0.216

Ablation study. Table 2 analyzes the influence of some of our key design choices. First, we remove stage II and thus evaluate the performance of stage I only, which is not able to produce animated 3D meshes. Interestingly, stage II preserves the 3D reconstruction quality while allowing us to predict an animated mesh. Second, we remove both stages; this amounts to running the image-to-3D model (TripoSG) on each frame independently. This experiment shows that stage I is the critical component to obtain accurate 4D reconstructions. Besides, thanks to our minimal modifications, it is worth noting that adding both stages does not harm the 3D reconstruction quality of our backbone. Finally, we replace the TripoSG [17] backbone with Craftsman [16] and observe that the method still achieves competitive performances. We provide additional analysis in our supplementary material.

5. Conclusion

We presented ActionMesh, a fast, feed-forward generative model that produces animated 3D meshes that are topology-consistent and rig-free, directly from diverse inputs. Our key insight relies on temporal 3D diffusion: we extend pretrained 3D diffusion models with a temporal axis to generate a sequence of synchronized shape latents, and then use a temporal 3D autoencoder to translate these shapes into deformations of a reference mesh, yielding an animation with consistent topology. This delivers high-fidelity shape and motion in 3 minutes, enabling rapid iteration and seamless downstream use in texturing and retargeting. We report state-of-the-art geometric accuracy and temporal consistency, demonstrating that our model is a simple, general, and practical path to production-ready animated 3D meshes.

Figure 7. **Limitations.** Typical failure cases arise for videos with topological changes (left) and regions that are occluded either on the reference frame (middle) or during the motion (right).

Limitations and directions. We highlight two typical failure cases in Figure 7 that we discuss next:

- *Topological changes (left).* We assume fixed connectivity and thus changes in topology cannot be modeled. Rather than explicit mesh surgery, a promising direction is to enable topology-aware latent updates that instantiate, fuse, or remove local parts without manual connectivity edits.
- *Strong occlusions (middle, right).* Although our model is able to hallucinate parts that are not visible, it sometimes fail at reconstructing occluded regions, in particular when they are missing from the reference frame or when they disappear during a complex motion.

ActionMesh's ability to lift everyday video into 4D unlocks learning geometric motion priors directly from videos. We believe this closes the loop between large-scale video corpora and mesh-native reasoning, paving the way for richer, more generalizable 4D understanding and generation.

References

[1] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE TPAMI*, 1992. 7, 3

[2] Wei Cao, Chang Luo, Biao Zhang, Matthias Nießner, and Jiapeng Tang. Motion2vecsets: 4d latent vector set diffusion for non-rigid shape reconstruction and tracking. *CVPR*, 2024. 3

[3] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. *CVPR*, 2022. 5

[4] Jianqi Chen, Biao Zhang, Xiangjun Tang, and Peter Wonka. V2m4: 4d mesh animation reconstruction from a single monocular video. *ICCV*, 2025. 2, 5, 6, 7

[5] Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long, Jiashi Feng, and Ping Tan. Dora: Sampling and benchmarking for 3d shape variational auto-encoders. *CVPR*, 2025. 2, 1

[6] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. *ICLR*, 2024. 5

[7] 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. *NeurIPS*, 2023. 7, 2, 3

[8] 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. *CVPR*, 2023. 2, 6, 7, 3

[9] Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T. Barron, and Ben Poole. Cat3d: Create anything in 3d with multi-view diffusion models. *NeurIPS*, 2024. 4, 5

[10] Zhiyang Guo, Jinxu Xiang, Kai Ma, Wengang Zhou, Houqiang Li, and Ran Zhang. Make-it-animatable: An efficient framework for authoring animation-ready 3d characters. *CVPR*, 2025. 3

[11] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models. *ArXiv*, 2022. 4

[12] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *ICLR*, 2024. 2

[13] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. *Int. Conf. Mach. Learn.*, 2021. 4

[14] Yanqin Jiang, Li Zhang, Jin Gao, Weimin Hu, and Yao Yao. Consistent4d: Consistent 360{\deg} dynamic object generation from monocular video. *ICLR*, 2024. 2, 6, 7, 1, 4

[15] Tianhong Li, Huiwen Chang, Shlok Kumar Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. MAGE: MAsked Generative Encoder to Unify Representation Learning and Image Synthesis. *CVPR*, 2023. 5

[16] Weiyu Li, Jiarui Liu, Hongyu Yan, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. Craftsman3d: High-fidelity mesh generation with 3d native generation and interactive geometry refiner. *CVPR*, 2025. 2, 4, 8

[17] 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 TPAMI*, 2025. 2, 4, 5, 8, 1

[18] Zhiqi Li, Yiming Chen, and Peidong Liu. Dreammesh4d: Video-to-4d generation with sparse-controlled gaussian-mesh hybrid representation. *NeurIPS*, 2024. 2, 5, 6, 7

[19] 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. *NeurIPS*, 2024. 4

[20] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *ICLR*, 2023. 4

[21] Isabella Liu, Zhan Xu, Wang Yifan, Hao Tan, Zexiang Xu, Xiaolong Wang, Hao Su, and Zifan Shi. Riganything: Template-free autoregressive rigging for diverse 3d assets. *ACM TOG*, 2025. 3

[22] Xingchao Liu, Chengyue Gong, and qiang liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. *ICLR*, 2023. 4

[23] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH*, 1987. 4

[24] Tom Monnier, Matthew Fisher, Alexei A Efros, and Mathieu Aubry. Share With Thy Neighbors: Single-View Reconstruction by Cross-Instance Consistency. *ECCV*, 2022. 3

[25] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *ArXiv*, 2023. 4

[26] William Peebles and Saining Xie. Scalable diffusion models with transformers. *ICCV*, 2023. 4

[27] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. *CVPR*, 2016. 7, 8, 1, 5

[28] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. DreamGaussian4d: Generative 4d gaussian splatting. *ArXiv*, 2023. 2

[29] Jiawei Ren, Kevin Xie, Ashkan Mirzaei, Hanxue Liang, Xiaohui Zeng, Karsten Kreis, Ziwei Liu, Antonio Torralba, Sanja Fidler, Seung Wook Kim, and Huan Ling. L4gm: Large 4d gaussian reconstruction model. *NeurIPS*, 2024. 3

[30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CVPR*, 2022. 4

[31] Remy Sabathier, Niloy J. Mitra, and David Novotny. LIM: Large interpolator model for dynamic reconstruction. *CVPR*, 2025. 2, 5, 6, 7

[32] Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *ICLR*, 2024. 4

[33] Yahao Shi, Yang Liu, Yanmin Wu, Xing Liu, Chen Zhao, Jie Luo, and Bin Zhou. Drive any mesh: 4d latent diffusion for mesh deformation from video. *ArXiv*, 2025. 3

[34] Yawar Siddiqui, Tom Monnier, Filippos Kokkinos, Mahendra Kariya, Yanir Kleiman, Emilien Garreau, Oran Gafni, Natalia Neverova, Andrea Vedaldi, Roman Shapovalov, and David Novotny. Meta 3D AssetGen: Text-to-Mesh Generation with High-Quality Geometry, Texture, and PBR Materials. *NeurIPS*, 2024. 2

[35] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data. *ICLR*, 2023. 4

[36] Chaoyue Song, Jianfeng Zhang, Xiu Li, Fan Yang, Yiwen Chen, Zhongcong Xu, Jun Hao Liew, Xiaoyang Guo, Fayao Liu, Jiashi Feng, and Guosheng Lin. Magicarticulate: Make your 3d models articulation-ready. *CVPR*, 2025. 3

[37] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2023. 5, 1

[38] 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. *ECCV*, 2024. 2

[39] Tencent Hunyuan3D Team. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details. *ArXiv*, 2025. 2, 4

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017. 4

[41] 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. *NeurIPS*, 2024. 2

[42] Daniel Watson, Saurabh Saxena, Lala Li, Andrea Tagliasacchi, and David J. Fleet. Controlling space and time with diffusion models. *ArXiv*, 2024. 4

[43] Rundi Wu, Ruiqi Gao, Ben Poole, Alex Trevithick, Changxi Zheng, Jonathan T. Barron, and Aleksander Holynski. Cat4d: Create anything in 4d with multi-view video diffusion models. *CVPR*, 2024. 2

[44] Zijie Wu, Chaohui Yu, Yanqin Jiang, Chenjie Cao, Fan Wang, and Xiang Bai. SC4d: Sparse-controlled video-to-4d generation and motion transfer. *ECCV*, 2024. 2

[45] 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. *CVPR*, 2025. 2, 1

[46] Yiming Xie, Chun-Han Yao, Vikram S. Voleti, Huaizu Jiang, and Varun Jampani. Sv4d: Dynamic 3d content generation with multi-frame and multi-view consistency. *ArXiv*, 2024. 2

[47] Zhen Xu, Zhengqin Li, Zhao Dong, Xiaowei Zhou, Richard Newcombe, and Zhaoyang Lv. 4dgt: Learning a 4d gaussian transformer using real-world monocular videos. *NeurIPS*, 2025. 3

[48] Chun-Han Yao, Yiming Xie, Vikram S. Voleti, Huaizu Jiang, and Varun Jampani. Sv4d 2.0: Enhancing spatio-temporal consistency in multi-view video diffusion for high-quality 4d generation. *ArXiv*, 2025. 2

[49] Yuxin Yao, Zhi Deng, and Junhui Hou. Riggs: Rigging of 3d gaussians for modeling articulated objects in videos. *CVPR*, 2025. 3

[50] 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*, 2025. 2, 3, 6, 7

[51] Yuyang Yin, Dejia Xu, Zhangyang Wang, Yao Zhao, and Yunchao Wei. 4dgen: Grounded 4d content generation with spatial-temporal consistency. *ArXiv*, 2024. 2

[52] Jesus Zarzar, Tom Monnier, Roman Shapovalov, Andrea Vedaldi, and David Novotny. Twinner: Shining Light on Digital Twins in a Few Snaps. *CVPR*, 2024. 2

[53] Yifei Zeng, Yanqin Jiang, Siyu Zhu, Yuanxun Lu, Youtian Lin, Hao Zhu, Weiming Hu, Xun Cao, and Yao Yao. STAG4d: Spatial-temporal anchored generative 4d gaussians. *ECCV*, 2024. 2

[54] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM TOG*, 2023. 2, 3

[55] Haiyu Zhang, Xinyuan Chen, Yaohui Wang, Xihui Liu, Yunhong Wang, and Yu Qiao. 4diffusion: Multi-view video diffusion model for 4d generation. *NeurIPS*, 2024. 2, 4

[56] 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 TOG*, 2024. 2, 4

ActionMesh: Animated 3D Mesh Generation with Temporal 3D Diffusion
Supplementary Material

In this supplementary document, we present additional results (Section A), more ablation studies (Section B) and details of implementations (Section C). We encourage readers to consult the video results provided on our webpage: **https://remysabathier.github.io/actionmesh/**.

A. Additional results

Consistent4D comparison. We include additional qualitative comparisons on the standard Consistent4D [14] evaluation set in Fig. 8. Our method produces reconstructions with sharper geometry, fewer temporal artifacts, and more faithful motion than competing approaches. We release videos for all Consistent4D scenes and all baselines, rendered from three viewpoints each, on the supplementary website under the section *Application 1: video-to-4D*.

DAVIS results. We add more qualitative results on real-world videos from DAVIS [27] in Fig. 9. Despite being trained exclusively on synthetic videos, our approach generalizes well to in-the-wild footage, recovering sharp geometry and plausible motion. Videos for all DAVIS examples are provided on the supplementary website under the section *Application 1: video-to-4D*. For the camel example, we also release reconstructed meshes as GLB files in the supplementary folder *meshes*.

Long animation generation. In the standard setting, our model is trained to generate animated sequences of 16 frames. Thanks to its autoregressive modeling, we can apply our model to longer videos by recursively rolling out predictions: the last 3D output of one inference is fed back as conditioning for the next one. In the supplementary website, we showcase video-to-4D reconstructions of sequences with 61 frames, obtained from a single standard 16-frame inference pass followed by three additional autoregressive passes. These examples illustrate that our model can maintain coherent geometry and motion over significantly extended time horizons beyond its training sequence length. Videos are listed under the section *Application 1: video-to-4D*.

Video results. We provide on our supplementary website a comprehensive set of videos illustrating the five main applications of our model. **(i) Video-to-4D:** we show reconstructions on the Consistent4D [14] evaluation set and on real-world DAVIS [27] videos (see paragraphs above). **(ii) {3D+text}-to-4D:** given a static 3D textured mesh and a motion description, we animate well-known benchmark meshes (the armadillo and the cow) as well as additional textured meshes produced by an external 3D generative model. Since our method outputs animated meshes with fixed topology, textures remain coherent and are consistently propagated throughout the entire sequence. **(iii) {Image+text}-to-4D:** starting from single images drawn from standard 3D evaluation sets used in TripoSG [17], Dora [5], and Trellis [45], we manually attach a short motion description to each image (*e.g.*, the astronaut "dancing", the mushroom "singing opera") and generate an animated 3D mesh out of them. **(iv) Text-to-4D:** we generate animated meshes directly from textual prompts describing both the object and its motion, such as "an octopus playing maracas". **(v) Motion transfer:** we include motion transfer videos with two input motions, each applied on two different meshes.

Table 3. **Ablation study - Temporal 3D denoiser.**

Ablation type	CD-3D↓	CD-4D↓
Full model	**0.050**	**0.069**
w/o rotary embedding	0.054	0.084
w/o masked modeling	0.062	0.116

Table 4. **Ablation study - Temporal 3D autoencoder.**

Ablation type	CD-M↓
Full model	**0.137**
w/o normals	0.148
w/o $\{t_{\text{src}}, t_{\text{tgt}}\}$ in self-attentions	0.151

B. Additional ablation studies

Temporal 3D diffusion (stage I) We report additional ablations of the temporal 3D diffusion model in Tab. 3. First, we study the importance of injecting relative frame information inside the inflated self-attention layers. In our default setting, we add rotary positional embedding [37] of the relative frame index in the inflated self-attention layers. We train a variant where inflated attention is kept but all temporal rotary embeddings are removed. On our Objaverse evaluation set, this leads to a clear degradation of both CD-3D and CD-4D, confirming that explicit temporal encoding is critical for stable, temporally coherent 4D generation. Second, we ablate the masked generation mechanism. In the default

configuration, for video-to-4D we compute a reference mesh latent $\mathbf{z}_1^*$ from an off-the-shelf image-to-3D model and inject it as a clean source latent while generating the remaining (masked) latents. We train a model where this mechanism is disabled and the diffusion model is conditioned only on video frames, with no noise-free latents. This variant cannot support several applications enabled by our masking strategy, including {3D+text}-to-4D, {image+text}-to-4D and autoregressive long-horizon generation. Beyond this loss of functionality, it also underperforms our full model on video-to-4D reconstruction metrics, indicating that leveraging a strong image-to-3D prior through masked modeling both broadens the application scope and improves geometric and motion fidelity.

Temporal 3D autoencoder (stage II) We report ablations of the temporal 3D autoencoder in Tab. 4. First, we study the impact of augmenting query 3D points with surface normals. In our default configuration, each query point is represented by its position and normal, a choice motivated by the need to disambiguate deformations of points that are spatially close but topologically distant on the surface. Removing normals and using positions yields a drop in performance. Second, we ablate how the source and target timesteps $(t_{\text{src}}, t_{\text{tgt}})$ that parameterize the deformation field are injected into the model. By default, we treat t_{src} and t_{tgt} as an additional token concatenated to the shape latents, so that they are jointly processed in the self-attention layers of the temporal 3D autoencoder. We compare this with a variant where $(t_{\text{src}}, t_{\text{tgt}})$ are instead appended as extra features to the query points (alongside position and normals). While this latter design has a practical advantage—enabling substantial speed-ups when predicting multiple deformation fields for the same shape tokens, since the post–self-attention context can be cached and reused—we observe a degradation in performance.

Autoregressive generation. We ablate our autoregressive design choices in Tabs. 5 and 6. In Table 5, we vary the number of keyframes N used to train the temporal 3D denoiser (stage I) and the temporal 3D autoencoder (stage II). In the default configuration, both stages operate on sequences of 16 keyframes. We additionally train variants of each stage with 4 or 8 keyframes, and evaluate some pairwise combinations of (stage I, stage II) configurations on our evaluation set of 16-frame sequences. For models trained with fewer than 16 keyframes, we reconstruct the sequence by splitting it into shorter chunks and we process them autoregressively. We observe that having a larger sequence length is particularly important in stage I: increasing the number of keyframes in the denoiser (autoencoder fixed) leads to a significant improvement across all metrics. In contrast, stage II is less sensitive to this choice. In Table 6, we study the design choices of our autoregressive regime. In the standard setting,

Table 5. **Ablation study - Number of frames.** We train the temporal 3D diffusion model and the temporal 3D autoencoder with various number of frames N and compare reconstructions on 16 keyframes.

Stage I - N	Stage II - N	CD-3D ↓	CD-4D ↓	M-CD ↓
4	16	0.057	0.091	0.187
8		0.055	0.086	0.176
16	4	0.051	0.073	0.144
	8	0.050	0.073	0.144
4	4	0.056	0.091	0.187
8	8	0.055	0.085	0.175
16	16	**0.050**	**0.069**	**0.137**

Table 6. **Ablation study - Autoregressive context window.** We evaluate our model trained with $N = 16$ keyframes on sequences of 31 timesteps for different context window c_{w}.

Stage I - c_{w}	Stage II - c_{w}	CD-3D ↓	CD-4D ↓	M-CD ↓
1	1	0.051	0.098	0.195
4		0.051	0.094	0.190
8		0.051	0.090	0.185
1	1	0.051	0.098	0.195
	4	0.050	0.097	0.196
	8	0.051	0.098	0.196
1	1	0.051	0.098	0.195
4	4	0.050	0.094	0.191
8	8	0.051	0.091	0.187

we process long videos by splitting them into consecutive chunks; the first chunk is reconstructed as usual, and for each subsequent chunk we feed the last reconstructed 3D output as additional input (reference) along with the current video segment. In practice, both the temporal 3D denoiser and the temporal 3D autoencoder can accept one or more reference meshes as input, thus defining a *context window* c_{w} for our model. Increasing the number of context frames, however, implicitly increases the total number of inference steps, since fewer new frames are generated per pass. On sequences of 31 timesteps, we find that enlarging the context window has limited effect on reconstruction metrics, while incurring additional computational cost. We therefore set $c_{\text{w}} = 1$ in both stages in our final model to maximize efficiency.

C. Implementation details

Temporal 3D diffusion. We train with AdamW (learning rate 1×10^{-4}, weight decay 1×10^{-2}) in bfloat16 mixed precision, using a global batch size of 96 for 170,000 steps. The dataset comprises 13,200 animated object sequences drawn from Objaverse [8], Objaverse-XL [7], and an internal corpus. **Inputs.** (i) *Input frames:* for each armature-driven sequence (at least 16 and up to 128 keyframes), we render 16 viewpoints per keyframe with uniformly spaced azimuths and elevations in $[40°, 85°]$. (ii) *Input point-clouds:* for each sequence we construct a canonical point cloud $\mathbf{P} \in \mathbb{R}^{N_{\text{p}} \times 6}$ with $N_{\text{p}} = 500{,}000$ points (XYZ and normals) together with the armature configurations over keyframes; we then

deterministically deform $\mathbf{P}$ to each keyframe k to obtain $\mathbf{P}_k$, aligned with the rendered image $\mathbf{I}_k$. Training uses 16-frame sequences: each batch contains a video clip $(16\times H\times W\times 3)$ and the aligned sequence of deformed point clouds with normals $(16 \times N_{\text{p}} \times 6)$. We extend TripoSG image preprocessing to video by computing the union object bounding box over the whole sequence and resizing every frame so that this box occupies 90% of the height or width; applying a consistent crop across frames avoids spurious scale/translation cues. Consistent with [50], we find that deforming a single canonical point cloud over time—rather than re-sampling points at each timestep—is critical for stable training of our temporal 3D diffusion model. We encode point clouds with a frozen $\mathcal{E}_{\text{3D}}$ to obtain latents; during training, we randomly select $N_{\text{S}} \in \{1, 2, 3\}$ latents as ground-truth conditioning and train on the remaining latents using the standard flow-matching objective. Due to memory constraint, we train with $T = 1024$ tokens, however, at inference, $T = 2048$.

Temporal 3D autoencoder. We train the temporal 3D autoencoder with the same optimization hyperparameters and bfloat16 mixed-precision setup as temporal 3D diffusion, using the identical corpus of 13,200 animated object sequences. Training again operates on 16-frame clips: for each sequence, we provide the model with the deformed point cloud trajectory, where each point cloud contains XYZ coordinates and normals at keyframe k. In addition, we sample a source and a target timestep $(t_{\text{src}}, t_{\text{tgt}})$ uniformly between the first and last keyframe and extract a set of surface points by deforming randomly sampled mesh points from t_{src} to t_{tgt}. The temporal 3D autoencoder is trained to regress these deformations from the input 3D latents on the keyframes, minimizing an ℓ_2 loss between predicted and ground-truth positions. Unlike the temporal 3D diffusion stage, this training setup does not rely on latents from a temporally synchronized canonical point cloud: surface points can be re-sampled independently at each timestep, simplifying data preparation and decoupling the autoencoder from the canonical-deformation assumption.

Quantitative evaluation. Our quantitative benchmark is composed of 32 animated scenes from Objaverse [7, 8], each comprising a textured, rigged mesh with a predefined animation sequence. For every scene we select the first $N = 16$ keyframes and render from a fixed camera with azimuth 70°. Each mesh is converted to a watertight representation and the animation is rigidly normalized to a canonical cube $[-1, 1]^3$, yielding synchronized sequences $\{\mathbf{V}_k, \mathbf{F}\}_{k=1}^N$ and $\{\mathbf{I}_k\}_{k=1}^N$.

Given watertight ground-truth meshes $\{\mathcal{M}_k\}_{k=1}^N$ and predicted meshes $\{\widehat{\mathcal{M}}_k\}_{k=1}^N$, we uniformly sample $P = 100{,}000$ surface points from each mesh (area-weighted). Let $\mathbf{S}_k = \{\mathbf{x}_{k,i}\}_{i=1}^P \subset \mathbb{R}^3$ and $\hat{\mathbf{S}}_k = \{\hat{\mathbf{x}}_{k,i}\}_{i=1}^P$ denote the sampled point sets. Prior to distance computation, we rigidly align predictions to the ground truth using Iterative Closest Point (ICP) [1]. We report two alignment protocols: *ICP3D* (frame-wise), where for each k, we estimate $(\mathbf{r}_k, \mathbf{t}_k) \in SO(3) \times \mathbb{R}^3$ that aligns $\hat{\mathbf{S}}_k$ to $\mathbf{S}_k$; and *ICP4D* (sequence-wise), where we estimate $(\mathbf{r}_1, \mathbf{t}_1)$ on $k = 1$ and apply it to all frames giving $\{\mathbf{r}_1\hat{\mathbf{x}}_{k,i} + \mathbf{t}_1\}_i$. We denote by $\bar{\mathbf{S}}_k^{\text{3D}}$ and $\bar{\mathbf{S}}_k^{\text{4D}}$ the aligned predictions under the chosen protocol. In practice, we use the gradient-based implementation of [24] to align shapes with ICP.

Chamfer-based shape metrics. The symmetric Chamfer distance (CD) between the ground-truth and predicted point sets at frame k is defined as

$$\text{CD}(\mathbf{S}_k, \hat{\mathbf{S}}_k) = \frac{1}{P}\sum_{i=1}^{P}\Big[\min_{\hat{\mathbf{x}}\in\hat{\mathbf{S}}_k}\|\mathbf{x}_{k,i}-\hat{\mathbf{x}}\|_2^2+\min_{\mathbf{x}\in\mathbf{S}_k}\|\mathbf{x}-\hat{\mathbf{x}}_{k,i}\|_2^2\Big]. \tag{2}$$

We define CD-3D as the temporal average of the symmetric Chamfer distance computed *per frame* under ICP3D alignment.

$$\text{CD-3D} = \frac{1}{K}\sum_{k=1}^{N}\text{CD}(\mathbf{S}_k, \bar{\mathbf{S}}_k^{\text{3D}}). \tag{3}$$

When the alignment is ICP4D (single transform from $k = 1$), we use $\bar{\mathbf{S}}_k^{\text{4D}}$ in the equation above and we refer to the resulting average as CD-4D.

Motion Chamfer distance. Analogous to Chamfer distance, after aligning sequence-wise using ICP4D, we establish two directed nearest-neighbor maps (GT $\rightarrow$ PRED and PRED $\rightarrow$ GT) at the first frame and keep them fixed for the whole sequence:

$$\sigma_i = \operatorname*{argmin}_{j\in\{1,\dots,P\}}\left\|\mathbf{x}_{1,i} - \hat{\mathbf{x}}_{1,j}\right\|_2^2, \tag{4}$$

$$\tau_i = \operatorname*{argmin}_{j\in\{1,\dots,P\}}\left\|\mathbf{x}_{1,j} - \hat{\mathbf{x}}_{1,i}\right\|_2^2. \tag{5}$$

We then propagate these correspondences to every frame k without recomputing nearest neighbors, and refer to the resulting average as CD-M:

$$\text{CD-M} = \frac{1}{NP}\sum_{k=1}^{N}\sum_{i=1}^{P}\left\|\mathbf{x}_{k,i} - \hat{\mathbf{x}}_{k,\sigma_i}\right\|_2^2 + \left\|\mathbf{x}_{k,\tau_i} - \hat{\mathbf{x}}_{k,i}\right\|_2^2. \tag{6}$$

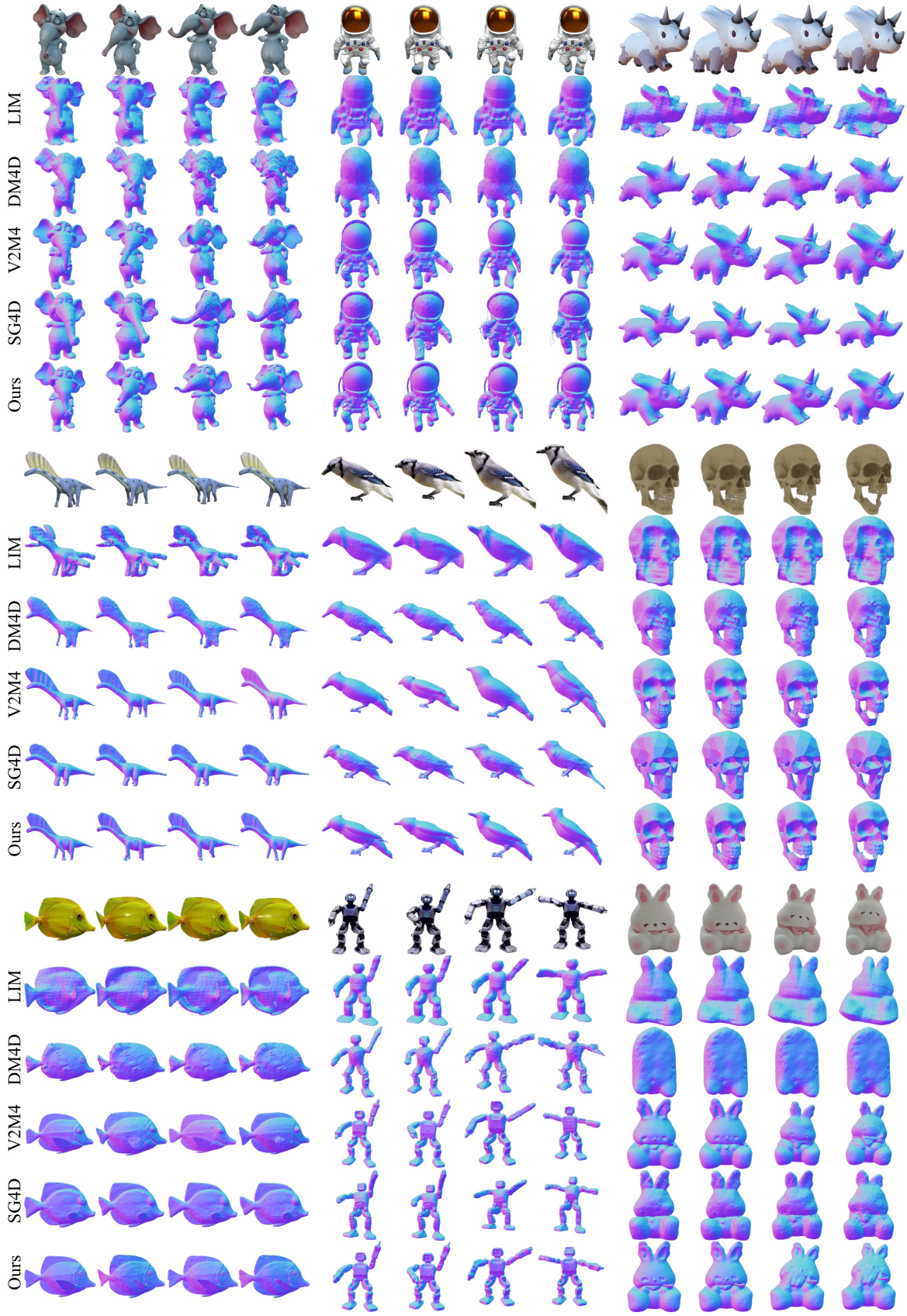

Figure 8. **Additional qualitative comparison on Consistent4D [14].**

Figure 9. **Additional qualitative results on real videos from DAVIS [27].**