# BUMBLEBEE: 4D MORPHING WITH EXACT DUAL-ENDPOINT CONTROL

**Anonymous authors**
Paper under double-blind review

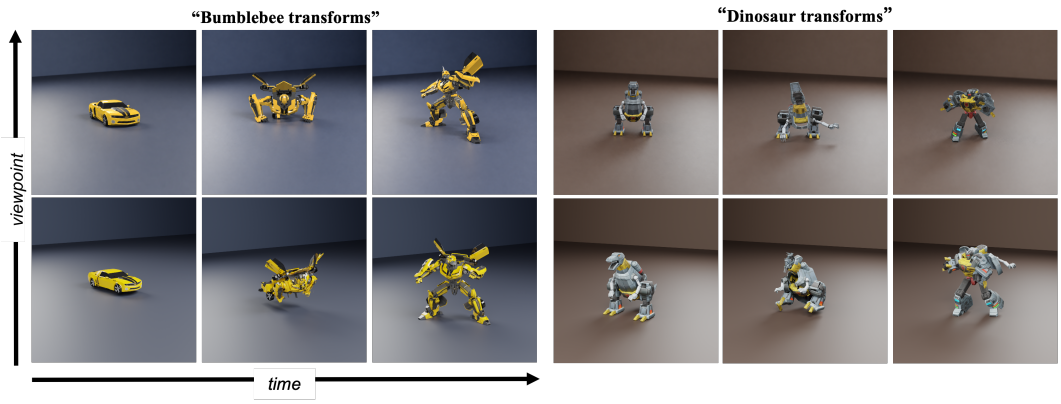



Figure 1: **Video-guided 4D morphing between exact keyframe meshes with Bumblebee.** Left: car-to-robot. Right: robot-to-dinosaur. Columns progress from the given source mesh to an intermediate frame generated from both meshes and the transformation video, and then to the given target mesh. The two rows show the same transformation from two viewpoints. Outputs preserve fixed topology, exact endpoints, and the original textures without re-projection or rigging.

## ABSTRACT

We study video-guided 4D morphing. Given a source mesh, a target mesh, and a reference video, the task is to generate a mesh sequence with fixed topology. The first and last frames must exactly match the given meshes, and the intermediate motion must follow the reference video. Existing video-to-4D methods mainly focus on reconstructing and generating dynamic meshes. They do not target exact matching of two given endpoints. Even when keyframes are provided as generation conditions, decoded meshes may still deviate from the given geometry. Directly correcting these deviations vertex by vertex can stretch and twist mechanical parts. We propose Bumblebee, a training-free framework built on a frozen video-to-4D generative model. It requires neither skeletal rigging nor optimization of model parameters for each sequence. The two keyframe meshes serve as observed frames at every denoising step to guide the generation of intermediate motion. We use positional differences between decoded meshes and the given keyframes as geometric feedback. This feedback corrects the trajectory frame by frame, ensuring exact matching at both endpoints. For mechanical assets, we correct motion through whole-part translation and rotation, then compensate for the remaining endpoint deviations. This satisfies the endpoint requirements while reducing stretching and twisting of the parts. Experiments on Motion80 and ActionBench show that Bumblebee outperforms the compared video-driven generation and shape interpolation baselines in geometry and appearance. Experiments on mechanical assets further show that part rigidity constraints better preserve part shapes than direct vertex-wise correction.

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

# 1 INTRODUCTION

Generating dynamic 3D assets is important for visual effects, games, and virtual character animation (Li et al., 2024; Wu et al., 2025; Sabathier et al., 2026). These applications require geometric representations of how objects move and change shape over time, allowing their motion to be rendered from different viewpoints and incorporated into 3D scenes (Li et al., 2024; Chen et al., 2025a; Wu et al., 2025; Jiang et al., 2026; Kwon et al., 2026). To support this process, 3D morphing methods generate intermediate shapes that connect source and target states (Eisenberger et al., 2021; Buonomo et al., 2025; Yang et al., 2025; Yin et al., 2025). Video-driven 4D mesh generation methods use reference videos to produce time-varying mesh sequences that capture object motion and deformation (Li et al., 2024; Chen et al., 2025a; Shi et al., 2025; Jiang et al., 2026; Sabathier et al., 2026). These directions support the creation of animated 3D geometry from shape specifications and visual motion references (Yin et al., 2025; Shi et al., 2025; Sabathier et al., 2026).

However, existing methods face limitations when animating an existing 3D asset. Shape interpolation methods connect endpoint meshes through geometric criteria (Alexa et al., 2000; Buonomo et al., 2025). Different motions can connect the same endpoints, so matching them alone does not determine the order, direction, or timing of intermediate movements. The transition may therefore differ from the intended action. Video-driven methods use reference videos to guide these movements, but do not guarantee that the generated meshes exactly match the given endpoint geometry (Chen et al., 2025a; Sabathier et al., 2026). Such deviations can alter the asset's shape and cause jumps in shape or pose when the transition joins existing animation segments. For assets with rigid parts, stretching or twisting those parts during the transition also changes their intended structure. These limitations raise a central question: *How can we generate a transition that follows the motion in a reference video, connects two given mesh states exactly, and preserves the shapes of the asset's rigid parts?*

To address the first challenge, we draw on methods that use known measurements to correct predictions from pretrained generative models (Kawar et al., 2022; Wang et al., 2023; Chung et al., 2023; Rout et al., 2023; Song et al., 2024). In DE-4DM, the given meshes specify the required position of every endpoint vertex. Their differences from the generated endpoints therefore provide geometric feedback for correction. We examine whether this feedback can overcome the limitations of keyframe conditioning. Our analysis shows that even directly encoding and decoding ground-truth meshes leaves geometric deviations. Yet a simple correction based on the endpoint differences makes both endpoints exact and improves average geometric accuracy without retraining. These findings support using the frozen model to generate motion and geometric feedback to enforce precise endpoint agreement.

We next examine whether such correction also preserves the shapes of mechanical parts (Sorkine et al., 2007; Sumner et al., 2007). A rigid part moves by changing its position and orientation while retaining its internal geometry (Lynch & Park, 2017; Huang et al., 2016). This motivates treating these quantities as its motion state and comparing corrections to part states and individual vertex positions. Under identical endpoint constraints, both achieve exact endpoint agreement. However, vertex-wise correction leaves stretched or twisted parts in intermediate frames, whereas whole-part translation and rotation substantially reduce these distortions. These results show that endpoint accuracy and part preservation depend on different aspects of the correction: endpoint differences specify the required adjustments, while part structure determines how to apply those adjustments.

Based on these observations, we propose Bumblebee, a training-free framework for DE-4DM. It uses a frozen video-to-4D model (Sabathier et al., 2026) to generate motion conditioned on the reference video and both endpoint meshes. To resolve the geometric deviations after decoding, we introduce geometric feedback correction, which uses the endpoint differences to compute adjustments across the trajectory and makes both endpoint meshes exact. To reduce structural distortion in mechanical assets, we estimate each part's position and orientation from the generated trajectory (Kabsch, 1976; Umeyama, 1991) and correct these motion states through whole-part translation and rotation. This preserves the internal geometry of each part during motion adjustment (Lynch & Park, 2017; Gao et al., 2025; Dong et al., 2025; Huang et al., 2016). A final compensation removes any remaining endpoint deviations, ensuring exact keyframe agreement. Together, these components provide precise endpoint control and reduce part distortion while retaining the asset's topology and texture mapping.

We evaluate Bumblebee on Motion80 (Chen et al., 2026) and ActionBench (Sabathier et al., 2026), and assess part shape preservation on mechanical transformation assets and robot models from MuJoCo Menagerie. Compared with ActionMesh (Sabathier et al., 2026) given the same reference

video and both endpoint meshes, Bumblebee reduces Chamfer distance by 7.2% on Motion80 and CD-4D by 12.5% on ActionBench, while matching both endpoints exactly. On mechanical assets, correcting part positions and orientations substantially reduces stretching and twisting remaining after vertex-wise correction. Ablations examine the contributions of geometric feedback and part motion correction, while static and shuffled video experiments verify the role of video guidance in determining intermediate motion (Section 5). Our main contributions are summarized below:

- We formulate *dual-endpoint video-guided 4D morphing* (DE-4DM), a setting that combines reference video guidance with exact agreement to two given endpoint meshes. The generated sequence must retain the asset's topology and textures and preserve the shapes of its rigid parts.
- We show that endpoint feedback can correct geometric deviations after decoding, but correcting individual vertices can still distort mechanical parts. These observations motivate Bumblebee, which combines geometric feedback with corrections to part positions and orientations, without model retraining. We further estimate part segmentation from relative motion between the two keyframes, enabling part-rigidity constraints for trajectory correction without manual part annotations.
- We conduct evaluations on two public benchmarks and mechanical assets. Comparisons and controlled ablations demonstrate improved geometric and motion accuracy, exact endpoint agreement, and reduced part distortion, and establish the contributions of the two correction components.

## 2 LEARNING SETUPS

**3D Meshes and Motion.** A 3D mesh represents an object's surface using vertices and faces. We write meshes as $M = (V, F)$, where $N$ is the number of vertices, $V \in \mathbb{R}^{N\times 3}$ is the matrix of their three-dimensional coordinates, and $F$ specifies the vertex indices forming each face. Each row of $V$ gives one vertex's position. A 4D mesh sequence describes how this surface changes over time. Following existing work, we consider sequences with fixed vertex indexing and face connectivity, while vertex positions vary to represent motion and deformation (Sabathier et al., 2026; Chen et al., 2025a).

**Reference Video and Endpoint Meshes.** Let $T$ denote the number of frames and $t \in \{0, \dots, T-1\}$ the frame index. The reference video is the ordered sequence $\mathcal{I} = (I_0, \dots, I_{T-1})$, where $I_t$ is the image at frame $t$. Let $V_s, V_e \in \mathbb{R}^{N\times 3}$ denote the vertex coordinates of the same asset in its starting and ending states, respectively. The two given keyframe meshes are $M_s = (V_s, F)$ and $M_e = (V_e, F)$. They share the face connectivity $F$ and use the same vertex ordering, so corresponding rows of $V_s$ and $V_e$ refer to the same point on the asset. We denote the complete input of the morphing task by

$$X = (\mathcal{I}, M_s, M_e). \tag{1}$$

The two meshes specify the geometry at the first and last frames. The video indicates how the asset should move between these states, including the order and timing of its movements. Video frame $I_t$ corresponds to output frame $t$. No intermediate 3D meshes are provided as additional input.

**Problem 1 (Dual-Endpoint Video-Guided 4D Morphing).** Given $X$, dual-endpoint video-guided 4D morphing (DE-4DM) aims to generate a mesh sequence that follows the reference video and connects the two given mesh states. Let $\widehat{V}_t \in \mathbb{R}^{N\times 3}$ denote the generated vertex coordinates at frame $t$, and let $\widehat{M}_t = (\widehat{V}_t, F)$ denote the corresponding mesh. The output is the ordered sequence

$$\widehat{\mathcal{M}} = (\widehat{M}_0, \dots, \widehat{M}_{T-1}). \tag{2}$$

Its starting and ending geometry must therefore satisfy the following two keyframe constraints

$$\widehat{V}_0 = V_s, \qquad \widehat{V}_{T-1} = V_e. \tag{3}$$

These equalities require every endpoint vertex to match its given position exactly. The intermediate meshes must form a coherent transition whose movements follow the order and timing shown in the video. All frames retain the input face connectivity $F$, the original texture, and its surface mapping.

Preserving connectivity alone does not preserve shape: vertices can remain connected while moving apart and stretching a part (Sorkine et al., 2007; Oehri et al., 2025). For assets composed of rigid mechanical parts, the sequence should therefore also preserve each part's shape as it moves and rotates, keeping unintended stretching and twisting to a minimum throughout the whole sequence.

# 3 OBSERVATIONS

## 3.1 KEYFRAME CONDITIONING AND GEOMETRIC FEEDBACK

The first challenge is to match the given endpoint meshes exactly while generating motion guided by the video. ActionMesh incorporates keyframes through encoded representations (Sabathier et al., 2026). Whether this provides exact geometric control depends on whether decoding preserves the given vertex positions. (Rombach et al., 2022) In DE-4DM, these positions are known, so differences between the generated and given endpoints provide explicit geometric errors. Prior work uses known measurements to correct predictions from pretrained generative models (Wang et al., 2023; Song et al., 2024). This motivates us to further investigate whether end-point error can similarly be leveraged as feedback to correct the generated mesh sequences, without retraining the generative model.

**Reconstruction accuracy.** We first test whether geometric errors arise even without generating unknown motion. We directly encode ground-truth 3D sequences and reconstruct them with the frozen temporal decoder, without applying any correction. Comparing the reconstructed geometry with the input isolates errors in the encoding and decoding pipeline. Figure 2 shows local deviations despite the use of ground-truth inputs. The error distributions on Motion80, ActionBench, and robot sequences show that these deviations are not limited to the illustrated examples (Appendix L). Thus, even representations obtained directly from the desired geometry do not ensure its exact reconstruction in the tested pipeline.

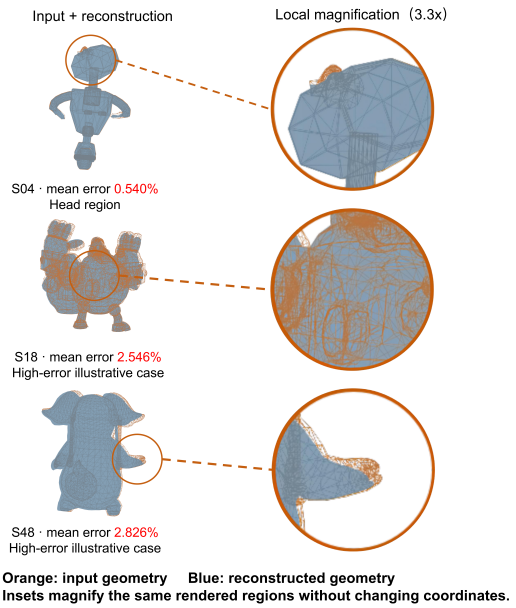



Figure 2: Taking the Motion80 dataset as an example, we demonstrate the reconstruction errors of encoded/decoded mesh sequences. Overlaying the input (orange) and reconstructed (blue) meshes highlights surface deviations, with close-ups revealing detail discrepancies. S18 and S48 are representative high-error cases.

**Correction using endpoint feedback.** We next test whether the known endpoint geometry can correct these output deviations. We generate sequences conditioned on the reference video and both endpoint meshes, keeping the generative model frozen. For each sequence, we compute the differences between its decoded endpoints and the given meshes. We then linearly interpolate these differences over time and add the resulting corrections to the decoded vertex positions. This simple test uses only the two given meshes for correction. Intermediate ground-truth meshes serve only for the final evaluation.

The corrected endpoints match the given vertex positions to numerical precision (Appendix C). The correction also improves average geometric accuracy: on Motion80, the mean Chamfer distance decreases from 0.0626 to 0.0581 (Table 9). Thus, the benefit extends beyond satisfying the endpoint equalities to improving the sequence-level geometric score averaged over all 80 identities.

**Implication.** Keyframe conditioning and geometric feedback play complementary roles. The former guides motion generation, while the latter directly corrects deviations from known endpoint coordinates. These results support using explicit geometric feedback to obtain precise endpoint control without retraining the generative model or any per-sequence optimization.

## 3.2 ENDPOINT ACCURACY AND PART SHAPE PRESERVATION

The first analysis addresses endpoint accuracy. The second challenge is to preserve the shapes of mechanical parts throughout the transition. Correcting vertices individually can change the distances between vertices within the same part, even when both endpoints become exact. A rigid part, however,

can move by changing its position and orientation while retaining its internal geometry (Sorkine et al., 2007; Oehri et al., 2025). This motivates treating position and orientation as the part's motion state and examining whether correcting these states reduces the distortions left by vertex-wise correction.

**Comparing vertex and part motion correction.** On mechanical transformation assets, we compare two corrections applied to the same decoded trajectories, with the same reference videos and endpoint meshes. The first applies the vertex-wise endpoint correction examined above. The second estimates each part's position and orientation along the decoded trajectory and adjusts these motion states using the given endpoints. It then places the original part geometry at the adjusted positions and orientations. Any remaining endpoint differences are compensated so that both variants satisfy the same endpoint constraints and differ only in how the correction is distributed within each part.

We evaluate endpoint agreement and part distortion separately. For distortion, we fit a rotation and translation (Kabsch, 1976) from each source part to its generated counterpart and measure the remaining vertex error, normalized by the bounding-box diagonal. This rigid-fit residual measures the deformation that cannot be explained by whole-part motion. We also measure relative changes in mesh edge lengths, which directly expose local stretching of the surface (Appendix C).

**Observed differences.** Although both corrections satisfy the endpoint constraints, vertex-wise correction leaves substantial stretching and twisting within the sequence. Correcting part motion reduces these distortions: the median rigid-fit residual decreases from 0.330 to 0.040 on the transformation split. The edge-length measurements also show substantially less deformation after part motion correction (Appendix C). The remaining residual of 0.040 is even below the 0.047 measured on the ground-truth rigs, so the corrected parts stay as rigid as the original assets throughout the sequence.

**Implication.** Exact endpoint agreement does not ensure that intermediate parts retain their shapes. For mechanical assets, the correction should therefore account for how vertices move together within each part. The comparison supports correcting part positions and orientations to reduce deformation while maintaining the endpoint constraints exactly at the first and last frames of each sequence.

Taken together, these observations provide two design principles. Known endpoint differences supply geometric feedback for precise matching, and part structure guides how corrections should be applied to mechanical assets. They motivate Bumblebee, which combines a frozen model for video-guided motion generation with geometric feedback correction and correction of part motion. Section 4 describes how these components produce a complete, textured mesh sequence with fixed topology.

# 4 METHODOLOGY

## 4.1 GEOMETRY-FEEDBACK-DRIVEN KEYFRAME ALIGNMENT

Observation 1 shows that positional deviations exist between the decoded meshes and the given geometry. We use the differences between the generated results and the given keyframes as geometric feedback to guide output correction without modifying the frozen generative model.

We first perform **geometric feedback extraction.** Let $V_s$ and $V_e$ denote the given source and target vertex positions, respectively, and let $\widetilde{V}_0$ and $\widetilde{V}_{15}$ denote the decoded vertex positions at the first and last frames. For the 16-frame sequence in the figure, we denote the geometric feedback at the two endpoints by $R_s$ and $R_e$, respectively, and define them as the vertex-wise differences

$$R_s = V_s - \widetilde{V}_0, \qquad R_e = V_e - \widetilde{V}_{15}. \tag{4}$$

After extracting the feedback in this way, we convert it into vertex compensation for each frame, that is, we use this geometric feedback to correct the trajectory. Let $B_t$ denote the correction at frame $t$. The endpoint arrows in Figure 3 correspond to the two boundary corrections of the trajectory

$$B_0 = R_s, \qquad B_{15} = R_e. \tag{5}$$

To propagate the endpoint feedback to the intermediate frames, we weight the compensation at the two endpoints according to the temporal position. Let $\tau_t = t/15$ be the normalized time. Then

$$B_t = (1 - \tau_t)R_s + \tau_t R_e. \tag{6}$$

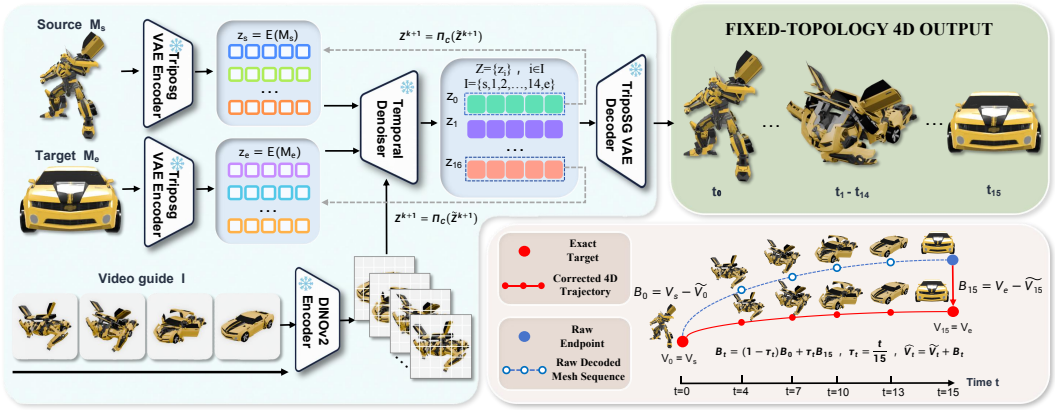



Figure 3: Bumblebee: video-guided 4D morphing between two exact keyframe meshes on a frozen 4D prior. Encoders, denoiser and decoder are frozen. Both meshes enter the generative model as observed frames at every denoising step. The geometric feedback correction (Section 4.1) turns the decoded trajectory (blue) into the corrected trajectory (red), whose first and last frames equal the given meshes exactly. For mechanical assets, the correction acts on whole parts (Section 4.2).

Let $\widehat{V}_t$ denote the corrected vertex positions. Adding the compensation to the decoded result at the corresponding frame keeps the vertex ordering and face connectivity of the input and gives

$$\widehat{V}_t = \widetilde{V}_t + B_t. \tag{7}$$

Thus, the blue trajectory becomes the red trajectory through frame-by-frame compensation and satisfies the two endpoint constraints of Problem 1 exactly at the first and last frames

$$\widehat{V}_0 = V_s, \qquad \widehat{V}_{15} = V_e. \tag{8}$$

Geometric feedback adjusts the decoded sequence frame by frame to match both endpoint meshes exactly. We interpolate the endpoint corrections linearly in time. This does not constrain the corrected motion to a straight or monotonic path. In fixed mesh coordinates with uniformly spaced frames, the correction has zero second temporal difference, $B_{t+1} - 2B_t + B_{t-1} = 0$ for $t = 1, \ldots, 14$. Thus, the linear vertex correction preserves the discrete acceleration of the decoded trajectory.

### 4.2 Morphing Correction under Part Rigidity Constraints

Observation 2 shows that vertex-wise endpoint correction can distort mechanical parts. We therefore represent each part by its position and orientation and restrict the structural correction to whole-part translation and rotation, which cannot stretch or shear the geometry inside a part.

Let $\mathcal{P}_k$ be the vertex set of part $k$, a connected component of the source mesh. Let $v_{s,i}$, $v_{e,i}$ and $\widetilde{v}_{t,i}$ be the rows of $V_s$, $V_e$ and $\widetilde{V}_t$. Let $c_{s,k}$ be the source part center and $q_{k,i} = v_{s,i} - c_{s,k}$ the offset of vertex $i$. For every decoded frame we estimate the part state with a least-squares rigid fit

$$(c_{t,k}, Q_{t,k}) = \left( \frac{1}{|\mathcal{P}_k|} \sum_{i \in \mathcal{P}_k} \widetilde{v}_{t,i},\ \arg\min_{Q \in SO(3)} \sum_{i \in \mathcal{P}_k} \| Q q_{k,i} + c_{t,k} - \widetilde{v}_{t,i} \|_2^2 \right). \tag{9}$$

The given keyframes define the translation feedback $d_{s,k} = c_{s,k} - c_{0,k}$ and $d_{e,k} = c_{e,k} - c_{15,k}$, and the rotation feedback $D_{s,k} = Q_{0,k}^{\mathsf{T}}$ and $D_{e,k} = Q_{e,k} Q_{15,k}^{\mathsf{T}}$. Here $(c_{e,k}, Q_{e,k})$ is the state of Eq. (9) computed on $V_e$. We use the smooth weight $w_t = 10\tau_t^3 - 15\tau_t^4 + 6\tau_t^5$. Its first and second derivatives vanish at both ends, so the correction starts and stops without a jump in velocity or acceleration. Every part is then placed rigidly with one rotation and one translation shared by its vertices,

$$\overline{v}_{t,i} = \Phi(D_{e,k}, w_t)\, \Phi(D_{s,k}, 1 - w_t)\, Q_{t,k}\, q_{k,i} + c_{t,k} + (1 - w_t)\, d_{s,k} + w_t\, d_{e,k}, \qquad i \in \mathcal{P}_k, \tag{10}$$

where $\Phi(D, w)$ scales the rotation angle of $D$ by $w$. It follows the geodesic from $I$ to $D$ in $SO(3)$, so every corrected $\overline{Q}_{t,k}$ is a rotation and the placement never shears or scales a part. Because $\Phi(D, 0) = I$ and $\Phi(D, 1) = D$, the placement at the two endpoints of the sequence gives

$$\overline{v}_{0,i} = v_{s,i}, \qquad \overline{v}_{15,i} = Q_{e,k}\, q_{k,i} + c_{e,k}. \tag{11}$$

The source is reached exactly. The target residual $r_{k,i} = v_{e,i} - \overline{v}_{15,i}$ is the residual of the least-squares rigid fit of target part $k$ to source part $k$. The geometric feedback correction of Section 4.1, applied to $\overline{V}_t$, adds $\tau_t r_{k,i}$ to every vertex. The output $\widehat{v}_{t,i} = \overline{v}_{t,i} + \tau_t r_{k,i}$ therefore matches both keyframe meshes exactly. Exactness holds for any vertex partition. The partition only decides how the residual is shared among parts, and the rigidity statement below holds within each part. Appendix K gives the full derivation of each step, together with the part-state extraction on the decoded frames.

**Lemma 1** (Within-part distance change)**.** *For every frame $t$ and all vertices $i, j \in \mathcal{P}_k$,*

$$\left| \|\widehat{v}_{t,i} - \widehat{v}_{t,j}\|_2 - \|v_{s,i} - v_{s,j}\|_2 \right| \leq \tau_t \, \|r_{k,i} - r_{k,j}\|_2. \tag{12}$$

*If the target part is a rigid transform of the source part, then $r_{k,i} = 0$ for all $i \in \mathcal{P}_k$ and the part stays rigid at every frame, with all within-part distances equal to those of the source mesh.*

In the difference of two vertices of one part, the shared translation cancels and the shared rotation preserves length. Only the residual term changes the distance, and the triangle inequality gives the bound. Appendix K gives the proof. For an edge of source length $\ell_{ij} = \|v_{s,i} - v_{s,j}\|_2$, the relative length change of this edge at every frame $t$ therefore obeys the following upper bound

$$\frac{\left| \|\widehat{v}_{t,i} - \widehat{v}_{t,j}\|_2 - \ell_{ij} \right|}{\ell_{ij}} \leq \tau_t \, \frac{\|r_{k,i} - r_{k,j}\|_2}{\ell_{ij}}. \tag{13}$$

The Edge column of Table 2 measures this change. A vertex-wise correction inherits every deformation of the decoded part. Part-state correction changes a part only through the residual differences inside it, which is the gap in rigidity seen in Table 2. The correction solves one $3 \times 3$ singular value decomposition per part and frame, so its cost is negligible next to the decoding pass.

## 5 Experiments

This section presents Bumblebee's performance on video-guided 4D morphing. The results show that Bumblebee outperforms the compared video-driven mesh generation methods in morphing accuracy and reduces stretching and twisting of mechanical parts. We also separately analyze the roles of the two core components, geometric feedback correction and part rigidity constraints.

### 5.1 Experimental setup

This section briefly describes the experimental setup. Evaluation details are in Appendix B, and the implementation, runtime and cost details of the full pipeline are reported in Appendix J.

**Generative model.** We use ActionMesh (Sabathier et al., 2026) as the generative model. All model parameters remain frozen, and no additional training or fine-tuning is required at any stage.

**Evaluation data.** We evaluate on two public benchmarks, Motion80 (Chen et al., 2026) and ActionBench (Sabathier et al., 2026), with the released evaluator of each. Appearance is scored with the released Consistent4D evaluator (Jiang et al., 2024). We also curate a mechanical transformation split from Objaverse (Deitke et al., 2023). It contains 20 clips from 9 assets with complete animations, valid endpoint meshes, consistent vertex correspondence, and usable geometry. We use this split and robot models from MuJoCo Menagerie to assess endpoint matching, intermediate trajectory accuracy, and part shape preservation on articulated assets whose ground-truth rigs are available.

**Baselines.** We compare with shape interpolation, ActionMesh, V2M4, and Motion 3-to-4. To analyze the two core components, we further compare the same decoded sequences without correction, with geometric feedback correction, and with morphing correction under part rigidity constraints.

**Evaluation metrics.** We use the official metrics of each benchmark to assess geometric accuracy, temporal motion, and visual quality. We also measure endpoint vertex errors and part rigidity residuals to evaluate keyframe matching and part shape preservation.

### 5.2 Main results

We evaluate Bumblebee on Motion80. Each sequence provides a source mesh, a target mesh and a reference video. Table 1 reports geometry and appearance.

Table 1: **Quantitative results on Motion80 and ActionBench.** V: video, S: source, T: target. Motion 3-to-4 and V2M4 do not take T (×). ActionMesh also supports T via masked observation; we additionally report ActionMesh variants with fewer inputs.

| Method | Inputs | | | Motion80 (Chamfer $\times 10^{-2}$) | | | | | | ActionBench ($\times 10^{-2}$) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | V | S | T | Chamfer↓ | $t$=15 ↓ | F-score↑ | CLIP↑ | LPIPS↓ | FVD↓ | CD-3D↓ | CD-4D↓ | CD-M↓ |
| *Shape interpolation* | | | | | | | | | | | | |
| Linear | × | ✓ | ✓ | 11.29 | **1.06** | 0.4878 | 0.965 | 0.112 | 944 | 4.55 | 7.19 | 14.63 |
| Minimum-jerk | × | ✓ | ✓ | 11.33 | **1.06** | 0.4880 | 0.966 | 0.113 | 937 | 4.52 | 7.23 | 14.62 |
| ARAP | × | ✓ | ✓ | 11.23 | **1.06** | 0.4883 | 0.964 | 0.114 | 957 | 4.52 | 7.28 | 14.78 |
| *Video-driven generation, released interfaces* | | | | | | | | | | | | |
| V2M4, video + source | ✓ | ✓ | × | 128.21 | 134.7 | 0.1340 | 0.826 | 0.177 | 1312 | 7.32 | 47.85 | 95.68 |
| Motion 3-to-4, video + source | ✓ | ✓ | × | 8.53 | 9.90 | 0.544 | 0.967 | 0.094 | 423 | 4.60 | 9.21 | 18.26 |
| ActionMesh, video | ✓ | × | × | 25.48 | 28.1 | 0.2533 | 0.860 | 0.154 | 968 | 5.47 | 8.54 | 15.26 |
| ActionMesh, video + source | ✓ | ✓ | × | 8.61 | 10.6 | 0.5400 | 0.972 | 0.087 | 327 | 4.12 | 8.38 | 17.18 |
| ActionMesh, video + source + target | ✓ | ✓ | ✓ | <u>6.26</u> | <u>3.64</u> | <u>0.6157</u> | <u>0.975</u> | <u>0.076</u> | <u>279</u> | <u>3.50</u> | <u>6.02</u> | <u>12.32</u> |
| **Bumblebee** | ✓ | ✓ | ✓ | **5.81** | **1.06** | **0.6260** | **0.976** | **0.073** | **266** | **3.49** | **5.27** | **10.74** |

V, S, T: the method receives the video, the source mesh and the target mesh. Motion80: means over 80 identities. $t$=15: Chamfer of the last frame against the given target mesh from the same evaluator, whose sampling floor for an exactly reproduced frame is 1.06. Chamfer and F-score from the released Motion 3-to-4 evaluator, appearance from the released Consistent4D evaluator (Jiang et al., 2024) on renders of the released Motion 3-to-4 renderer (Appendix G). On Motion80. ActionBench: means over 128 sequences with the released evaluator and point clouds. Best in bold, second best underlined.

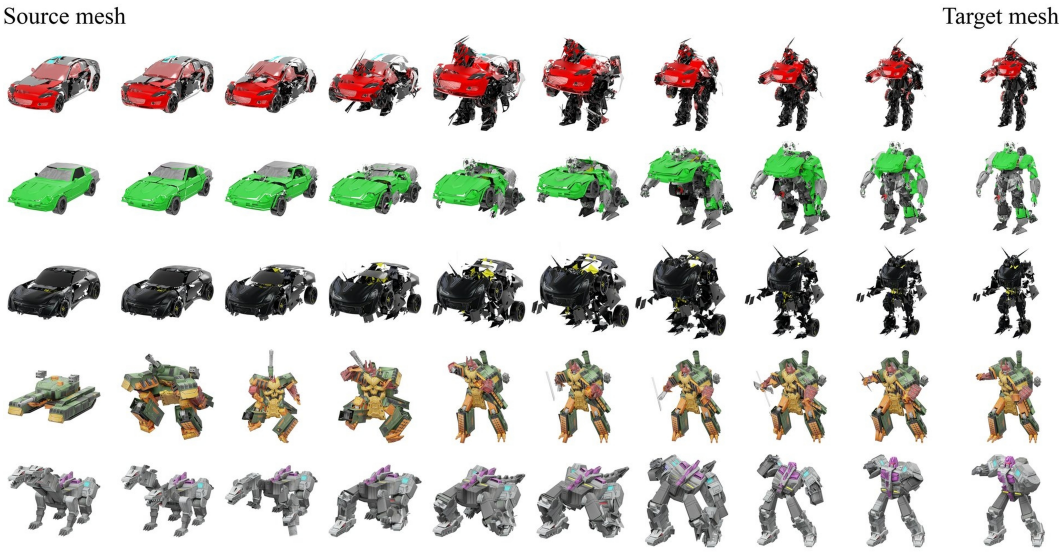



Figure 4: Qualitative results of Bumblebee on textured mechanical assets. Each row shows a morphing sequence from the source mesh (left) to the target mesh (right), keeping the original textures.

**Baselines.** We compare with interpolation without video (linear, minimum-jerk, ARAP (Alexa et al., 2000)) and with video-driven generation (ActionMesh (Sabathier et al., 2026), Motion 3-to-4 (Chen et al., 2026), V2M4 (Chen et al., 2025a)). Each baseline receives every input its interface accepts. No baseline uses our correction, and all methods share the same released evaluators.

**Comparison with released interfaces.** Bumblebee achieves the best values in Table 1. It lowers Chamfer by 77% against ActionMesh with the video only, by 33% against ActionMesh with the source mesh and by 32% against Motion 3-to-4. ActionMesh with both keyframes reaches 6.26 and misses the target. Bumblebee reaches 5.81 and matches the target exactly. On the unobserved frames, geometric feedback lowers Chamfer from 0.0682 to 0.0649 on Motion80 and from 0.0819 to 0.0600 on ActionBench-GT. Bumblebee runs in 1.2 minutes per sequence on a single GPU.

Table 2: **Part rigidity on mechanical assets.**

| Method | Transformation | | | | Menagerie | | | |
|---|---|---|---|---|---|---|---|---|
| | Rigid.↓ | Edge↓ | Chamfer↓ | F-score↑ | Rigid.↓ | Edge↓ | Chamfer↓ | F-score↑ |
| Linear interpolation | <u>0.106</u> | <u>64%</u> | <u>0.430</u> | <u>0.312</u> | <u>0.0428</u> | <u>25.4%</u> | <u>0.0595</u> | **0.757** |
| ActionMesh, video + source | 0.267 | 218% | 0.595 | 0.213 | 0.0707 | 76.1% | 0.1682 | 0.612 |
| ActionMesh, video + source + target | 0.341 | 433% | 0.509 | 0.256 | 0.1171 | 177.9% | 0.0930 | 0.694 |
| **Bumblebee** | **0.040** | **0.8%** | **0.373** | **0.319** | **0.0000** | **0.0%** | **0.0542** | <u>0.745</u> |

Rigid.: median per-part rigid-fit residual as a fraction of the bounding-box diagonal. Edge: 95th percentile of the relative edge-length change. The ground-truth rigs have a residual of 0.047 and 0.0000 with no edge change. No baseline uses our correction. Interpolation receives both keyframes, and ActionMesh receives the inputs its interface accepts (Appendices E and F). Best in bold, second best underlined.

**ActionBench.** Bumblebee achieves the best CD-3D, CD-4D and CD-M. ActionMesh with both keyframes is second. Bumblebee lowers its CD-4D by 12% and its CD-M by 13% on the full set.

**Appearance.** All methods use the released textures and the same renderer. Bumblebee achieves the best CLIP, LPIPS and FVD. Figure 4 and Appendix D show qualitative results on textured assets.

### 5.3 Ablations and controls

**Part rigidity.** On the transformation split, Bumblebee achieves a rigidity residual of 0.040, compared with 0.047 for the ground-truth rig, with only 0.8% edge-length change. On Menagerie, these values are 0.0000 and 0.0%, respectively. It also achieves the lowest Chamfer on both splits and the highest F-score on the transformation split among all compared methods (Table 2).

**Endpoint conditioning and correction.** Target conditioning reduces Chamfer from 0.0861 to 0.0626. Geometric feedback further reduces it to 0.0581 and enforces exact endpoint matching at both keyframes of every Motion80 sequence (Table 9, Appendix H).

**Local surface consistency.** With the same two-keyframe decode, part-state correction reduces excess local normal inconsistency from 9.794% to 0.083% on the transformation split and from 1.549% to 0.008% on Menagerie, compared with linear vertex correction (Appendix I.2).

**Video and keyframe controls.** The video decides the inner motion. With the endpoints and the correction fixed, a static guide raises Chamfer by +120% and a guide with shuffled inner frames raises it by +119%, both relative to the real video (Table 4). Further keyframes enter the prior as observed frames in the same way. On a Motion80 subset, four observed states lower the Chamfer of the unobserved frames from 0.0651 to 0.0534, and six states lower it further to 0.0497.

## 6 Conclusion

We have presented Bumblebee, a training-free method for video-guided 4D morphing between two exact keyframe meshes. Our key idea is to use the difference between the decoded trajectory and the given keyframes as geometric feedback. The keyframes are observed inside a frozen video-to-4D prior, and the feedback makes the exported first and last frames match the inputs exactly.

Together, these results show that endpoint accuracy and part preservation are distinct but complementary requirements. Geometric feedback determines the required endpoint corrections, while part-state correction determines how these corrections are distributed within mechanical parts. Combining both with a frozen video-to-4D prior yields exact keyframe agreement without retraining or per-sequence neural optimization. Bumblebee achieves the best geometric accuracy on Motion80 and ActionBench and the best appearance on Motion80, and it keeps mechanical parts as rigid as the ground-truth rigs. The method targets topology-aligned keyframe meshes with fixed vertex correspondence. Handling topology changes and ambiguous correspondences remains a natural direction for future work.

## Reproducibility Statement

To ensure the full reproducibility of our work, we have made the complete source code and execution scripts publicly available at https://anonymous.4open.science/r/4D-570D/. Additional method details and evaluation protocols are provided in the appendix.

## AI Use Statement

We used generative AI tools (large language models) to polish and edit the manuscript text.

## References


Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2000, New Orleans, LA, USA, July 23-28, 2000*, pp. 157–164, 2000.

Sherwin Bahmani, Ivan Skorokhodov, Victor Rong, et al. 4d-fy: Text-to-4d generation using hybrid score distillation sampling. In *Proc. of CVPR*, pp. 7996–8006, 2024.

Camille Buonomo, Julie Digne, and Raphaëlle Chaine. Volume preserving neural shape morphing. *Comput. Graph. Forum*, 2025.

Hongyuan Chen, Xingyu Chen, Zexiang Xu, et al. Motion 3-to-4: 3d motion reconstruction for 4d synthesis. In *Proc. of CVPR*, pp. 28947–28958, 2026.

Jianqi Chen, Biao Zhang, Xiangjun Tang, et al. V2M4: 4d mesh animation reconstruction from a single monocular video. In *Proc. of ICCV*, pp. 11643–11653, 2025a.

Zhaoxi Chen, Tianqi Liu, Long Zhuo, et al. 4dnex: Feed-forward 4d generative modeling made easy, 2025b. arXiv preprint arXiv:2508.13154.

Hyungjin Chung, Jeongsol Kim, Michael Thompson McCann, et al. Diffusion posterior sampling for general noisy inverse problems. In *Proc. of ICLR*, 2023.

Matt Deitke, Dustin Schwenk, Jordi Salvador, et al. Objaverse: A universe of annotated 3d objects. In *Proc. of CVPR*, pp. 13142–13153, 2023.

Shaocong Dong, Lihe Ding, Xiao Chen, et al. From one to more: Contextual part latents for 3d generation. In *Proc. of ICCV*, pp. 8230–8240, 2025.

Marvin Eisenberger, David Novotný, Gael Kerchenbaum, et al. Neuromorph: Unsupervised shape interpolation and correspondence in one go. In *Proc. of CVPR*, pp. 7473–7483, 2021.

Danielle Ezuz, Behrend Heeren, Omri Azencot, et al. Elastic correspondence between triangle meshes. *Comput. Graph. Forum*, pp. 121–134, 2019.

Tamar Flash and Neville Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *The journal of Neuroscience*, pp. 1688–1703, 1985.

Daoyi Gao, Yawar Siddiqui, Lei Li, et al. Meshart: Generating articulated meshes with structure-guided transformers. In *Proc. of CVPR*, pp. 618–627, 2025.

Félix G Harvey, Mike Yurick, Derek Nowrouzezahrai, et al. Robust motion in-betweening. *ACM Transactions on Graphics (TOG)*, pp. 60–1, 2020.

Yi-Jheng Huang, Shu-Yuan Chan, Wen-Chieh Lin, et al. Making and animating transformable 3d models. *Computers & Graphics*, pp. 127–134, 2016.

Yanqin Jiang, Li Zhang, Jin Gao, et al. Consistent4d: Consistent 360 dynamic object generation from monocular video. In *Proc. of ICLR*, pp. 51844–51861, 2024.

Zeren Jiang, Chuanxia Zheng, Iro Laina, et al. Mesh4d: 4d mesh reconstruction and tracking from monocular video, 2026. arXiv preprint arXiv:2601.05251.

Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Foundations of Crystallography*, pp. 922–923, 1976.

Bahjat Kawar, Michael Elad, Stefano Ermon, et al. Denoising diffusion restoration models. In *Proc. of NeurIPS*, 2022.

Minkyung Kwon, Jinhyeok Choi, Youngjin Shin, et al. Morphos: Autoregressive 4d generation with temporal structured latents, 2026. arXiv preprint arXiv:2606.02491.

Binh Huy Le and Zhigang Deng. Smooth skinning decomposition with rigid bones. *ACM Transactions on Graphics (TOG)*, pp. 1–10, 2012.

Zhibing Li, Mengchen Zhang, Tong Wu, et al. Ss4d: Native 4d generative model via structured spacetime latents. *ACM Transactions on Graphics (TOG)*, pp. 1–12, 2025.

Zhiqi Li, Yiming Chen, and Peidong Liu. Dreammesh4d: Video-to-4d generation with sparse-controlled gaussian-mesh hybrid representation. In *Proc. of NeurIPS*, 2024.

Hanwen Liang, Yuyang Yin, Dejia Xu, et al. Diffusion4d: Fast spatial-temporal consistent 4d generation via video diffusion models. In *Proc. of NeurIPS*, 2024.

Huan Ling, Seung Wook Kim, Antonio Torralba, et al. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. In *Proc. of CVPR*, pp. 8576–8588, 2024.

Rong Liu and Hao Zhang. Segmentation of 3d meshes through spectral clustering. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, pp. 298–305, 2004.

Tianqi Liu, Zihao Huang, Zhaoxi Chen, et al. Free4d: Tuning-free 4d scene generation with spatial-temporal consistency. In *Proc. of ICCV*, pp. 25571–25582, 2025.

Andreas Lugmayr, Martin Danelljan, Andres Romero, et al. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proc. of CVPR*, pp. 11451–11461, 2022.

Kevin M Lynch and Frank C Park. Modern robotics. *Mechanics, Planning, and Control*, 2017.

Sauradip Nag, Daniel Cohen-Or, Hao Zhang, et al. In-2-4d: Inbetweening from two single-view images to 4d generation. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–12, 2025.

Hiroki Nishizawa, Hubert PH Shum, Yoshihiro Fukuhara, et al. Feed-forward motion in-betweening for any 4d, 2026. arXiv preprint arXiv:2606.22131.

Annika Oehri, Philipp Herholz, and Olga Sorkine-Hornung. Higher order continuity for smooth as-rigid-as-possible shape modeling. *The Journal of Computer Graphics Techniques*, pp. 198–215, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, et al. Dinov2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024.

Ben Poole, Ajay Jain, Jonathan T. Barron, et al. Dreamfusion: Text-to-3d using 2d diffusion. In *Proc. of ICLR*, 2023.

Jiawei Ren, Cheng Xie, Ashkan Mirzaei, et al. L4GM: large 4d gaussian reconstruction model. In *Proc. of NeurIPS*, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, et al. High-resolution image synthesis with latent diffusion models. In *Proc. of CVPR*, pp. 10674–10685, 2022.

Litu Rout, Negin Raoof, Giannis Daras, et al. Solving linear inverse problems provably via posterior sampling with latent diffusion models. In *Proc. of NeurIPS*, 2023.

Remy Sabathier, David Novotny, Niloy J Mitra, et al. Actionmesh: Animated 3d mesh generation with temporal 3d diffusion, 2026. arXiv preprint arXiv:2601.16148.

Dvir Samuel, Yuval Atzmon, Gal Chechik, et al. Fast 4d mesh generation by spatio-temporal attention chains, 2026. arXiv preprint arXiv:2605.19786.

Lu Sang, Zehranaz Canfes, Dongliang Cao, et al. Twosquared: 4d generation from 2d image pairs. In *Proc. of 3DV*, pp. 659–669, 2026.

Steven M Seitz and Charles R Dyer. View morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 21–30, 1996.

Eli Shechtman, Alex Rav-Acha, Michal Irani, et al. Regenerative morphing. In *Proc. of CVPR*, pp. 615–622, 2010.

Yahao Shi, Yang Liu, Yanmin Wu, et al. Drive any mesh: 4d latent diffusion for mesh deformation from video, 2025. arXiv preprint arXiv:2506.07489.

Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pp. 245–254, 1985.

Uriel Singer, Shelly Sheynin, Adam Polyak, et al. Text-to-4d dynamic scene generation. In *Proc. of ICML*, pp. 31915–31929, 2023.

Bowen Song, Soo Min Kwon, Zecheng Zhang, et al. Solving inverse problems with latent diffusion models via hard data consistency. In *Proc. of ICLR*, 2024.

Olga Sorkine, Marc Alexa, et al. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, pp. 109–116, 2007.

Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, pp. 80, 2007.

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

Andrej Nikolaevich Tikhonov and Vasiliy Yakovlevich Arsenin. Solutions of ill-posed problems. *mathematics of computation*, 32(144):1320, 1978.

Chih-Jung Tsai, Cheng Sun, and Hwann-Tzong Chen. Multiview regenerative morphing with dual flows. In *Proc. of ECCV*, pp. 492–509, 2022.

Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on pattern analysis and machine intelligence*, pp. 376–380, 1991.

Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. In *Proc. of ICLR*, 2023.

Zijie Wu, Chaohui Yu, Fan Wang, et al. Animateanymesh: A feed-forward 4d foundation model for text-driven universal mesh animation. In *Proc. of ICCV*, pp. 13557–13568, 2025.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, et al. Structured 3d latents for scalable and versatile 3d generation. In *Proc. of CVPR*, pp. 21469–21480, 2025.

Yiming Xie, Chun-Han Yao, Vikram Voleti, et al. SV4D: dynamic 3d content generation with multi-frame and multi-view consistency. In *Proc. of ICLR*, 2025.

Songlin Yang, Yushi Lan, Honghua Chen, et al. Textured 3d regenerative morphing with 3d diffusion prior. In *Proc. of ICCV*, pp. 15159–15170, 2025.

Chun-Han Yao, Yiming Xie, Vikram Voleti, et al. SV4D 2.0: Enhancing spatio-temporal consistency in multi-view video diffusion for high-quality 4d generation. In *Proc. of ICCV*, pp. 13248–13258, 2025.

Jiraphon Yenphraphai, Jianqi Chen, Jian Wang, et al. Helix4d: Complex 4d mesh generation, 2026. arXiv preprint arXiv:2605.26109.

Minghao Yin, Yukang Cao, and Kai Han. Wukong's 72 transformations: High-fidelity textured 3d morphing via flow models. In *Proc. of NeurIPS*, 2025.

Bohan Zeng, Ling Yang, Siyu Li, et al. Trans4d: Realistic geometry-aware transition for compositional text-to-4d synthesis, 2024. arXiv preprint arXiv:2410.07155.

Kaiwen Zhang, Yifan Zhou, Xudong Xu, et al. Diffmorpher: Unleashing the capability of diffusion models for image morphing. In *Proc. of CVPR*, pp. 7912–7921, 2024.

Peng Zheng, Dehong Gao, Deng-Ping Fan, et al. Bilateral reference for high-resolution dichotomous image segmentation, 2024. arXiv preprint arXiv:2401.03407.

Linqi Zhou, Aaron Lou, Samar Khanna, et al. Denoising diffusion bridge models. In *Proc. of ICLR*, 2024.

# A Related Work

**Keyframe-conditioned in-betweening.** CompletionAny4D is a concurrent feed-forward method that conditions motion synthesis on sparse keyframe meshes and a text motion prompt, and fixes the supplied keyframes at their specified timesteps (Nishizawa et al., 2026). Its released interface uses sparse keyframe meshes and a text motion prompt, rather than a reference video. Bumblebee studies a different protocol with a reference video, two topology-aligned endpoint meshes, and post-decoding geometric correction without retraining. Applying CompletionAny4D to our input would require an additional video-to-text conversion and would change its released interface. We therefore compare the conditioning protocols and endpoint guarantees, while evaluating Bumblebee against video-conditioned baselines under the DE-4DM protocol.

**Video-to-4D generation and 4D mesh animation.** These methods aim to produce an animated 3D representation from a monocular video. They either distill 2D video priors into a 4D representation by score distillation (Poole et al., 2023; Singer et al., 2023; Bahmani et al., 2024; Ling et al., 2024; Zeng et al., 2024; Liang et al., 2024), generate multi-view video and reconstruct from it (Xie et al., 2025; Yao et al., 2025; Li et al., 2025; Liu et al., 2025; Chen et al., 2025b), fit a mesh sequence per video by render-based optimization (Chen et al., 2025a; Jiang et al., 2026), drive a given mesh with a trained deformation model (Shi et al., 2025; Kwon et al., 2026), or run a feed-forward temporal 3D diffusion model that decodes one latent per frame as deformations of a reference mesh (Sabathier et al., 2026; Samuel et al., 2026; Yenphraphai et al., 2026); Motion 3-to-4 trains a reconstructor for the same input and releases the Motion80 evaluation we build on (Chen et al., 2026). In every case the input is a video and at most one anchor mesh, and the last frame is a prediction. ActionMesh is the closest to our setting: its interface accepts an observed latent at any frame and holds it fixed during denoising, but the released inference observes one frame and nothing constrains the exported mesh to equal the observed one (Sabathier et al., 2026). Bumblebee keeps this prior frozen, observes both keyframes, and adds the shape-space correction that makes the exported keyframes exact and keeps the parts of the asset intact.

**Morphing and in-betweening between two given states.** These methods aim to produce the transition between a source state and a target state. Mesh morphing chooses the path by a geometric criterion, such as as-rigid-as-possible interpolation (Alexa et al., 2000), or by a learned correspondence and interpolation (Eisenberger et al., 2021); image and view morphing regenerate the in-between appearance from the two endpoints (Seitz & Dyer, 1996; Shechtman et al., 2010; Tsai et al., 2022; Zhang et al., 2024); Wukong steers a frozen 3D generative flow between two prompts and morphs shape and texture without training (Yin et al., 2025; Xiang et al., 2025). In 4D, motion in-betweening fills the frames between skeletal keyframes (Harvey et al., 2020), In-2-4D fits Gaussian deformation fields to generated video fragments between two images (Nag et al., 2025), TwoSquared reconstructs two endpoints and predicts their deformation (Sang et al., 2026), and the concurrent CompletionAny4D trains a flow model on 260k sequences to preserve given keyframe meshes from a text prompt (Nishizawa et al., 2026). The mismatch with our requirement is one of three: the path is chosen by a criterion rather than by a reference video, the endpoints are generated or reconstructed rather than given as the artist's meshes, or a model is trained for the task. The keyframe-replacement baseline of CompletionAny4D is our latent-level operation without the shape-space level. Bumblebee takes the given meshes and a video and trains nothing; the direct adaptations of these routes are our baselines (Table 3, Section 5).

**Diffusion bridges and test-time consistency.** Denoising diffusion bridge models learn to transport samples between paired endpoint distributions (Zhou et al., 2024). Data-consistency methods instead condition pretrained diffusion models on known measurements. RePaint replaces observed regions during sampling (Lugmayr et al., 2022), while DPS uses measurement-likelihood gradients (Chung et al., 2023). DDNM and ReSample also enforce measurement consistency during inference (Wang et al., 2023; Song et al., 2024). In 4D reconstruction, V2M4 fits a mesh animation to a video (Chen et al., 2025a), and Consistent4D optimizes a dynamic radiance field with rendering-based consistency losses (Jiang et al., 2024). Bumblebee uses a frozen model to generate motion with both keyframe latents observed. Our geometric feedback then enforces exact endpoint agreement in the decoded mesh, where latent conditioning alone leaves reconstruction errors. Part-state correction further

controls how these adjustments act on mechanical structures. This combines exact mesh-space endpoint control with part shape preservation without training the generator.

**Shape deformation and distortion measures.** Our corrections and our integrity metric come from the geometry-processing literature: minimum-jerk trajectories (Flash & Hogan, 1985), Tikhonov-type smoothness regularization (Tikhonov & Arsenin, 1978), piecewise-rigid and as-rigid-as-possible deformation (Sumner et al., 2007; Sorkine et al., 2007), orthogonal Procrustes pose fitting (Umeyama, 1991), rotation interpolation (Shoemake, 1985), and the Dirichlet energy used to audit the distortion of shape interpolations (Ezuz et al., 2019; Sun et al., 2024). We use them to close a decoder residual on a generated trajectory rather than to define the transition itself, and we report the released evaluators of Motion 3-to-4 (Chen et al., 2026), ActionBench (Sabathier et al., 2026) and Consistent4D (Jiang et al., 2024) unchanged, with their released settings.

Table 3: **Task taxonomy.** Only the last row lets an animator drive their own asset between two of its states with a video: without a guide the motion is invented, without two keyframes the terminal state drifts, and without a fixed vertex correspondence the output is not the asset. ✓: provided; ○: partial (endpoints as images or prompts, video generated internally, identity carried by a deformation field; for ActionMesh the released inference observes one keyframe and the exported endpoints are not guaranteed); ×: not provided.

| Method (family) | Video guide | Two given 3D endpoints | Exact endpoints | Fixed topology | Vertex corresp. | Feed-forward | No per-sequence fitting |
|---|---|---|---|---|---|---|---|
| ActionMesh (Sabathier et al., 2026) (video-to-4D mesh) | ✓ | ○ | × | ✓ | ✓ | ✓ | ✓ |
| L4GM (Ren et al., 2024) (video-to-4D Gaussians) | ✓ | × | × | × | ○ | ✓ | ✓ |
| V2M4 (Chen et al., 2025a) (video-to-4D mesh, per-sequence) | ✓ | × | × | ✓ | ✓ | × | × |
| In-2-4D (Nag et al., 2025) (two-image 4D inbetweening) | ○ | ○ | × | × | ○ | × | × |
| TwoSquared (Sang et al., 2026) (two-image 4D generation) | × | ○ | × | ○ | ○ | ○ | ○ |
| CompletionAny4D (Nishizawa et al., 2026) (keyframe-conditioned 4D in-betweening, trained) | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Wukong (Yin et al., 2025) (static 3D morphing) | × | ○ | × | × | ○ | ✓ | ✓ |
| Bumblebee (video-guided 4D morphing between exact keyframe meshes) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

## B Evaluation details

**Keyframes.** In every table the two keyframes are the ground-truth meshes at frames 0 and 15 of the evaluated sequence. The Motion80 columns of Table 1 and Table 9 use the 80 released Motion80 sequences with the released camera-0 video, and the keyframes are frames 0 and 15 of the released mesh sequence. The ActionBench columns of Table 1, Table 7 and Table 8 use the 128 ActionBench sequences with the released video, and the keyframes are frames 0 and 15 of the animation recovered from the Objaverse source asset and aligned to the released point clouds (Appendix H). Table 2 uses the transformation clips and the Menagerie sequences, and the keyframes are the first and last frames of each clip.

**Inputs of each row.** Shape interpolation receives both keyframes and no video. ActionMesh with the video only receives the video. ActionMesh with the video and the source mesh and Motion 3-to-4 receive the video and keyframe 0. ActionMesh with the video, the source mesh and the target mesh

receives both keyframes as masked observations. Bumblebee receives the video and both keyframes. V2M4 receives the video and the source mesh. No baseline uses our correction.

**Schedules and checkouts.** The official ActionMesh checkout is fixed at commit `d5c01f50 45df55819e337369c9617f603c667e00` with the official `actionmesh_lowram` configuration in bfloat16, unchanged for every run reported in this paper. ActionMesh with the video only and with the video and the source mesh run the released inference scripts with their default 30-step Stage I schedule. The two-keyframe decode used by ActionMesh with both keyframes and by Bumblebee uses the released 15-step schedule. Weights are the released ActionMesh denoiser and temporal autoencoder, TripoSG VAE, DINOv2 (Oquab et al., 2024) and RMBG (Zheng et al., 2024).

**Evaluators.** Motion80 uses the released Motion 3-to-4 repository at commit `6dc121d8...`, evaluator `evaluation_pcd.py` unmodified, 50,000 surface points and `trimesh` 5.0.0. For each identity the whole released camera-0 sequence is denoised at once with the ground-truth meshes at its first and last frame observed as the two keyframes. The official ActionBench evaluator uses CUDA, 10k-point ICP and 100k-point Chamfer over sixteen frames, and the released point clouds are used unchanged. Consistent4D uses the released evaluator at commit `e6d3e258...`, seven objects, four official views and 32 frames rendered at $256 \times 256$ on a white background.

## C Invariants verified before evaluation

An independent script reloaded every exported sequence and checked: sixteen finite meshes per sequence; identical vertex and face counts across the sixteen frames; maximum absolute error of the two observed latents after the last denoising step equal to 0; maximum vertex error to the source mesh at frame 0 and to the target mesh at frame 15 equal to 0 after export and reload; and a manifest flag confirming that no ground truth was read during generation or correction. All 128 sequences passed and no sequence was removed.

The same per-vertex check is applied to every keyframe-setting row from the exported meshes, in the normalized units of the task metrics. Over all 80 identities, every closed row of Table 9 (Bumblebee, the ramp and regularized corrections of the two-endpoint decode, and the endpoint-only interpolants) has a maximum per-vertex error of $4.3 \cdot 10^{-15}$ to the source at frame 0 and $3.3 \cdot 10^{-15}$ to the target at frame 15, which is floating-point round-off. The released evaluator samples 50,000 points, so even an exactly reproduced frame scores a sampling floor rather than zero: 0.0107 on characters and 0.0164 on the transformation split. The unclosed rows miss the target per vertex by 0.043 on average (maximum 0.83) for the two-endpoint decode, 0.13 (maximum 1.08) for the source-anchored prior and 0.11 (maximum 1.04) for Motion 3-to-4, which also moves the given source by up to 0.048 at frame 0.

**Part rigidity.** Because a Dirichlet energy of 2 is necessary but not sufficient for rigidity, the transformation split is checked directly. For every clip, frame and connected component we fit the rigid transform from the source part to the output part by orthogonal Procrustes and record the largest vertex residual as a fraction of the bounding-box diagonal, and the 95th percentile of the relative edge-length change. The ground-truth trajectories themselves have a median per-clip residual of 0.047 (maximum 0.200), because the rigs deform their parts slightly; the piecewise-rigid output of Bumblebee has 0.040 (maximum 0.192), a median difference of $-0.0005$ to the ground truth of the same clip, whereas the vertex-wise corrections of the two-keyframe and source-anchored trajectories reach 0.330 and 0.254. The edge-length statistic is 0 for the ground truth, 0.8% for Bumblebee and 510% and 226% for the two vertex-wise corrections. The output of Bumblebee is therefore rigid per part to the same degree as the ground truth.

## D Qualitative comparisons

## E Transformation split: rigid-part assets

Rigged transformation assets (a Dinobot that unfolds from robot to dinosaur, jets, a tank, cars, a shuttle and a truck that unfold into robots, in both directions) are the extreme case of the task:

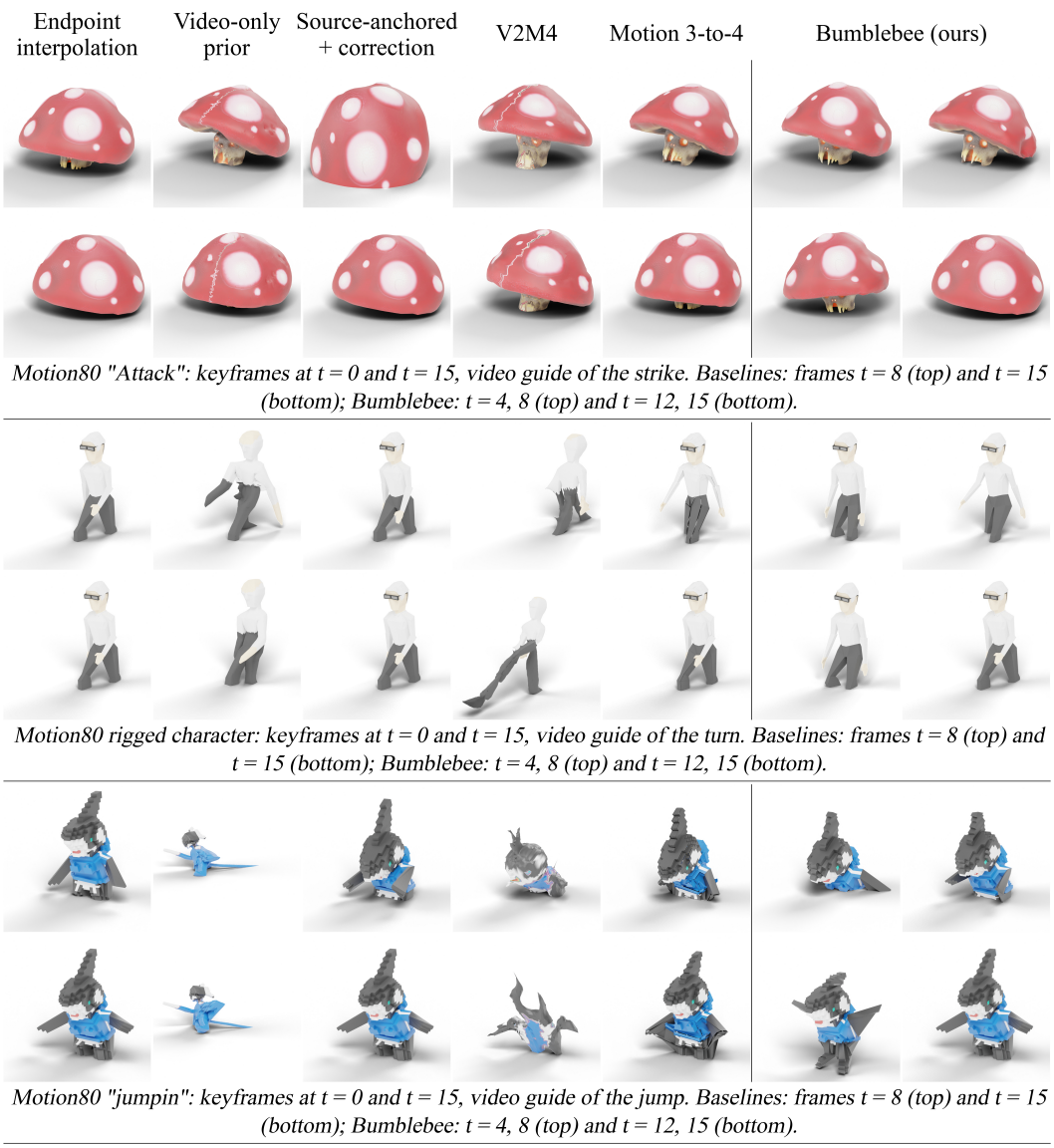



Figure 5: **Qualitative comparison on Motion80.** Columns: endpoint-only interpolation, ActionMesh with the video only, ActionMesh with the source mesh and the linear correction (Table 9), V2M4, Motion 3-to-4 and Bumblebee (right, two columns). For each identity the baselines show frames $t = 8$ (top) and $t = 15$ (bottom); Bumblebee shows $t = 4, 8$ (top) and $t = 12, 15$ (bottom). All meshes are rendered with the released textures under identical cameras; the caption under each block names the clip, the guide and the frames shown in each row.

the endpoints are two states of one asset and every part moves rigidly. We curate this split from publicly available rigged assets on Objaverse (Deitke et al., 2023). We retain clips with complete transformations, valid endpoint meshes, consistent vertex correspondence, and usable geometry. The resulting Objaverse transformation split contains 20 clips from 9 assets. The whole transformation is the guide, the first and last frames are the endpoints, and sixteen frames are sampled evenly. Table 5 summarizes the evaluation splits. Table 2 reports the results on the transformation split, and Figure 8 shows qualitative results.

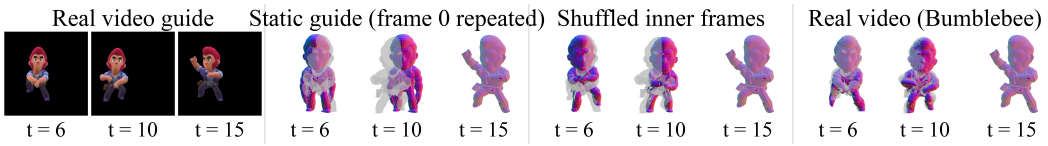



Figure 6: **Intervention on the video guide** (identity of Figure 7, grey ghost: ground truth). A static guide leaves the pose near the source while the ground truth strikes, shuffled inner frames give wrong poses, the real guide is followed; the keyframes are exact in all three.

Table 4: **Video intervention on Motion80 (keyframe setting) with ground-truth endpoints.** Endpoints, steps and the linear correction are those of the main run, only the guide changes: the real video, its inner frames shuffled (frames 0 and 15 kept), or a static video (frame 0 repeated). The last column is the relative change against the real video. Best per column in bold, second best underlined; the real video is best in every column.

| guide | Chamfer↓ | F-score↑ | vs real (Chamfer) |
|---|---|---|---|
| endpoint-only interpolation (no video) | <u>0.1129</u> | <u>0.4878</u> | +94% |
| Bumblebee, static video | 0.1280 | 0.4553 | +120% |
| Bumblebee, shuffled inner frames | 0.1276 | 0.4485 | +119% |
| Bumblebee, real video | **0.0581** | **0.6260** | – |

Table 5: **Evaluation splits for part-shape preservation.** The Objaverse split contains mechanical transformation clips. MuJoCo Menagerie provides robot models with ground-truth rigid-part motion.

| Split | Source | Assets | Clips/models | Purpose |
|---|---|---|---|---|
| Objaverse transformation | Objaverse (Deitke et al., 2023) | 9 | 20 clips | Textured mechanical transformations |
| MuJoCo Menagerie | MuJoCo Menagerie | 14 | 14 models | Ground-truth rigid-part evaluation |

## F Official robot models: MuJoCo Menagerie

The transformation split is assembled by us. To test the rigid-part conclusions on an official, openly licensed source of articulated assets, we take robot models from MuJoCo Menagerie, the curated collection of Google DeepMind (Apache-2.0 and BSD-3 per model), covering quadrupeds, bipeds and humanoids, arms, a gripper and a dexterous hand. Each model gives exact ground truth for free: a motion is a joint-space trajectory between two joint configurations, the start being the released home configuration and the end an offset of 35% of each limited joint range, at most 1.2 rad, alternating in sign across joints, and forward kinematics places every visual mesh of the robot at every frame, so the ground-truth trajectory is exactly rigid per part. We bake 73 frames at 24 fps into an animated GLB with the meshes and colours of the release, and run the protocol of Appendix E unchanged. Table 2 reports the results.

### F.1 Keyframe-Based Part Partitioning and Rigidity Correction

The distance bound in Lemma 1 applies within each estimated vertex group. We evaluate the final output on the true mechanical parts to test whether a partition preserves the asset's original structure. Table 6 compares four vertex partitions on the 14 Menagerie assets. All settings use the same two-keyframe decode and the same correction procedure from Section 4.2. Only the vertex partition changes.

**Partition construction.** The default uses connected components of the source mesh. The geometry-based alternative applies spectral clustering to a $k$-nearest-neighbour graph of source vertex positions, following the spectral segmentation approach of Liu & Zhang (2004). The motion-based alternative follows the rigid-bone initialization of SSDR (Le & Deng, 2012). It alternates rigid fits with vertex

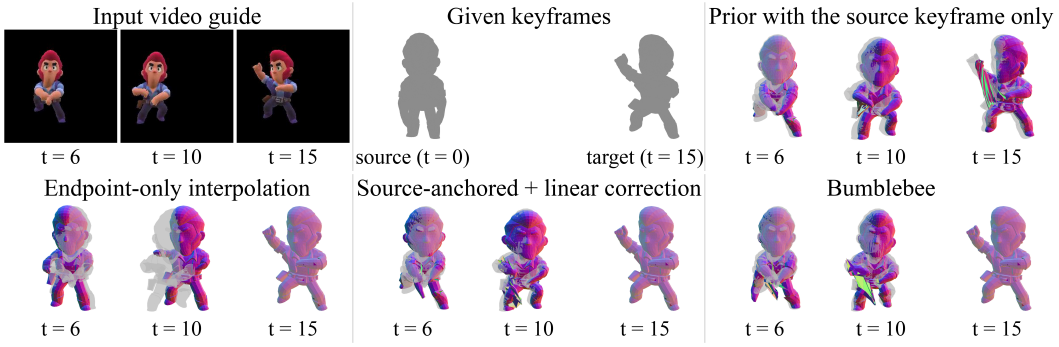



Figure 7: **What each family satisfies and what it loses.** One Motion80 identity at $t \in \{6, 10, 15\}$; grey ghost: ground truth. The prior given the source keyframe only follows the guide but misses the target; endpoint-only interpolation hits both keyframes but invents the motion; the source-anchored decode with the linear correction (Table 9) reaches the target but keeps the inner-frame error of the one-keyframe decode; Bumblebee (two observed keyframes, linear correction) follows the guide and hits both keyframes exactly.

reassignment on the uncorrected decoded trajectory. Both alternatives are given the number of connected components as their initial part count.

Our keyframe-based partition uses the corresponding vertices of the two given endpoint meshes. We alternate rigid fitting and vertex reassignment on this pair, then merge groups whose endpoint transforms agree within 0.5% of the source bounding-box diagonal on every vertex of the group being merged. A majority vote over one-ring mesh neighbours resolves ambiguous assignments near joint axes. Fitting uses a 6,000-vertex subsample, and assignments are propagated to the full mesh. Thus, mesh adjacency is used for boundary cleanup, while the grouping is estimated from endpoint motion. No partition uses ground-truth intermediate frames or ground-truth part labels.

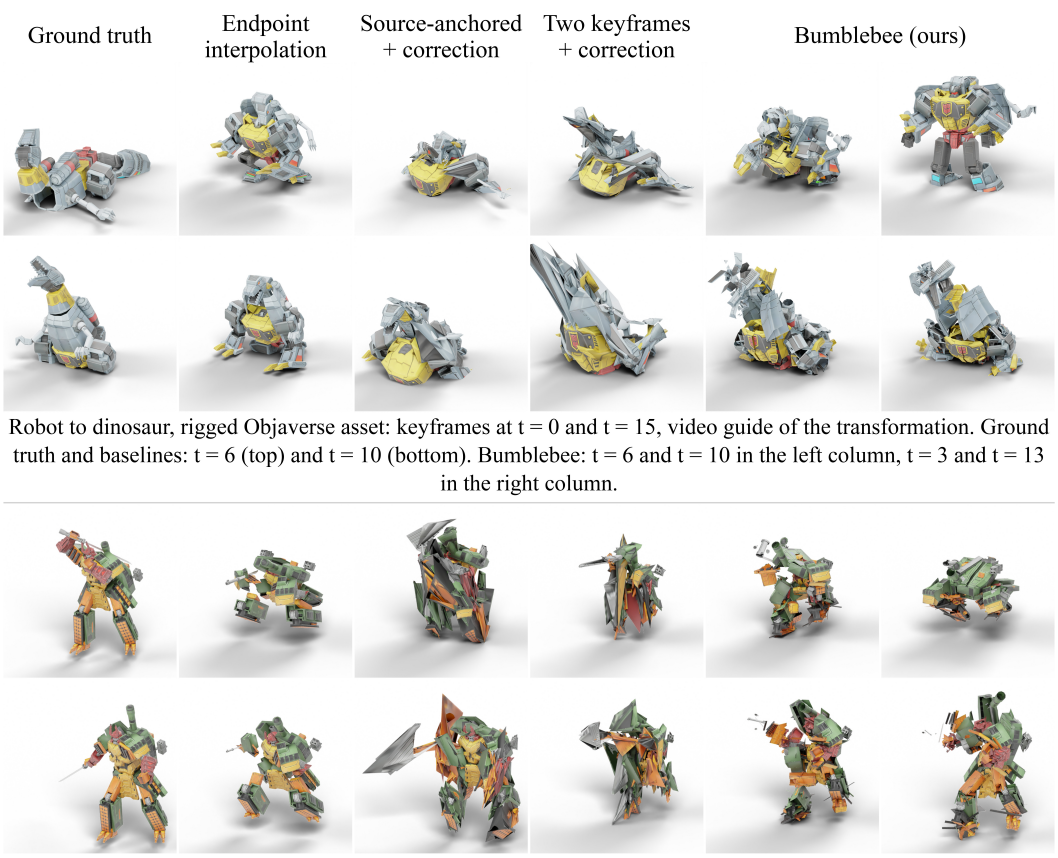
Table 6: **Effect of part partitioning on Menagerie.** All rows use the same decoded trajectories and correction. Parts is the mean number of groups. GT fit measures how well a partition fits ground-truth motion. Rigid. and Edge measure the corrected trajectory on the true asset parts. Lower is better except for F-score.

| Partition | Input | Parts | Chamfer↓ | F-score↑ | GT fit↓ | Rigid.↓ | Edge↓ |
|---|---|---|---|---|---|---|---|
| Connected components | Topology | 104 | 0.0542 | 0.745 | 0.0000 | 0.0000 (0.0012) | 0.0% |
| Spectral clustering | Source geometry | 104 | 0.0555 | 0.736 | 0.1385 | 0.1207 (0.1884) | 16.6% |
| Motion-based clustering | Decoded motion | 100 | 0.0556 | 0.737 | 0.0912 | 0.1243 (0.1911) | 30.0% |
| Keyframe-based rigid groups | Source and target | 11 | **0.0537** | **0.746** | 0.0000 | 0.0000 (0.1729) | 0.0% |

GT fit is the median across assets of the maximum rigid-fit residual of ground-truth motion under the estimated partition. Rigid. reports the median across assets, with the maximum in parentheses. Both residuals are normalized by the bounding-box diagonal. Edge is the median across assets of the 95th-percentile relative edge-length change. Values of 0.0000 reflect the displayed precision.

**Results.** The keyframe-based partition achieves the lowest Chamfer distance (0.0537) and highest F-score (0.746) among the four settings. It reduces the mean number of parts from 104 to 11 while retaining comparable geometric accuracy to the default. Its median rigidity residual is 0.0000, and its edge-change score is 0.0%. Geometry-based and motion-based clustering have similar geometric accuracy but edge changes of 16.6% and 30.0%, respectively. These results show that the given keyframes provide useful information for part partitioning and subsequent rigidity correction.

The keyframe-based partition has a maximum output rigidity residual of 0.1729 and a maximum GT-fit residual of 0.0609. Two endpoints determine relative motion between the given states. Parts




Robot to dinosaur, rigged Objaverse asset: keyframes at t = 0 and t = 15, video guide of the transformation. Ground truth and baselines: t = 6 (top) and t = 10 (bottom). Bumblebee: t = 6 and t = 10 in the left column, t = 3 and t = 13 in the right column.

Tank to robot, rigged Objaverse asset: keyframes at t = 0 and t = 15, video guide of the transformation. Ground truth and baselines: t = 6 (top) and t = 10 (bottom). Bumblebee: t = 6 and t = 10 in the left column, t = 3 and t = 13 in the right column.

Figure 8: **Qualitative comparison on the transformation split.** Columns: ground truth, endpoint-only interpolation, ActionMesh with the source mesh, ActionMesh with both keyframes, and Bumblebee (right, two columns). Ground truth and baselines show frames $t = 6$ (top) and $t = 10$ (bottom). Bumblebee shows the same two frames in its left column and $t = 3$ (top) and $t = 13$ (bottom) in its right column. The baselines reach the target by shearing the parts. Bumblebee moves each part rigidly. All meshes carry the released textures under identical cameras.

that share this endpoint motion but move differently in between can be merged. This explains why the median and maximum residuals must be reported separately.

## G Texture and UV inheritance

Every output frame carries the face set $\mathbf{F}$ of the source mesh, so the UV layout and the texture images of the asset apply to every frame without reprojection or baking. We checked this on all textured outputs used in this paper (the transformation split, the high-poly assets and three characters): the face array and the UV array of every frame beyond frame 0 equal those of the source frame exactly. For rigid-part outputs the texture is undistorted by construction; for deformable characters the texture follows the surface.

**Rendering protocol.** Table 1 uses the released Motion 3-to-4 renderer unchanged: its import path, EEVEE settings, environment map and camera 0 (azimuth 270°, radius 1.5), 512-pixel renders composited over black and downsampled to 256 pixels as the released frames are. The released GLB of each identity is imported with its materials and UV layout, its vertices are replaced frame by frame by the trajectory under evaluation (Motion 3-to-4 and Bumblebee keep the source face set; V2M4

uses the transfer below on its own topology), and the scene is normalized by the released rule (largest extent 1, centred) computed on the ground-truth first frame and applied identically to every method. The reference is the ground-truth trajectory rendered with the same pipeline; the released JPEG frames themselves are not used as reference because their normalization constants and colour pipeline are not published. The renders are scored with the released Consistent4D evaluation scripts (Jiang et al., 2024) unchanged (CLIP similarity and LPIPS per frame, FVD per clip with the released I3D weights); the protocol has one camera, so the scripts' four-view average reduces to that view.

**Texture-transfer procedure for generative baselines.** For Table 1 V2M4's first-frame mesh is aligned to the given source by bounding-box normalisation and ICP with scale (50k surface samples per mesh, 200 iterations); every V2M4 vertex then receives the UV coordinates and material of its nearest point on the source surface (200k surface samples) by barycentric interpolation, and the assignment is carried along V2M4's own trajectory, so that its temporal consistency is credited in full. V2M4's own texture stage produced flat grey textures in our runs, so its own textures are not used; the transfer is strictly more favourable to V2M4 than its own output. Bumblebee reaches CLIP similarity 0.976 against 0.826 for V2M4 and 0.967 for Motion 3-to-4, LPIPS 0.073 against 0.177 and 0.094, and FVD 266 against 1312 and 423. Motion 3-to-4 with the given source keeps the source face set, and the projections of Bumblebee keep every vertex on the asset's surface, so for both the released UV layout applies directly, without re-projection or baking.

## H Full ablation on Motion80

Table 7: **ActionBench comparison including the leaderboard methods.**

| Method | Inputs | | | Seq. | Time | Official metrics ($\times 10^{-2}$) | | |
|---|---|---|---|---|---|---|---|---|
| | V | S | T | | | CD-3D↓ | CD-4D↓ | CD-M↓ |
| *Shape interpolation* | | | | | | | | |
| Linear | × | ✓ | ✓ | 128 | <0.1 min | 4.55 | 7.19 | 14.63 |
| Minimum-jerk | × | ✓ | ✓ | 128 | <0.1 min | 4.52 | 7.23 | 14.62 |
| ARAP | × | ✓ | ✓ | 128 | <0.1 min | 4.52 | 7.28 | 14.78 |
| *Video-driven generation, released interfaces* | | | | | | | | |
| DreamMesh4D† (Li et al., 2024) | ✓ | × | × | 128 | – | 10.4 | 15.2 | 26.5 |
| LIM† | ✓ | × | × | 128 | – | 8.9 | 12.6 | 24.3 |
| ShapeGen4D† | ✓ | × | × | 128 | – | 5.6 | 17.0 | 34.8 |
| V2M4† (Chen et al., 2025a) | ✓ | × | × | 128 | 20 min | 6.8 | 34.0 | 61.6 |
| V2M4, video + source | ✓ | ✓ | × | 128 | 26 min | 7.32 | 47.85 | 95.68 |
| ActionMesh, video | ✓ | × | × | 128 | 2.4 min | 5.47 | 8.54 | 15.26 |
| ActionMesh, video + source | ✓ | ✓ | × | 128 | 2.4 min | 4.12 | 8.38 | 17.18 |
| ActionMesh, video + source + target | ✓ | ✓ | ✓ | 128 | 1.2 min | <u>3.50</u> | <u>6.02</u> | <u>12.32</u> |
| **Bumblebee** | ✓ | ✓ | ✓ | 128 | 1.2 min | **3.49** | **5.27** | **10.74** |

Means over the 128 ActionBench sequences with the released evaluator and point clouds. †: numbers from the official ActionBench leaderboard, where ActionMesh with the video only scores 5.4, 8.5 and 15.3. Best in bold, second best underlined.

**ActionBench keyframe setting.** ActionBench releases tracked point clouds. We recover the mesh sequences of its 128 animations from the released Objaverse source assets, take frames 0 to 15 of the released animation, and align them to the released point clouds. Frames 0 and 15 are the two keyframes. The official evaluator and point clouds are used unchanged for Table 7. Table 8 reports Chamfer and F-score on the recovered meshes with the Motion 3-to-4 evaluator.

**Full ablation.** Table 9 lists every closure of Section 5.3 on the standard split.

Table 8: **ActionBench, Chamfer and F-score on the recovered meshes.**

| Method | Chamfer↓ | F-score↑ |
|---|---|---|
| Linear interpolation | 0.0663 | 0.6809 |
| Minimum-jerk interpolation | 0.0670 | 0.6813 |
| ARAP interpolation | 0.0675 | 0.6766 |
| V2M4, video + source | 0.5995 | 0.1611 |
| ActionMesh, video + source | 0.1042 | 0.6437 |
| ActionMesh, video + source + target | <u>0.0772</u> | <u>0.7308</u> |
| **Bumblebee** | **0.0539** | **0.7469** |

Means over 128 sequences, Motion 3-to-4 evaluator.

Table 9: **Full ablation on Motion80.**

| Method | Geometry on Motion80 | | |
|---|---|---|---|
| | Chamfer↓ | F-score↑ | Inner Chamfer↓ |
| *Endpoint-only interpolation (no video)* | | | |
| linear | 0.1129 | 0.4878 | 0.1275 |
| minimum-jerk | 0.1133 | 0.4880 | 0.1280 |
| *Frozen prior, one observed keyframe* | | | |
| video only (one anchor) | 0.2548 | 0.2533 | 0.2607 |
| video + source | 0.0861 | 0.5400 | 0.0901 |
| *Frozen prior, both keyframes observed* | | | |
| both keyframes, no correction | 0.0626 | 0.6157 | 0.0682 |
| **+ linear correction (Bumblebee)** | **0.0581** | **0.6260** | **0.0649** |
| + minimum-jerk correction | <u>0.0587</u> | <u>0.6251</u> | <u>0.0656</u> |
| + regularized correction | 0.0608 | 0.6219 | 0.0680 |

**Ablation on the Motion80 keyframe setting** (80 identities): what the frozen prior observes and how the exported residual is closed; all rows share the frozen prior. Chamfer and F-score from the released evaluator over all sixteen frames. Inner Chamfer: the per-frame Chamfer of the same evaluator averaged over the fourteen unobserved frames $t = 1, \ldots, 14$. Best in bold, second best underlined.

# I Intermediate-Frame Accuracy and Local Surface Consistency

## I.1 Endpoint-Excluded Evaluation

We evaluate the fourteen unobserved frames $t = 1, \ldots, 14$ to assess trajectory accuracy away from the given keyframes. Table 10 follows the methods and input configurations of Table 1. We average the released Motion 3-to-4 evaluator's per-frame Chamfer and F-score over these frames and then over sequences. Motion80 uses 80 identities. The supplementary ActionBench-GT evaluation uses 128 sequences and their ground-truth meshes under the same mesh evaluation protocol. The official ActionBench metrics are reported separately in Table 1.

Geometric feedback improves accuracy beyond the observed endpoints. Relative to the same two-keyframe decode without correction, it lowers intermediate-frame Chamfer from 0.0682 to 0.0649 on Motion80 and from 0.0819 to 0.0600 on ActionBench-GT. F-score also increases on both splits. These results show that the correction improves the unobserved trajectory as well as enforcing endpoint agreement.

## I.2 Local Surface Consistency

We isolate the effect of the correction on local surface structure. All three settings in Table 11 use the same two-keyframe decoded trajectories. We compare linear vertex correction, regularized vertex correction, and part-state correction on 20 transformation clips and 14 Menagerie models.

Table 10: **Geometric accuracy on the fourteen intermediate frames.** The two endpoint frames are excluded. Chamfer values are unscaled. Best results among the reported configurations are in bold.

| Method | Motion80 | | ActionBench-GT | |
|---|---|---|---|---|
| | Chamfer↓ | F-score↑ | Chamfer↓ | F-score↑ |
| *Shape interpolation* | | | | |
| Linear | 0.1275 | 0.4147 | 0.0743 | 0.6368 |
| Minimum-jerk | 0.1280 | 0.4149 | 0.0751 | 0.6372 |
| ARAP | 0.1268 | 0.4153 | 0.0756 | 0.6318 |
| *Video-driven generation, released interfaces* | | | | |
| V2M4, video + source | 1.3683 | 0.0781 | 0.6229 | 0.1078 |
| Motion 3-to-4, video + source | 0.0892 | 0.5187 | 0.1134 | 0.5664 |
| ActionMesh, video | 0.2607 | 0.2486 | – | – |
| ActionMesh, video + source | 0.0901 | 0.5102 | 0.1063 | 0.6287 |
| ActionMesh, video + source + target | 0.0682 | 0.5679 | 0.0819 | 0.7002 |
| **Bumblebee** | **0.0649** | **0.5727** | **0.0600** | **0.7122** |

All results are reported on the full Motion80 split. A dash denotes an unavailable result in this supplementary evaluation. Baselines retain their original pipelines and do not use our correction.

A face is flagged when its unit normal has a negative dot product with the sum of the unit normals of its edge-adjacent faces. We report the percentage of faces flagged in the output but not in the ground truth at the same frame. This excess local normal inconsistency rate measures local normal agreement. It complements the rigidity and edge-length metrics in Table 2.

Table 11: **Effect of correction on local surface consistency.** Excess local normal inconsistency rates are percentages. All settings share the same two-keyframe decode. Lower is better.

| Correction | Transformation (20 clips) | | Menagerie (14 models) | |
|---|---|---|---|---|
| | Mean↓ | Temporal max.↓ | Mean↓ | Temporal max.↓ |
| Linear vertex correction | 9.794 | 14.645 | 1.549 | 2.016 |
| Regularized vertex correction | 6.589 | 11.460 | 1.431 | 2.040 |
| **Part-state correction** | **0.083** | **0.177** | **0.008** | **0.010** |

Mean averages frames $1, \ldots, 14$ within each sequence and then averages sequences. Temporal max. takes the maximum over all 16 frames of each sequence and then averages sequences.

Part-state correction lowers the mean excess local normal inconsistency rate from 9.794% to 0.083% on the transformation split and from 1.549% to 0.008% on Menagerie, compared with linear vertex correction. It also has the lowest temporal-maximum score on both splits. Together with the rigidity and edge-length results, this comparison supports correcting whole-part motion to preserve local mechanical structure.

## J Runtime

Bumblebee takes approximately 1.2 minutes per sequence on an RTX 5090 GPU, using 15 denoising steps and linear geometric feedback correction.

## K Part-State Correction and Part Geometry

This appendix gives the full part-state correction of Section 4.2 and analyzes its effect on part geometry.

**Step 1: Extract the position and orientation of each part.** We first determine where each part is located along the blue trajectory and how much it has rotated relative to its initial state. Our correction takes a vertex partition as input. By default, we use the connected components of the source mesh as

parts. We also estimate a partition from vertex correspondences between the two given keyframes through rigid fitting and vertex reassignment. Both partitions feed the same part-state correction without requiring a rig or manual part labels (Appendix F.1). Let $\mathcal{P}_k$ denote the vertex set of part $k$, $v_{s,i}$ the initial vertex position, and $c_{s,k}$ the center of the initial part. Let $q_{k,i}$ denote the vertex position relative to this center, defined as

$$c_{s,k} = \frac{1}{|\mathcal{P}_k|} \sum_{i \in \mathcal{P}_k} v_{s,i}, \qquad q_{k,i} = v_{s,i} - c_{s,k}. \tag{14}$$

Here, $q_{k,i}$ encodes the original shape of the part. We then extract motion from the decoded results and use this motion to position the original part.

Let $\widetilde{v}_{t,i}$ denote the decoded vertex position after alignment to the source mesh coordinate system. Let $c_{t,k}$ and $Q_{t,k}$ denote the part center and rotation at frame $t$, respectively, defined as

$$c_{t,k} = \frac{1}{|\mathcal{P}_k|} \sum_{i \in \mathcal{P}_k} \widetilde{v}_{t,i}, \tag{15}$$

$$Q_{t,k} = \underset{Q \in SO(3)}{\arg\min} \sum_{i \in \mathcal{P}_k} \|Q q_{k,i} + c_{t,k} - \widetilde{v}_{t,i}\|_2^2 . \tag{16}$$

Here, $SO(3)$ denotes the set of three-dimensional rotation matrices. The first equation extracts the center position of the part. The second equation finds the whole-part rotation that best matches the decoded result. This gives the position and orientation of each part along the blue trajectory.

**Step 2: Compare endpoint states and extract the required translation and rotation compensation.** Let $c_{e,k}$ and $Q_{e,k}$ denote the part center and rotation extracted from the target mesh in the same way. For the 16-frame sequence in Figure 3, let $d_{s,k}$ and $d_{e,k}$ denote the translation feedback at the two endpoints, and let $D_{s,k}$ and $D_{e,k}$ denote the rotation feedback, defined as

$$\begin{aligned} d_{s,k} &= c_{s,k} - c_{0,k}, \qquad & d_{e,k} &= c_{e,k} - c_{15,k}, \\ D_{s,k} &= Q_{0,k}^{\mathsf{T}}, & D_{e,k} &= Q_{e,k} Q_{15,k}^{\mathsf{T}}. \end{aligned} \tag{17}$$

This feedback expresses the geometric differences at the endpoints in terms of individual parts. Translation feedback specifies where and how far a part needs to move. Rotation feedback specifies how the part needs to rotate to reach its orientation in the given keyframe.

**Step 3: Propagate endpoint feedback to intermediate frames and correct whole-part motion.** Let $\tau_t = t/15$, and define a smoothly varying temporal weight as

$$w_t = 10\tau_t^3 - 15\tau_t^4 + 6\tau_t^5. \tag{18}$$

Let $\Phi(D, w)$ denote scaling the angle of rotation $D$ by a factor of $w$ while keeping its rotation axis unchanged. Let $\overline{c}_{t,k}$ and $\overline{Q}_{t,k}$ denote the corrected part center and rotation, respectively. We construct the part-state correction equations from the feedback at the two endpoints as

$$\begin{aligned} \overline{c}_{t,k} &= c_{t,k} + (1 - w_t) d_{s,k} + w_t d_{e,k}, \\ \overline{Q}_{t,k} &= \Phi(D_{e,k}, w_t)\, \Phi(D_{s,k}, 1 - w_t)\, Q_{t,k}. \end{aligned} \tag{19}$$

We then place the original part at the corrected position and orientation, yielding

$$\overline{v}_{t,i} = \overline{Q}_{t,k} q_{k,i} + \overline{c}_{t,k}, \qquad i \in \mathcal{P}_k. \tag{20}$$

In this step, all vertices of the same part undergo the same translation and rotation. The distances between vertices within the part therefore remain unchanged. Mechanical parts such as car doors can move and rotate as a whole while retaining their original shapes. **Step 4: Compensate for the remaining geometric deviations to match the endpoint meshes exactly.** Whole-part translation and rotation cannot eliminate shape differences between the source and target parts. Let $\overline{V}_t$ denote the vertex matrix obtained in Step 3. We use the geometric feedback correction of Section 4.1 to compensate for the remaining endpoint deviations as

$$\widehat{V}_t = \overline{V}_t + (1 - \tau_t)(V_s - \overline{V}_0) + \tau_t (V_e - \overline{V}_{15}). \tag{21}$$

$$\text{The final output satisfies} \qquad \widehat{V}_0 = V_s, \qquad \widehat{V}_{15} = V_e. \tag{22}$$

Step 3 adjusts each part through whole-part translation and rotation and matches the source mesh exactly. Step 4 therefore compensates only for the remaining target residual. The change in any within-part distance is bounded by the temporal weight times the norm of the difference between the two vertex residuals (Lemma 1). If the target part is a rigid transform of the source part, its target fitting residual is zero. In this case, Step 4 leaves the part shape unchanged, and the final trajectory preserves part rigidity at every frame.

We analyze the four-step correction above for a fixed vertex partition $\mathcal{P}_k$. The analysis uses rotations in $SO(3)$ and exact arithmetic. Step 3 places each source part through a rigid transform,

$$\overline{v}_{t,i} = \overline{Q}_{t,k} q_{k,i} + \overline{c}_{t,k}, \qquad q_{k,i} = v_{s,i} - c_{s,k}. \tag{23}$$

The endpoint properties $\Phi(D, 0) = I$ and $\Phi(D, 1) = D$ give

$$\begin{aligned} \overline{c}_{0,k} &= c_{0,k} + d_{s,k} = c_{s,k}, \qquad \overline{Q}_{0,k} = Q_{0,k}^{\mathsf{T}} Q_{0,k} = I, \\ \overline{c}_{15,k} &= c_{15,k} + d_{e,k} = c_{e,k}, \quad \overline{Q}_{15,k} = Q_{e,k} Q_{15,k}^{\mathsf{T}} Q_{15,k} = Q_{e,k}. \end{aligned} \tag{24}$$

Thus, $\overline{v}_{0,i} = v_{s,i}$. Define the remaining target residual as

$$r_{k,i} = v_{e,i} - \overline{v}_{15,i} = v_{e,i} - (Q_{e,k} q_{k,i} + c_{e,k}). \tag{25}$$

The source residual in Eq. (21) is zero. Step 4 therefore reduces to

$$\widehat{v}_{t,i} = \overline{v}_{t,i} + \tau_t r_{k,i}, \qquad \tau_t = t/15. \tag{26}$$

We now prove Lemma 1 stated in Section 4.2.

*Proof.* Subtracting the positions of two vertices cancels the translation,

$$\widehat{v}_{t,i} - \widehat{v}_{t,j} = \overline{Q}_{t,k}(q_{k,i} - q_{k,j}) + \tau_t (r_{k,i} - r_{k,j}). \tag{27}$$

Since $\overline{Q}_{t,k} \in SO(3)$, it preserves Euclidean lengths. Hence,

$$\|\overline{Q}_{t,k}(q_{k,i} - q_{k,j})\|_2 = \|q_{k,i} - q_{k,j}\|_2 = \|v_{s,i} - v_{s,j}\|_2. \tag{28}$$

Applying the reverse triangle inequality gives Eq. (12). If the endpoint parts are rigidly compatible, the target rigid-fitting problem has a zero-error solution. Its least-squares minimum is therefore zero, so $r_{k,i} = 0$ for every vertex. Equation (26) then gives $\widehat{v}_{t,i} = \overline{v}_{t,i}$, which is a rigid transform of the source part at every frame. □

**Relation to edge distortion.** For a nonzero source edge of length $\ell_{ij} = \|v_{s,i} - v_{s,j}\|_2$, the relative edge-length change obeys

$$\frac{|\|\widehat{v}_{t,i} - \widehat{v}_{t,j}\|_2 - \ell_{ij}|}{\ell_{ij}} \leq \tau_t \frac{\|r_{k,i} - r_{k,j}\|_2}{\ell_{ij}}. \tag{29}$$

The bound depends on residual differences across vertices. A shared residual moves the whole part without changing its shape. These statements apply within each estimated group. Appendix F.1 evaluates the final trajectories on the true asset parts to assess how the chosen partition preserves the original mechanical structure.

## L RECONSTRUCTION ERROR DISTRIBUTIONS

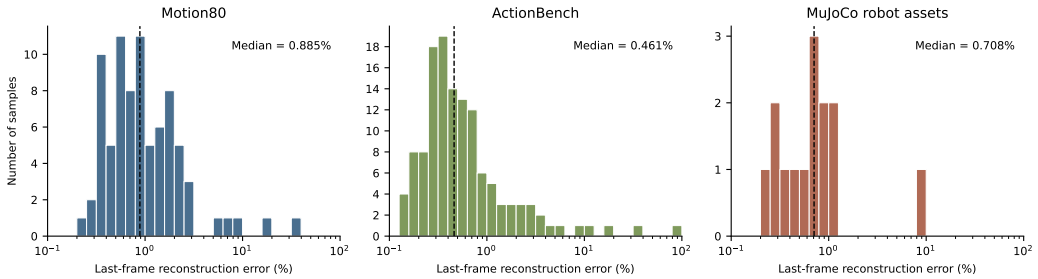



Figure 9: Last-frame reconstruction error distributions for all 80 Motion80 samples, all 128 ActionBench samples, and the sequences built from MuJoCo Menagerie robot assets. Error is the mean corresponding-point distance divided by the source bounding-box longest side. ActionBench uses tracked surface points, while the other groups use mesh vertices. The horizontal axis is logarithmic, and dashed lines mark medians.