MESHLOOM: FEED-FORWARD NON-RIGID REGISTRATION OF MESH SEQUENCES

Anonymous authors
Paper under double-blind review

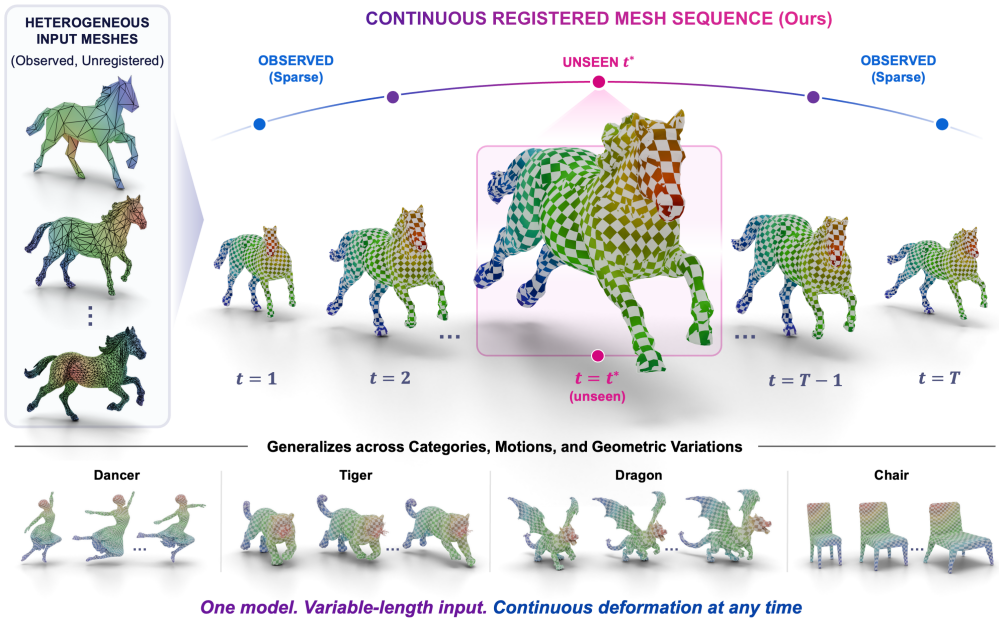

Figure 1: **MeshLoom unifies input meshes into a topology-consistent output sequence.** Beyond registering the observed input frames, our network can also synthesize mesh deformations at arbitrary unseen intermediate timestamps. The method generalizes across diverse object categories and motions, and remains robust under geometric variations. The registered meshes are rendered using the same colored coordinates as the anchor mesh for visually apparent vertex correspondences across frames. Please refer to our *supplementary webpage* for interactive 3D visualization results.

ABSTRACT

We present MeshLoom, a feed-forward registration network that directly reconstructs vertex deformations across mesh sequences. Our approach advances non-rigid registration beyond existing models, which are typically constrained by costly per-instance optimization, narrow object categories, pairwise-only inputs, or merely intermediate outputs. The network is simple and efficient, registering multiple meshes within seconds. At its core lies a topology-aware encoder–decoder design. Specifically, we first introduce a topology-aware point representation that encodes the anchor (reference) mesh's topology into its per-vertex features. This representation strengthens the network's understanding of the anchor-mesh geometry and disambiguates points that are Euclidean-close yet geodesically distant. We then propose a multi-modal encoder that fuses this anchor-mesh representation with complementary cues from each frame, such as shape latents and image features. These multi-source signals are compressed into a compact global motion embedding that captures dense inter-frame correspondence. A lightweight decoder then queries this global embedding with the anchor-mesh point representation, retrieving per-vertex deformations at target timestamps. Through extensive experiments across diverse motions and object categories, we show that MeshLoom achieves state-of-the-art results on non-rigid registration. In addition,

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.

the decoder can be queried at intermediate timestamps, which extends MeshLoom to motion interpolation and mesh morphing.

1 INTRODUCTION

Non-rigid mesh registration (Deng et al., 2022), the task of recovering a continuous deformation field that aligns a source shape to one or more target shapes, is a long-standing problem in computer graphics and computer vision. Producing a registered mesh sequence with a single, persistent topology is fundamental to many downstream applications, including motion capture, animation retargeting, and physically-based simulation. More recently, it has also become the missing link for extending modern 3D foundation models (Zhao et al., 2025b; Xiang et al., 2025a) from static 3D reconstruction to dynamic 4D reconstruction, in which the reconstructed 3D shapes vary in vertex count and connectivity across frames. Despite decades of progress, a registration solution that is simultaneously fast, accurate, and broadly applicable across object categories remains elusive.

Traditional registration methods rely on spatial deformation optimization (Li et al., 2009; Chang & Zwicker, 2011; Jian & Vemuri, 2005) or intrinsic descriptor matching (Bronstein et al., 2006; Lipman & Funkhouser, 2009). However, they are often slow and highly sensitive to initialization and topological variations, making them impractical for modern dynamic-content pipelines. Recent learning-based registration methods have emerged to address these issues by leveraging the representational power of neural networks. Yet, these methods are typically bottlenecked along one or more of the following axes. *(i) Domain-restricted* models (Groueix et al., 2018; Marin et al., 2024; Gao et al., 2026) do not reliably generalize across object categories. *(ii) Iterative* models (Li & Harada, 2022; Feng et al., 2023; Jiang et al., 2023) still require test-time optimization or repeated refinement passes, sacrificing the speed advantage of feed-forward inference. *(iii) Indirect* models (Aigerman et al., 2022; Sundararaman et al., 2022; Sun et al., 2023) do not regress vertex positions directly, but predict an intermediate field, such as per-face Jacobians or a correspondence map, from which the deformation is recovered by a global linear solve on the mesh or by a map-conversion step. *(iv) Pairwise* models (Trappolini et al., 2021; Sundararaman et al., 2024; Cao et al., 2024) register only a single source–target pair at a time, and cannot ingest variable-length sequences in one pass. Crucially, addressing all of these limitations within a single framework remains underexplored.

To close this gap, we propose MeshLoom, a generalizable feed-forward framework that addresses all of the above limitations within a single model. It encodes an entire mesh sequence into a compact motion embedding and queries it with the anchor mesh to recover dense deformations. We first introduce a topology-aware point representation for the anchor mesh (*i.e.*, the reference mesh that defines the canonical vertex set), enabling the network to better extract and comprehend its geometry. Specifically, by augmenting each point's position with features aggregated over its mesh neighborhood, we embed the anchor mesh's topology into per-point descriptors. This disambiguates Euclidean-close yet geodesically distant vertices and prevents them from erroneously moving together under deformation. Then an encoder fuses the anchor-mesh representation with complementary cues from each mesh frame across the sequence, such as shape latents and image features. These multi-modal signals are projected into a compact global motion embedding that captures dense inter-frame correspondence. A lightweight deformation decoder then queries this embedding with the anchor-mesh point representation, retrieving per-vertex deformations at target timestamps.

Extensive evaluation across diverse object categories and motion patterns demonstrates that MeshLoom is effective, efficient, and broadly generalizable (Fig. 1 shows selected results). Even against the most recent feed-forward attempt, ActionMesh (Sabathier et al., 2026), our network is markedly more accurate and robust. Furthermore, the decoder can also be queried at intermediate timestamps, extending the method to applications such as motion interpolation and mesh morphing.

Our main contributions are summarized as follows.

- We propose MeshLoom, a feed-forward non-rigid mesh registration framework that, in a single model, supports variable-length inputs, predicts per-vertex deformations, and generalizes across diverse object categories, resolving several bottlenecks of existing registration methods.
- We introduce a topology-aware point representation that augments each point with features aggregated over its mesh neighborhood. By embedding topology into per-point descriptors, it disambiguates Euclidean-close yet geodesically distant points and alleviates vertex entanglement.

- We design a multi-modal motion encoder that fuses the anchor representation with the shape latents and optional image features of all frames in one pass, and a lightweight decoder that queries the resulting embedding for per-vertex deformations at any timestamp, enabling motion interpolation and mesh morphing.

2 Related Work

Classical non-rigid registration. Unlike rigid registration, which estimates a single global rotation and translation, non-rigid registration aims to recover a deformation field that aligns a source surface to a target. Common deformation parameterizations include per-vertex displacements or local affine transforms (Huang et al., 2011; Allen et al., 2003), deformation graphs (Sumner et al., 2007), and Gaussian kernel-based fields (Myronenko et al., 2006). Classical methods roughly split into two complementary families. *Extrinsic* methods (Li et al., 2009; Chang & Zwicker, 2011; Jian & Vemuri, 2005) optimize the deformation directly in the ambient 3D space, typically minimizing point-to-point, point-to-plane, or probabilistic distances together with regularizers that enforce smoothness and local rigidity. *Intrinsic* methods (Ovsjanikov et al., 2012; Solomon et al., 2012; Rustamov et al., 2013) instead exploit surface-intrinsic properties and focus on correspondence estimation rather than explicit deformation. Examples include minimum-distortion approaches (Rodola et al., 2012; Anguelov et al., 2004; Edelstein et al., 2020) that preserve geodesic or conformal structure, and spectral or functional methods (Kovnatsky et al., 2013; Cosmo et al., 2016) that represent mappings in a low-frequency basis. The two families have complementary failure modes: extrinsic methods are sensitive to initialization and prone to local minima, while intrinsic methods are vulnerable to topology change and noise. Hybrid schemes bridge the gap by combining intrinsic correspondence cues with extrinsic geometric registration or refinement (Marin et al., 2020; Eisenberger et al., 2020; Ezuz et al., 2019), but typically rely on costly iterative optimization.

Learning-based non-rigid registration. To address the limitations of the classical methods, recent work has turned to neural networks for non-rigid registration. By replacing individual stages of the classical pipeline with learned counterparts, such as learned shape descriptors, learned functional maps, or networks that directly regress deformation, these methods have achieved encouraging progress. They remain, however, bottlenecked along one or more orthogonal axes. *Domain-restricted* models (Groueix et al., 2018; Marin et al., 2024; Gao et al., 2026) learn category-specific shape spaces, typically of humans or animals, and do not well generalize to in-the-wild objects. *Iterative* models (Li & Harada, 2022; Feng et al., 2023; Jiang et al., 2023; Chen et al., 2025; Bednarik et al., 2021) embed the network inside a refinement loop or rely on per-instance test-time fitting, sacrificing the speed advantage of feed-forward inference. *Indirect* models (Aigerman et al., 2022; Sundararaman et al., 2022; Sun et al., 2023; Pierson et al., 2025) predict an intermediate field, such as per-face Jacobians or a functional map, from which vertex positions are recovered by a linear solve or a map-conversion step. *Pairwise* models (Trappolini et al., 2021; Sundararaman et al., 2024; Cao et al., 2024) register only a single source–target pair, leaving multi-frame consistency as a downstream concern. Together, these bottlenecks prevent existing learning-based methods from meeting the demands of modern dynamic-content pipelines. In contrast, we propose a feed-forward network that addresses all of these issues in a unified, efficient, and robust manner. The closest feed-forward peer is the recent ActionMesh (Sabathier et al., 2026), which confirms the viability of feed-forward registration at scale but breaks down in challenging cases. Ours is overall more accurate and robust than ActionMesh.

3 Method

Let $\mathcal{M} = \{\mathcal{M}_t\}_{t=1}^{T}$ denote an input sequence of T meshes, where $\mathcal{M}_t = (\mathcal{V}_t, \mathcal{F}_t)$ has vertices $\mathcal{V}_t \in \mathbb{R}^{N_t \times 3}$ and faces $\mathcal{F}_t$. Both the vertex count and the face connectivity may differ across frames, *i.e.*, $N_{t_1} \neq N_{t_2}$ or $\mathcal{F}_{t_1} \neq \mathcal{F}_{t_2}$ for two distinct frames t_1, t_2. Without loss of generality, we designate the first frame as the *anchor* mesh $\mathcal{M}_a = (\mathcal{V}_a, \mathcal{F}_a) = (\mathcal{V}_1, \mathcal{F}_1)$, which contains $N = N_1$ vertices. Its vertices $\mathcal{V}_a$ are deformed to match each of the remaining frames, so all outputs share its connectivity $\mathcal{F}_a$ (any other frame can serve as the anchor by reordering the sequence). The network then unifies these mesh frames into a topology-consistent output sequence $\widetilde{\mathcal{M}} = \mathrm{Network}(\mathcal{M}) = \left\{ (\widetilde{\mathcal{V}}_t,\, \mathcal{F}_a) \right\}_{t=1}^{T}$ with $\widetilde{\mathcal{V}}_t \in \mathbb{R}^{N \times 3}$, so that all frames share the topology $\mathcal{F}_a$ and vertex count N.

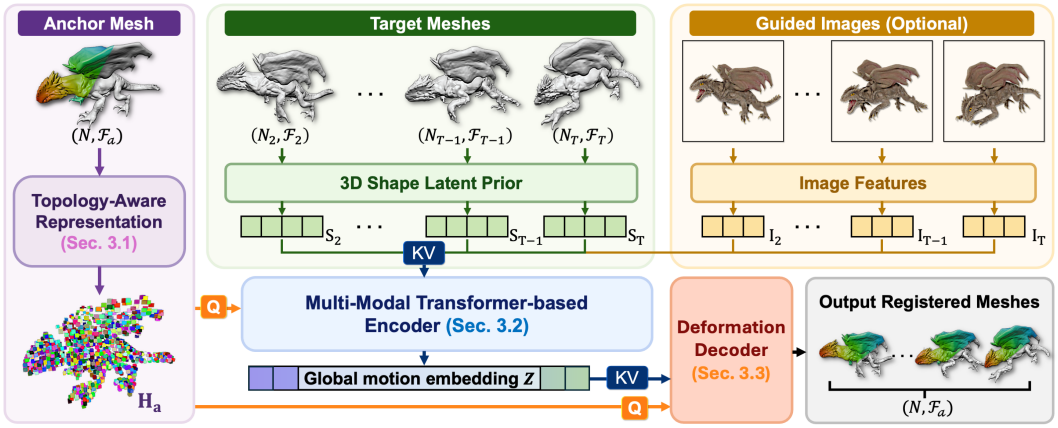

Figure 2: **Workflow of MeshLoom.** Given an input mesh sequence whose frames differ in vertex count N_t and connectivity $\mathcal{F}_t$, together with (optional) accompanying reference images, our network proceeds in three steps. (1) We designate the first frame as the anchor mesh and embed it into a topology-aware representation $\mathbf{H}_a$. (2) The remaining frames and their images are encoded into per-frame shape latents $\mathbf{S}_t$ and image features $\mathbf{I}_t$ (the anchor frame is encoded in the same way; omitted here for brevity), and all features are fused with $\mathbf{H}_a$ by a transformer-based encoder to produce a global motion embedding $\mathbf{Z}$. (3) A lightweight decoder then queries $\mathbf{Z}$ with $\mathbf{H}_a$ to predict the per-vertex deformation of the anchor mesh at every frame, yielding an output sequence with a consistent vertex count N and face connectivity $\mathcal{F}_a$ across all frames.

Fig. 2 illustrates our topology-aware encoder–decoder pipeline. First, we introduce a *topology-aware point representation* (Sec. 3.1) for the anchor mesh. The multi-modal *motion encoder* (Sec. 3.2) then fuses this anchor-mesh representation with multi-source cues across all T mesh frames, producing a global motion embedding $\mathbf{Z}$ that implicitly encodes inter-frame correspondence. The *deformation decoder* (Sec. 3.3) queries $\mathbf{Z}$ with the anchor-mesh representation and predicts the per-vertex deformation at each frame. Appendix B gives further network details.

3.1 Topology-Aware Point Representation

Since every output frame inherits the vertex count and face connectivity of the anchor mesh, it is crucial for the network to faithfully understand its shape and topology.

Mainstream networks (Zhang et al., 2023; Xiang et al., 2025b;a) encode only vertex positions and normals (Sabathier et al., 2026), ignoring the mesh connectivity. They thus fail to distinguish two vertices that are Euclidean-close but topologically distant, *e.g.*, a sleeve–torso contact (Fig. 3), and are prone to vertex entanglement, where such close-but-disjoint vertices erroneously move together.

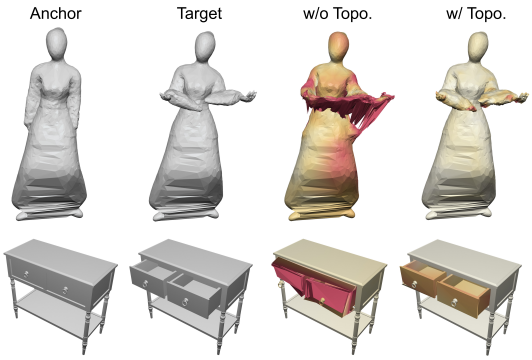

Figure 3: **Effect of the topology-aware point representation.** Comparison with and without topology information in the per-vertex features. Left to right: the anchor, the target, and each variant's result, colored by per-vertex tracking error.

To resolve this ambiguity, we employ a GCN-style (Kipf & Welling, 2017) network to extract topology-aware features for each vertex. Intrinsic mesh networks such as DiffusionNet (Sharp et al., 2022) and PoissonNet (Maesumi et al., 2025) are also natural choices here, but we empirically find them less effective than our simple design (Appendix H). The network takes a vertex feature matrix $\mathbf{H} \in \mathbb{R}^{N \times d}$ as input and aggregates mesh connectivity by propagating vertex features across the graph structure. The feature matrix $\mathbf{H}$ is constructed by concatenating the position-embeddings and surface normals of the mesh. To implement the aggregation, we define a row-normalized adjacency operator $\mathbf{A} \in \mathbb{R}^{N \times N}$ induced by the face set $\mathcal{F}_a$, with $A_{ij} = 1/\deg(i)$ if $(i, j) \in \mathcal{E}$ and $A_{ij} = 0$ otherwise, where $\mathcal{E}$ denotes the edge set of the input mesh and $\deg(i)$ is the degree of vertex

i. Therefore, the i-th row of $\mathbf{AH}$ represents the feature average across the one-ring neighborhood of vertex i. To capture the overall topology, we apply $\mathbf{A}$ repeatedly, so that $\mathbf{A}^p\mathbf{H}$ aggregates p-hop neighborhoods, and stack L residual blocks on the mesh graph:

$$\mathbf{H}^{(\ell+1)} = \mathrm{LayerNorm}\Big(\mathbf{H}^{(\ell)} + \mathrm{MLP}\Big(\Big[\mathbf{H}^{(\ell)} \,\Big\|\, \mathbf{A}^p\mathbf{H}^{(\ell)}\Big]\Big)\Big) \tag{1}$$

where $[\cdot\|\cdot]$ denotes channel-wise concatenation. We define the final layer output $\mathbf{H}^L$ as the topology-aware representation. For anchor mesh, we denote its corresponding feature matrix as $\mathbf{H}_a \in \mathbb{R}^{N\times d}$.

3.2 Motion Encoder

The motion encoder builds a global embedding that describes the input sequence and captures inter-frame correspondence. To enrich the input, the encoder ingests multiple sources: the topology-aware anchor representation $\mathbf{H}_a$ and per-frame features such as 3D shape-latent priors and image features, which several transformer blocks process into the global motion embedding $\mathbf{Z}$.

Encoder Input. A natural design is to feed topology-aware features from every frame. However, this requires running the GCN-style network for every frame, resulting in substantial computational overhead. Empirically, we find that using only the anchor-mesh representation achieves comparable performance while significantly reducing computation (see Table 3). Therefore, we use only $\mathbf{H}_a$ as the encoder input. Specifically, since the original feature matrix is dense, we first downsample $\mathbf{H}_a$ to N_{ds} points using farthest-point sampling (FPS), obtaining $\mathbf{H}_a^{\mathrm{ds}} \in \mathbb{R}^{N_{\mathrm{ds}}\times d}$. We then replicate $\mathbf{H}_a^{\mathrm{ds}}$ across all T time steps to form the initial encoder input $\mathbf{Z}^{(0)} \in \mathbb{R}^{T\times N_{\mathrm{ds}}\times d}$.

Multi-Modal Transformer Block. Each transformer block comprises two cross-attention layers followed by two self-attention layers. The two cross-attention layers attend to two complementary sources, respectively. The first is a per-frame *3D shape-latent prior* $\{\mathbf{S}_t\}_{t=1}^T$, where each $\mathbf{S}_t \in \mathbb{R}^{N_s\times d}$ is the latent of frame t produced by an off-the-shelf pretrained 3D foundation model (Xiang et al., 2025a) (the size of $\mathbf{S}_t$ may vary across frames; we omit this dependency in the notation for clarity). Having been trained on large-scale 3D-asset datasets, $\mathbf{S}_t$ encodes strong priors over mesh geometry that sharpen the encoder's understanding of per-frame shape structure. In addition, we optionally allow the network to leverage per-frame *image features* $\{\mathbf{I}_t\}_{t=1}^T$, where $\mathbf{I}_t \in \mathbb{R}^{N_m\times d}$ is extracted from a reference image of frame t by a pretrained image encoder (Siméoni et al., 2025) and carries object-appearance cues that further enrich the semantic representation. This optional conditioning is compatible with modern Image-to-3D pipelines, where reference images are readily available. Formally, given the input tokens $\mathbf{Z}_t \in \mathbb{R}^{N_{\mathrm{ds}}\times d}$, the feature injection is defined as:

$$\mathbf{Z}_t^s = \mathrm{CrossAttn}\big(\mathrm{Q}\leftarrow\mathbf{Z}_t,\ \mathrm{KV}\leftarrow\mathbf{S}_t\big), \qquad \mathbf{Z}_t^{s,m} = \mathrm{CrossAttn}\big(\mathrm{Q}\leftarrow\mathbf{Z}_t^s,\ \mathrm{KV}\leftarrow\mathbf{I}_t\big) \tag{2}$$

where the Q/KV tags indicate which input provides the queries and which provides the keys and values, and $\mathbf{Z}_t^s$ and $\mathbf{Z}_t^{s,m}$ represent the features enriched by the 3D shape prior and image guidance.

The self-attention layers consist of an *inter-frame* self-attention, which fuses temporal information, and an *intra-frame* self-attention, which refines the spatial structure of each frame. Letting $\mathbf{Z}_{1:T}$ denote the concatenation of all per-frame tokens at the current layer, the former is computed once over all $T\cdot N_{\mathrm{ds}}$ tokens as $\mathrm{SelfAttn}(\mathrm{QKV}\leftarrow\mathbf{Z}_{1:T})$, and the latter separately for each frame as $\mathrm{SelfAttn}(\mathrm{QKV}\leftarrow\mathbf{Z}_t)$, $t=1,\dots,T$. After several transformer blocks, the output tokens of all frames are concatenated into the final global motion embedding $\mathbf{Z}\in\mathbb{R}^{(T\cdot N_{\mathrm{ds}})\times d}$.

3.3 Deformation Decoder

Given the global embedding $\mathbf{Z}$, the decoder queries it to obtain per-vertex deformations of the anchor mesh at any timestep, yielding a mesh sequence with consistent topology. Specifically, for an anchor vertex $\mathbf{v}_i\in\mathbb{R}^3$ ($i\in\{1,\dots,N\}$) and a target timestamp t, the decoder predicts a deformed position $\widetilde{\mathbf{v}}_i^{(t)}\in\mathbb{R}^3$. Rather than regressing the full displacement directly, we decompose it into two components: a *global centroid shift* $\Delta\mathbf{c}^{(t)}\in\mathbb{R}^3$ and a *local residual* $\Delta\mathbf{r}_i^{(t)}\in\mathbb{R}^3$:

$$\widetilde{\mathbf{v}}_i^{(t)} = \mathbf{v}_i + \Delta\mathbf{c}^{(t)} + \Delta\mathbf{r}_i^{(t)} \tag{3}$$

Specifically, the global centroid shift denotes the rigid translation of the mesh centroid at frame t. The local residual describes the per-vertex non-rigid deformation in the local frame. Compared with a single-branch design, we empirically find that this decoupling improves robustness under large translational motion and suppresses high-frequency surface artifacts (see Fig. 4).

Global-translation branch. To estimate the global translation $\Delta\mathbf{c}^{(t)}$ at any given time t, we first apply average pooling to $\mathbf{Z}$ across all $T \cdot N_{\text{ds}}$ tokens. This yields a single sequence-level descriptor $\bar{\mathbf{z}} \in \mathbb{R}^d$, which encodes the global motion trajectory of the entire animation. To resolve the specific translation for a target timestamp t, we concatenate $\bar{\mathbf{z}}$ with the Fourier time embedding $\phi(t) \in \mathbb{R}^{d_\phi}$ and predict the centroid shift through a lightweight MLP: $\Delta\mathbf{c}^{(t)} = \text{MLP}\big([\bar{\mathbf{z}} \,\|\, \phi(t)]\big)$.

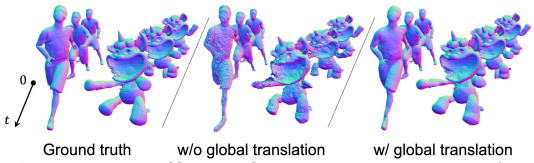

Figure 4: **Effect of the global-translation branch.** Left to right: ground truth, results of a single-branch decoder, and results of our two-branch decoder with global-translation prediction; zoom in to compare surface smoothness.

Local-deformation branch. To predict the local residual $\Delta\mathbf{r}_i^{(t)}$, we employ a single cross-attention block whose query is the topology representation $\mathbf{h}_i \in \mathbb{R}^d$ (i-th row of $\mathbf{H}_a$) of vertex i, modulated with the target timestamp t via FiLM (Perez et al., 2018), and whose keys and values are the global motion embedding $\mathbf{Z}$. An MLP then projects the output to the local residual:

$$\Delta\mathbf{r}_i^{(t)} = \text{MLP}\big(\text{CrossAttn}\big(\text{Q}\leftarrow\text{FiLM}(\mathbf{h}_i, t),\ \text{KV}\leftarrow\mathbf{Z}\big)\big) \tag{4}$$

Since the decoder predicts the deformation at timestep t from $\mathbf{Z}$ conditioned only on t, $\mathbf{Z}$ can be kept fixed while t varies, which in principle yields deformations at timestamps that the encoder has never observed. We exploit this during training by also supervising the decoder at such timestamps (Appendix C), which extends MeshLoom to motion interpolation and mesh morphing (Sec. 4.3).

3.4 Training Objectives

We train MeshLoom end-to-end with several objectives. The global and local deformations predicted in Sec. 3.3 are directly supervised with a mean-squared error:

$$\mathcal{L}_{\text{global}} = \big\|\Delta\mathbf{c}^{(t)} - \Delta\mathbf{c}^{*(t)}\big\|_2^2, \qquad \mathcal{L}_{\text{local}} = \frac{1}{N}\sum_{i=1}^{N}\big\|\Delta\mathbf{r}_i^{(t)} - \Delta\mathbf{r}_i^{*(t)}\big\|_2^2 \tag{5}$$

where $\Delta\mathbf{c}^{*(t)}$ and $\Delta\mathbf{r}_i^{*(t)}$ are the ground-truth global translation and local residual, respectively.

Beyond direct vertex supervision, two losses on the anchor-mesh connectivity graph further facilitate training and reduce fold-overs and volume collapse (Fig. 5).

The *edge-vector loss* penalizes the L_1 difference between predicted and ground-truth edge vectors:

$$\mathcal{L}_{\text{edge}} = \frac{1}{|\mathcal{E}|}\sum_{(i,j)\in\mathcal{E}}\big\|\big(\widetilde{\mathbf{v}}_i^{(t)} - \widetilde{\mathbf{v}}_j^{(t)}\big) - \big(\mathbf{v}_i^{*(t)} - \mathbf{v}_j^{*(t)}\big)\big\|_1 \tag{6}$$

where $\mathbf{v}_i^{*(t)}$ is the ground-truth position of vertex i at time t, and $\mathcal{E}$ the anchor-mesh edge set.

To better capture the local deformation field, we further adopt a *Jacobian loss* that supervises the local deformation gradient (Sumner & Popović, 2004) at each vertex. Specifically, for vertex i, we estimate a Jacobian matrix $\mathbf{J}_i \in \mathbb{R}^{3\times3}$ that best fits the deformation of its neighborhood in a least-squares sense. The ground-truth Jacobian $\mathbf{J}_i^*$ is derived analogously. We penalize their discrepancy with a smooth-L_1 loss:

$$\mathcal{L}_{\text{jac}} = \frac{1}{N}\sum_{i=1}^{N}\text{SmoothL1}(\mathbf{J}_i,\ \mathbf{J}_i^*) \tag{7}$$

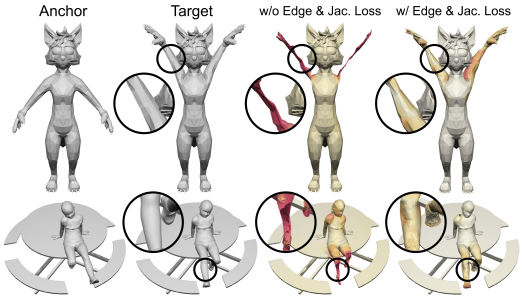

Figure 5: **Effect of the edge-vector and Jacobian losses.** Left to right: the anchor, the target, and the registered results under different loss configurations, colored by per-vertex tracking error; zoomed-in views highlight the differences.

This supervision constrains local structure more precisely than vertex-position loss alone. Unlike local-rigidity regularizers such as as-rigid-as-possible (Sorkine et al., 2007), which impose a strong inductive bias by constraining local transformations to be rigid, our Jacobian supervision adopts a data-driven approach. By supervising

Table 1: **Quantitative comparison on ActionBench and DeformingThings4D.** Correspondence metrics (CD-Motion; on DT4D also NGE (Kim et al., 2011)) and rendering-based metrics (CLIP, DreamSim), averaged over the 2-, 4-, and 8-frame settings (per-setting results: Appendix G). DT4D assets are mostly untextured, so only our image-free model (w/o img) is evaluated there. Inference time: 8-frame inputs, 150K vertices and 300K faces per frame. Best in **bold**, second-best underlined.

Method	ActionBench			DeformingThings4D				Inf. Time (s)
	CD-M ↓	CLIP ↑	DreamSim ↓	CD-M ↓	NGE ↓	CLIP ↑	DreamSim ↓	
TransMatch (Trappolini et al., 2021)	0.8457	0.7610	0.2455	0.6113	0.2469	0.8313	0.1595	14.2
NDP (Li & Harada, 2022)	0.1699	0.9666	0.0325	0.1472	0.0663	0.9544	0.0457	5.9
LNDP (Li & Harada, 2022)	0.1760	0.9605	0.0480	0.1820	0.0710	0.9150	0.1035	279.9
DPF (Prokudin et al., 2023)	0.1770	0.9715	0.0208	0.2397	0.1006	0.9537	0.0328	70.2
ClusterReg (Zhao et al., 2024)	0.1902	0.8752	0.1045	0.2988	0.1118	0.8784	0.1202	42.6
OAReg (Zhao et al., 2025a)	0.1592	0.9675	0.0334	0.1776	0.0773	0.9470	0.0587	25.3
ActionMesh (Sabathier et al., 2026)	0.1557	0.9707	0.0287	0.1310	0.0464	0.9698	0.0269	7.7
Ours (w/o img)	0.1493	0.9786	0.0197	**0.0940**	**0.0342**	**0.9759**	**0.0246**	**3.1**
Ours	**0.1471**	**0.9793**	**0.0184**	–	–	–	–	

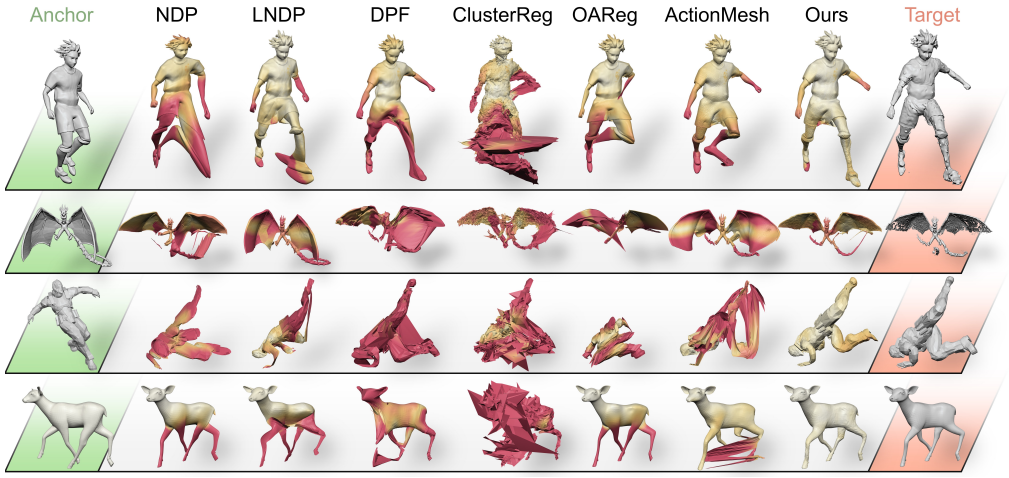

Figure 6: **Visual comparison with baseline methods on ActionBench (top two rows) and DeformingThings4D (bottom two rows).** From left to right: the anchor mesh, the registered results of each method, and the target mesh. Registered meshes are colored by per-vertex tracking error, from white (low) to red (high); on DT4D, ours runs image-free.

the fitted deformation gradient, it preserves the fidelity of the local deformation field and naturally accommodates non-rigid effects such as stretching and shearing. Appendix C gives implementation details of the losses and our training schemes, and Appendix D lists all hyperparameters.

The overall training objective is:

$$\mathcal{L} = \lambda_{\text{global}}\,\mathcal{L}_{\text{global}} + \lambda_{\text{local}}\,\mathcal{L}_{\text{local}} + \lambda_{\text{edge}}\,\mathcal{L}_{\text{edge}} + \lambda_{\text{jac}}\,\mathcal{L}_{\text{jac}} \tag{8}$$

4 Experiments

Datasets. We train our network on the animation subset of TexVerse (Zhang et al., 2025), which contains animated sequences of diverse categories such as characters, animals, and mechanical objects. After applying a series of filtering steps (detailed in Appendix E), approximately 30K high-quality 32-frame training sequences remain. For evaluation, we adopt two synthetic benchmarks and one real-world dataset. ActionBench (Sabathier et al., 2026) covers diverse object categories and rich motions and is therefore well suited for assessing registration in general settings. In addition, we evaluate on DeformingThings4D (DT4D) (Li et al., 2021), a classic benchmark for non-rigid registration and shape correspondence. Although DT4D is limited to animal and humanoid characters, it provides ground-truth mesh sequences with fixed connectivity, which enables additional geodesic-error evaluation (Kim et al., 2011). We also evaluate on 4D-DRESS (Wang et al., 2024), real scans of clothed humans in motion, to test transfer to real captures. Appendix E details all three.

Evaluation metrics. We evaluate along two complementary axes: *correspondence accuracy* and *rendering-based alignment*. For the former, we follow the ActionBench protocol (Sabathier et al., 2026) and report *CD-Motion*, which measures how consistently matched surface points move over

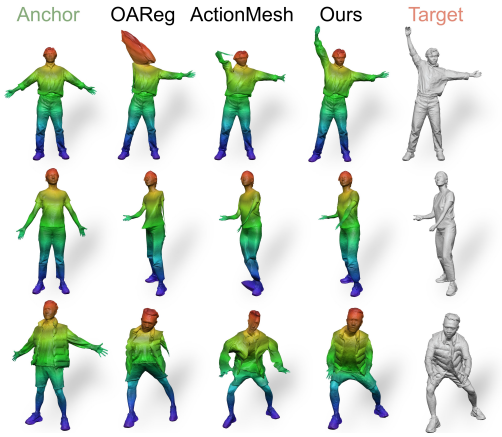

Figure 7: **Visual comparison on real 4D-DRESS scans.** Columns: anchor, OAReg, ActionMesh, MeshLoom, target.

Table 2: **Quantitative results on real 4D-DRESS scans** (100 takes, 32 subjects).

Method	Chamfer ↓	CLIP ↑	DreamSim ↓
OAReg	0.0152	0.9390	0.0705
ActionMesh	0.0152	0.9358	0.0640
Ours (w/o img)	**0.0109**	**0.9570**	**0.0525**

Table 3: **Ablation studies** (protocol of Sec. 4.2). Each row alters one component of the full model.

Variant	CD-M ↓	CLIP ↑	DreamSim ↓
Full model	0.1539	**0.9753**	**0.0231**
Per-frame encoder input	0.1523	0.9747	0.0239
w/o shape-latent prior	0.2518	0.9542	0.0577
w/o image features	**0.1522**	0.9740	0.0248
w/o edge & Jac. losses	0.1556	0.9724	0.0274

time, and on DT4D, additionally, the standard *normalized geodesic error* (NGE) (Kim et al., 2011), which requires the fixed-connectivity ground truth that only DT4D provides. For the latter, we render normal maps of the target and registered meshes from multiple viewpoints and compare them with *CLIP Similarity* (Radford et al., 2021) and *DreamSim* (Fu et al., 2023). Appendix F defines all metrics, and Appendix G gives additional metrics and analyses.

4.1 Comparisons

We compare against state-of-the-art methods: TransMatch (Trappolini et al., 2021), NDP and its learned-correspondence variant LNDP (Li & Harada, 2022), DPF (Prokudin et al., 2023), ClusterReg (Zhao et al., 2024), OAReg (Zhao et al., 2025a), and ActionMesh (Sabathier et al., 2026).

Quantitative comparisons. We evaluate with three input lengths (2, 4, and 8 frames) and report the averaged performance on both benchmarks in Table 1. As shown in Table 1, our method achieves the best scores on each metric on both benchmarks. On CD-Motion, which measures the temporal consistency of per-vertex deformation trajectories, we improve on the second-best method by 28% on DT4D, and on NGE, which directly measures vertex-level correspondence accuracy, by 26%. Our method is also top-ranked on the visual metrics, exceeding the best baseline by over 10% on DreamSim on ActionBench, reflecting closer agreement with the targets, whereas several baselines exhibit vertex-entanglement artifacts (Fig. 6). It delivers all of these gains at the fastest inference speed, roughly 50% faster than the closest baseline. Per-setting numbers, further analyses, and comparisons with more methods are given in Appendix G.

Qualitative comparisons. Fig. 6 presents qualitative comparisons; more examples on both benchmarks are given in Appendix K. Across diverse motions and categories, our method registers the anchor to the target with noticeably fewer artifacts than the baselines.

Results on real captured data. Both benchmarks above consist of synthetic or generated animations. To verify that MeshLoom transfers to real captured data, we further evaluate on 4D-DRESS (Wang et al., 2024), real multi-view scans of humans in everyday clothing, using 100 takes sampled from its 32 subjects. Since the ground truth provides no cross-frame correspondence, CD-Motion and NGE do not apply. We therefore report the Chamfer distance to the full-resolution scans together with the rendering-based metrics, and compare against two strong baselines of Table 1. As shown in Table 2, our method leads on all three metrics, improving the second-best method by 28% on Chamfer distance and 18% on DreamSim, so the gains observed on the two benchmarks carry over to real captures. Fig. 7 shows the same trend qualitatively. Although fine structures such as fingers, often partially fused in the raw scans, are not recovered perfectly, our registration closely follows the target pose and garment shape, whereas both baselines leave visible misalignment.

4.2 Ablation Studies

We evaluate the effectiveness of our key design choices. All variants are trained on TexVerse for 100K iterations and evaluated on 4-frame ActionBench sequences.

Encoder Input & Modality. As described in Sec. 3.2, our encoder replicates the anchor-mesh representation across all T time steps as its input. Table 3 compares this design against feeding a separately computed topology-aware representation for every frame. The per-frame variant yields no consistent gain across metrics yet inflates peak training memory from 23.5 to 37.5 GB, so the shared-anchor design is a favorable accuracy–efficiency trade-off.

We further ablate the two complementary modalities ingested by the encoder. As shown in Table 3, removing the 3D shape-latent input markedly degrades all metrics, showing that the encoder relies on this per-frame geometric information. Image features add a smaller gain, mainly on the rendering-based metrics, by sharpening the encoder's semantic understanding of the sequence.

Loss Functions. Table 3 confirms the benefit of the edge-vector and Jacobian losses, which yield consistent gains across all metrics on top of the per-vertex loss. As Fig. 5 illustrates, both losses are defined over the anchor-mesh connectivity graph and thus enjoy a larger receptive field than per-vertex supervision, suppressing geometric-collapse artifacts and preserving local structural fidelity.

Point Representation & Global-Translation Prediction. Fig. 3 shows that the topology-aware representation prevents vertex entanglement by correctly disambiguating Euclidean-close yet geodesically distant regions. As few ActionBench sequences exhibit such entanglement, we additionally construct 50 near-contact sequences, where the representation reduces the vertex error in these regions by 42% (Appendix H). As for the global-translation branch, Fig. 4 shows that it stabilizes the output under large translations and yields noticeably smoother surfaces.

4.3 Additional Results

Registration across Geometric Variations. Fig. 8(a) shows that MeshLoom generalizes across geometric variations, *e.g.*, species, and conforms to the inputs.

Motion Interpolation and Mesh Morphing. Since the decoder queries a global motion embedding (Sec. 3.3), it can be evaluated at arbitrary intermediate timestamps that the encoder has never observed. Fig. 8(b) shows motion-interpolation results between the anchor shape and multiple target motion states. The synthesized intermediate frames exhibit smooth temporal transitions while faithfully preserving object identity and local structure. Appendix I quantifies this on a held-out intermediate-frame benchmark.

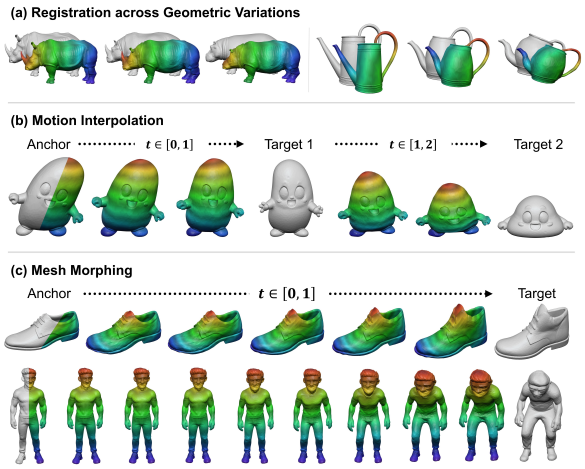

Figure 8: **Registration across geometric variations, motion interpolation, and mesh morphing.** Unregistered input meshes are rendered in pure white, while registered or interpolated outputs are rendered with colored coordinates. The anchor mesh is shown half white and half colored.

Fig. 8(c) further extends this capability to mesh morphing between two distinct objects, where our method produces smooth and coherent transitions from the anchor to the target. Our encoder thus learns continuous deformation trajectories from sparsely observed motion states rather than copying the input frames.

5 Conclusions

In this work, we present MeshLoom, a feed-forward non-rigid mesh registration framework that unifies heterogeneous mesh sequences into a topology-consistent sequence. Within a single model, MeshLoom resolves the key bottlenecks of existing registration methods: the network runs at high speed, supports diverse object categories, directly outputs per-vertex deformations, and processes variable-length sequences in a single pass. Extensive experiments show state-of-the-art performance across diverse object animations, and the same model further supports motion interpolation and mesh morphing. Although MeshLoom can be affected by the quality of the anchor mesh and can lose track under extreme motion (Appendix J), we believe it will benefit the many downstream tasks that require persistent mesh identity over time.

AI USE STATEMENT

We did *not* use generative AI tools for developing theoretical models or conceptual frameworks, proposing or refining hypotheses, or designing the research methodology. Formulating or proving mathematical claims, translation, and qualitative or thematic data analysis are not applicable to this work. Generative AI tools were used for four limited purposes. For the 50 near-contact sequences of Appendix H, they wrote the procedural scripts that generate and organize the data following the authors' specification, and helped identify metrics beyond NGE that quantify the effect of the topology-aware representation. They were further used for code completion while implementing the method, ranging from single lines to utility functions written from detailed specifications provided by the authors, where every completion was reviewed and tested by the authors before being kept, and for improving the readability of text that the authors had already drafted. The authors have reviewed all AI-assisted work and take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We have aimed to describe the method in enough detail to be reimplemented. Sec. 3 defines the topology-aware representation, the encoder, the decoder, and the training objectives, and Appendix B gives the remaining network details, including an efficient implementation of the topology-aware representation and the time and positional conditioning. The training losses and the training schemes are specified in Appendix C. Appendix D lists the main architectural and optimization hyperparameters, the pretrained models we build on, and the hardware and number of iterations used for training. Appendix E describes how the training corpus was filtered and how the three evaluation datasets were processed, and Appendix F gives the exact definition and computation of every metric we report. Per-setting results for all input lengths, together with the additional metrics, are reported in Appendix G, the further ablations in Appendix H, and the held-out interpolation protocol in Appendix I.

REFERENCES

Noam Aigerman, Kunal Gupta, Vladimir G Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. Neural jacobian fields: learning intrinsic mappings of arbitrary meshes. *ACM Transactions on Graphics (TOG)*, 41(4):1–17, 2022.

Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM transactions on graphics (TOG)*, 22(3):587–594, 2003.

Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. In *2007 IEEE conference on computer vision and pattern recognition*, pp. 1–8. IEEE, 2007.

Dragomir Anguelov, Praveen Srinivasan, Hoi-Cheung Pang, Daphne Koller, Sebastian Thrun, and James Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. *Advances in neural information processing systems*, 17, 2004.

Jan Bednarik, Vladimir G Kim, Siddhartha Chaudhuri, Shaifali Parashar, Mathieu Salzmann, Pascal Fua, and Noam Aigerman. Temporally-coherent surface reconstruction via metric-consistent atlases. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10438–10447. IEEE, 2021.

Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences*, 103(5):1168–1172, 2006.

Dongliang Cao, Marvin Eisenberger, Nafie El Amrani, Daniel Cremers, and Florian Bernard. Spectral meets spatial: Harmonising 3d shape matching and interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3658–3668, 2024.

Will Chang and Matthias Zwicker. Global registration of dynamic range scans for articulated model reconstruction. *ACM Transactions on Graphics (TOG)*, 30(3):1–15, 2011.

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

Luca Cosmo, Emanuele Rodola, Jonathan Masci, Andrea Torsello, and Michael M Bronstein. Matching deformable objects in clutter. In *2016 Fourth international conference on 3D vision (3DV)*, pp. 1–10. IEEE, 2016.

Bailin Deng, Yuxin Yao, Roberto M Dyke, and Juyong Zhang. A survey of non-rigid 3d registration. In *Computer Graphics Forum*, volume 41, pp. 559–589. Wiley Online Library, 2022.

Michal Edelstein, Danielle Ezuz, and Mirela Ben-Chen. Enigma: evolutionary non-isometric geometry matching. *ACM Trans. Graph.*, 39(4), August 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392447. URL https://doi.org/10.1145/3386569.3392447.

Marvin Eisenberger, Zorah Lahner, and Daniel Cremers. Smooth shells: Multi-scale shape registration with functional maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12265–12274, 2020.

Danielle Ezuz, Behrend Heeren, Omri Azencot, Martin Rumpf, and Mirela Ben-Chen. Elastic correspondence between triangle meshes. In *Computer Graphics Forum*, volume 38, pp. 121–134. Wiley Online Library, 2019.

Wanquan Feng, Hongrui Cai, Junhui Hou, Bailin Deng, and Juyong Zhang. Differentiable deformation graph-based neural non-rigid registration. *Communications in Mathematics and Statistics*, 11(1):151–167, 2023.

Stephanie Fu, Netanel Y Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. *Advances in Neural Information Processing Systems*, 36:2208, 2023.

Maolin Gao, Shao Jie Hu-Chen, Congyue Deng, Riccardo Marin, Leonidas Guibas, and Daniel Cremers. Rino: Rotation-invariant non-rigid correspondences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 34683–34693, 2026.

Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the european conference on computer vision (ECCV)*, pp. 230–246, 2018.

Osamu Hirose. A bayesian formulation of coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 43(7):2269–2286, 2020.

Osamu Hirose. Geodesic-based bayesian coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5816–5832, 2022.

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

Peng Huang, Chris Budd, and Adrian Hilton. Global temporal registration of multiple non-rigid surface sequences. In *CVPR 2011*, pp. 3473–3480. IEEE, 2011.

Bing Jian and Baba C Vemuri. A robust algorithm for point set registration using mixture of gaussians. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pp. 1246–1251. IEEE, 2005.

Puhua Jiang, Mingze Sun, and Ruqi Huang. Non-rigid shape registration via deep functional maps prior. *Advances in Neural Information Processing Systems*, 36:58409–58427, 2023.

Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. Blended intrinsic maps. *ACM transactions on graphics (TOG)*, 30(4):1–12, 2011.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Artiom Kovnatsky, Michael M Bronstein, Alexander M Bronstein, Klaus Glashoff, and Ron Kimmel. Coupled quasi-harmonic bases. In *Computer Graphics Forum*, volume 32, pp. 439–448. Wiley Online Library, 2013.

Hao Li, Bart Adams, Leonidas J Guibas, and Mark Pauly. Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics (ToG)*, 28(5):1–10, 2009.

Yang Li and Tatsuya Harada. Non-rigid point cloud registration with neural deformation pyramid. *Advances in Neural Information Processing Systems*, 35:27757–27768, 2022.

Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12706–12716, 2021.

Yaron Lipman and Thomas Funkhouser. Möbius voting for surface correspondence. *ACM Transactions on Graphics (ToG)*, 28(3):1–12, 2009.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Arman Maesumi, Tanish Makadia, Thibault Groueix, Vladimir Kim, Daniel Ritchie, and Noam Aigerman. Poissonnet: A local-global approach for learning on surfaces. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 2025.

Riccardo Marin, Simone Melzi, Emanuele Rodola, and Umberto Castellani. Farm: Functional automatic registration method for 3d human bodies. In *Computer Graphics Forum*, volume 39, pp. 160–173. Wiley Online Library, 2020.

Riccardo Marin, Enric Corona, and Gerard Pons-Moll. Nicp: neural icp for 3d human registration at scale. In *European Conference on Computer Vision*, pp. 265–285. Springer, 2024.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Andriy Myronenko, Xubo Song, and Miguel Carreira-Perpinan. Non-rigid point set registration: Coherent point drift. *Advances in neural information processing systems*, 19, 2006.

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

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Emery Pierson, Lei Li, Angela Dai, and Maks Ovsjanikov. Diffumatch: Category-agnostic spectral diffusion priors for robust non-rigid shape matching. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5745–5756. IEEE, 2025.

Sergey Prokudin, Qianli Ma, Maxime Raafat, Julien Valentin, and Siyu Tang. Dynamic point fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7964–7976, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Emanuele Rodola, Alex M Bronstein, Andrea Albarelli, Filippo Bergamasco, and Andrea Torsello. A game-theoretic approach to deformable shape matching. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 182–189. IEEE, 2012.

Raif M Rustamov, Maks Ovsjanikov, Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Leonidas Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013.

Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pp. 3212–3217. IEEE, 2009.

Remy Sabathier, David Novotny, Niloy J. Mitra, and Tom Monnier. Actionmesh: Animated 3d mesh generation with temporal 3d diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 34312–34321, June 2026.

Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (ToG)*, 41(3):1–16, 2022.

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

Justin Solomon, Andy Nguyen, Adrian Butscher, Mirela Ben-Chen, and Leonidas Guibas. Soft maps between surfaces. In *Computer Graphics Forum*, volume 31, pp. 1617–1626. Wiley Online Library, 2012.

Olga Sorkine, Marc Alexa, et al. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pp. 109–116, 2007.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004.

Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM siggraph 2007 papers*, pp. 80–es. 2007.

Mingze Sun, Shiwei Mao, Puhua Jiang, Maks Ovsjanikov, and Ruqi Huang. Spatially and spectrally consistent deep functional maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14497–14507, 2023.

Ramana Sundararaman, Nicolas Donati, Simone Melzi, Etienne Corman, and Maks Ovsjanikov. Deformation recovery: Localized learning for detail-preserving deformations. *ACM Transactions on Graphics (TOG)*, 43(6):1–16, 2024.

Ramana Subramanyam Sundararaman, Riccardo Marin, Emanuele Rodola, and Maks Ovsjanikov. Reduced representation of deformation fields for effective non-rigid shape matching. *Advances in Neural Information Processing Systems*, 35:10405–10420, 2022.

Giovanni Trappolini, Luca Cosmo, Luca Moschella, Riccardo Marin, Simone Melzi, and Emanuele Rodolà. Shape registration in the time of transformers. *Advances in Neural Information Processing Systems*, 34:5731–5744, 2021.

Wenbo Wang, Hsuan-I Ho, Chen Guo, Boxiang Rong, Artur Grigorev, Jie Song, Juan Jose Zarate, and Otmar Hilliges. 4d-dress: A 4d dataset of real-world human clothing with semantic annotations. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 550–560. IEEE, 2024.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, et al. Native and compact structured latents for 3d generation. *arXiv preprint arXiv:2512.14692*, 2025a.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 21469–21480, 2025b.

Yuxin Yao, Bailin Deng, Weiwei Xu, and Juyong Zhang. Fast and robust non-rigid registration using accelerated majorization-minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):9681–9698, 2023.

Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions On Graphics (TOG)*, 42(4):1–16, 2023.

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

Yibo Zhang, Li Zhang, Rui Ma, and Nan Cao. Texverse: A universe of 3d objects with high-resolution textures. *arXiv preprint arXiv:2508.10868*, 2025.

Mingyang Zhao, Jingen Jiang, Lei Ma, Shiqing Xin, Gaofeng Meng, and Dong-Ming Yan. Correspondence-free non-rigid point set registration using unsupervised clustering analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 21199–21208, 2024.

Mingyang Zhao, Gaofeng Meng, and Dong-ming Yan. Occlusion-aware non-rigid point cloud registration via unsupervised neural deformation correntropy. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025b.

A Overview

The appendix is organized as follows. Appendix B provides additional network-design details, including an efficient implementation of the topology-aware representation, the time and 3D positional conditioning used in the encoder, and the positional encoding used in the decoder. Appendix C elaborates on the training objectives and introduces our training schemes, and Appendix D lists the implementation details. Appendix E describes the processing of the training and evaluation datasets, and Appendix F specifies how the evaluation metrics are computed. To complement Table 1 of the main paper, Appendix G reports per-setting quantitative results on both benchmarks together with additional metrics and analyses. Appendix H presents additional ablation studies, Appendix I quantitatively evaluates motion interpolation on held-out frames, and Appendix J discusses limitations and future work. Finally, Appendix K collects additional qualitative results.

B Additional Network-Design Details

B.1 Topology-Aware Point Representation

As shown in Eq. (1) of the main paper, we enlarge the receptive field of each per-vertex feature by applying the adjacency operator p times, yielding $\mathbf{A}^p$.

Although $\mathbf{A}$ itself is stored as a sparse matrix, naively materializing $\mathbf{A}^p$ via p sparse–sparse multiplications (*i.e.*, repeatedly multiplying $\mathbf{A}$ by itself) produces an increasingly dense matrix and incurs memory cost quadratic in N for large anchor meshes. We instead compute the propagated feature $\mathbf{A}^p\mathbf{H}^{(\ell)}$ through p successive sparse–dense products (*i.e.*, multiplying the sparse matrix $\mathbf{A}$ with the dense matrix $\mathbf{H}^{(\ell)}$ at each step),

$$\mathbf{A}^p\mathbf{H}^{(\ell)} \;=\; \mathbf{A}\big(\mathbf{A}(\cdots(\mathbf{A}\,\mathbf{H}^{(\ell)}))\big), \tag{9}$$

which is mathematically equivalent yet keeps peak memory linear in the number of mesh edges.

B.2 Motion Encoder

Time conditioning. To make each token aware of the frame it belongs to, we inject the corresponding timestamp into the token features before every attention layer of every transformer block (the two cross-attention and two self-attention layers described in Sec. 3.2 of the main paper). Specifically, we encode each timestamp t with a sinusoidal embedding and route it through a FiLM (Perez et al., 2018) modulator. The modulator produces frame-specific scale and shift parameters (α_t, β_t) that modulate token activations before every attention layer.

Sparse-tensor structure and 3D positional encoding. TRELLIS.2 (Xiang et al., 2025a) stores its 3D shape latents in a sparse-tensor format, in which each token carries both a 3D coordinate and a feature vector. To remain natively compatible with this representation, we organize the encoder input in the same sparse-tensor format, where each point sampled from the topology-aware representation of the anchor mesh is stored together with its 3D coordinate, so that the encoder tokens themselves become sparse-tensor tokens. This allows the cross-attention with the TRELLIS.2 shape latents and the inter-/intra-frame self-attention to operate directly on these variable-length token sets.

To disambiguate sparse-tensor tokens that share similar features but reside at different 3D positions, we apply 3D rotary positional embedding (RoPE) (Su et al., 2024) during all sparse-attention computations. RoPE additionally injects relative spatial-distance information between tokens, helping the encoder reason about local geometry.

For the cross-attention with image features, we directly attend between the sparse-tensor tokens and the image features without any additional positional embedding.

B.3 Deformation Decoder

Since the global motion embedding $\mathbf{Z}$ produced by the encoder also consists of sparse-tensor tokens, a natural choice would be to reuse 3D RoPE when these tokens serve as keys and values for the decoder cross-attention (Eq. (4) of the main paper) with the queries in the local-deformation branch.

However, we empirically find that using explicit 3D Fourier positional embeddings (Mildenhall et al., 2021) of the associated 3D coordinates yields better performance. We attribute this to the

fact that RoPE encodes only *relative* spatial relations between tokens, whereas Fourier embeddings preserve *absolute* position information, which is more informative for deformation reconstruction in which absolute coordinates matter. We therefore augment the key/value tokens of the local-deformation branch as follows:

$$\widetilde{\mathbf{Z}} = \big[\, \mathbf{Z} \,\big\|\, \gamma(\mathbf{p}_{\mathbf{Z}} - \bar{\mathbf{v}}^a) \big], \tag{10}$$

where $\mathbf{p}_{\mathbf{Z}}$ denotes the 3D coordinate associated with each token in $\mathbf{Z}$, and $\gamma(\cdot) : \mathbb{R}^3 \to \mathbb{R}^{D_\gamma}$ is a Fourier embedding that lifts a 3D point into a multi-frequency sinusoidal basis of dimension D_γ. Since our decoder decouples global and local transformations into two separate branches (Sec. 3.3 of the main paper), we center all coordinates by the anchor centroid $\bar{\mathbf{v}}^a$ before applying γ, which removes the anchor's global position from the positional embedding and helps the local branch focus on deformation rather than global translation.

C Additional Training Details

C.1 Training Objectives

Global and local deformation loss. We provide explicit definitions for the ground-truth supervision targets $\Delta\mathbf{c}^{*(t)}$ and $\Delta\mathbf{r}_i^{*(t)}$ used by the global- and local-deformation losses,

$$\Delta\mathbf{c}^{*(t)} = \bar{\mathbf{v}}^{*(t)} - \bar{\mathbf{v}}^a, \qquad \Delta\mathbf{r}_i^{*(t)} = \big(\mathbf{v}_i^{*(t)} - \bar{\mathbf{v}}^{*(t)}\big) - \big(\mathbf{v}_i^a - \bar{\mathbf{v}}^a\big), \tag{11}$$

where $\mathbf{v}_i^a$ and $\mathbf{v}_i^{*(t)}$ are the positions of anchor vertex i in the anchor frame and at the ground-truth target frame t, respectively, and $\bar{\mathbf{v}}^a$ and $\bar{\mathbf{v}}^{*(t)}$ are the corresponding full-mesh centroids. Subtracting both centroids supervises the local residual in a centroid-removed coordinate frame, so that it is fully decoupled from the global shift.

Edge and Jacobian losses. The edge-vector loss $\mathcal{L}_{\text{edge}}$ and the Jacobian loss $\mathcal{L}_{\text{jac}}$ defined in Sec. 3.4 of the main paper rely on a connectivity graph built over a subset of anchor vertices. To balance supervision density and computational cost, we start from the decoder query points and run a multi-hop breadth-first search (BFS) along the anchor-mesh edges to collect topologically linked vertices, yielding an expanded set $\mathcal{V}'$ of $N' = N_{\text{q}} + N_{\text{extra}}$ points. On this expanded set we build a k-nearest face-adjacent neighbor graph $\mathcal{G} = (\mathcal{V}', \mathcal{E})$ in which every edge corresponds to an actual anchor-mesh edge, so that no spurious long-range shortcuts are introduced. Both $\mathcal{L}_{\text{edge}}$ and $\mathcal{L}_{\text{jac}}$ are evaluated on this graph.

For each vertex i in the expanded set, the per-vertex Jacobian $\mathbf{J}_i$ defined in Sec. 3.4 is estimated by Tikhonov-regularized least squares over its k face-adjacent neighbors,

$$\mathbf{J}_i = \underset{\mathbf{J}\in\mathbb{R}^{3\times 3}}{\arg\min} \sum_{j\in\mathcal{N}(i)} \big\|(\widetilde{\mathbf{v}}_j^{(t)} - \widetilde{\mathbf{v}}_i^{(t)}) - \mathbf{J}(\mathbf{v}_j^a - \mathbf{v}_i^a)\big\|_2^2 + \lambda_{\text{reg}}\|\mathbf{J}\|_F^2, \tag{12}$$

where $\mathcal{N}(i)$ denotes the k face-adjacent neighbors of vertex i in $\mathcal{G}$, and $\lambda_{\text{reg}}{=}10^{-4}$ is a small regularization weight that stabilizes the linear system in nearly degenerate, *e.g.*, locally planar, neighborhoods. The ground-truth Jacobian $\mathbf{J}_i^*$ is obtained analogously from the ground-truth deformation.

C.2 Training Schemes

Beyond the architectural designs in the main paper, we adopt three training schemes that further enhance the model: an *intermediate-frame supervision* scheme that equips the decoder with motion-interpolation ability at unseen timestamps; a *dynamic temporal-budget* sampler that exposes the model to variable-length contexts so that the trained network generalizes across input lengths at inference; and a *random image-dropout* scheme that randomly replaces the per-frame image features with an all-zero placeholder, allowing the network to gracefully handle inputs without accompanying images. The first two schemes are illustrated in Fig. 9.

Intermediate-frame supervision. A subtle but important point in our training scheme is the distinction between the frames fed to the encoder and the timestamps at which the decoder is supervised. Each training sequence covers L_{seq} mesh frames whose frame indices are normalized to timestamps in $[0, 1]$. From this pool, we draw two disjoint subsets at every iteration. A set of T *main* frames is passed to the motion encoder to construct $\mathbf{Z}$. An additional set of T' *auxiliary* frames is drawn from the remaining timestamps and is used *only* for decoder supervision. The encoder, therefore, sees only a sparse subset of keyframes, yet the decoder is asked to reconstruct the

full deformation at every main and auxiliary timestamp, including timestamps that the encoder has never observed. This explicitly trains the decoder to *interpolate* the motion embedding across time, and allows the trained model to synthesize plausible mesh deformations at arbitrary intermediate timestamps at inference, supporting motion interpolation and mesh morphing.

Dynamic temporal-budget training. Fixing the input length T during training would induce a train–test distribution mismatch, since at inference, the user may supply sequences of different lengths. We therefore adopt a *dynamic-budget* strategy that randomizes T per iteration while keeping the number of tokens per GPU roughly constant. We fix a per-GPU token budget $\mathcal{B} = B \cdot T$ (batch size $\times$ temporal frames) and, at each step, sample

$$T \sim \mathrm{Uniform}[T_{\min}, T_{\max}], \quad B = \lfloor \mathcal{B}/T \rfloor, \tag{13}$$

so that a smaller T yields a proportionally larger effective batch size, and vice versa. When $T < T_{\max}$, we temporally subsample the main frames while always retaining the first and last frames to preserve the full temporal span. The random seed governing T is synchronized across all distributed processes to ensure identical tensor shapes across ranks. Combined with the intermediate-frame supervision above, this strategy exposes the model to temporal contexts ranging from two-frame pairs to long sequences, enabling the network to generalize to variable-length inputs at inference without any architectural change.

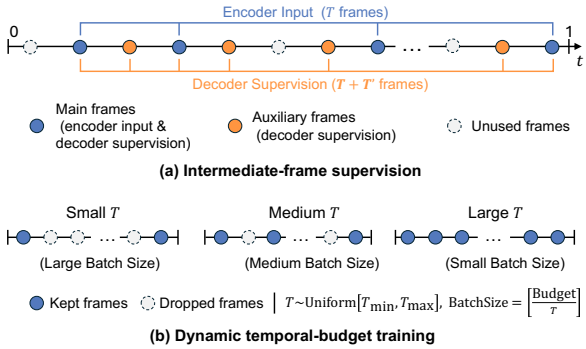

Figure 9: **Two of our training schemes.** Illustration of (a) intermediate-frame supervision, in which the decoder is supervised at both main and auxiliary timestamps while the encoder observes only the main subset, and (b) dynamic temporal-budget training, in which the temporal length T and the batch size B are jointly resampled per iteration so that the per-GPU token count $\mathcal{B} = B \cdot T$ remains roughly constant.

Random image dropout. In real-world deployments, the input mesh sequence may not always be accompanied by RGB images (although such images can be readily obtained from off-the-shelf image-generation models). To make our network robust to this scenario, we train it to handle both the with-image and without-image input modes. Inspired by classifier-free guidance (Ho & Salimans, 2022) in image-generation models, at each training iteration, we replace the per-frame image features $\mathbf{I}_t$ with an all-zero placeholder with probability 0.5. This allows the network to fall back gracefully to a vision-free mode when reference images are unavailable at inference.

D Implementation Details

The topology-aware point representation uses $L{=}4$ residual GCN-style layers, each aggregating neighbor information through a $p{=}4$-hop adjacency, which together provide a sufficiently large receptive field over the anchor mesh. For the encoder input, we FPS-sample $N_{\mathrm{ds}}{=}2{,}048$ points from the anchor-mesh representation $\mathbf{H}_a$. We adopt the recent TRELLIS.2 (Xiang et al., 2025a) 3D foundation model to obtain the shape-latent prior of each mesh frame, and DINOv3 (Siméoni et al., 2025) to extract image features. The encoder consists of 5 transformer blocks. To bound memory usage when processing meshes with very large vertex counts, at each training iteration we sample $N_{\mathrm{q}}{=}4{,}096$ anchor vertices per mesh frame (half via FPS and half via uniform random sampling) as decoder query points for vertex-level supervision. For the geometry graph used by the edge-vector and Jacobian losses (Appendix C), we expand this query set via a 2-hop BFS along anchor-mesh edges (adding up to $N_{\mathrm{extra}}{=}4{,}096$ extra vertices) and build a $k{=}8$ face-adjacent nearest-neighbor graph on the expanded set. The dynamic-budget sampler uses $\mathcal{B}{=}16$ with $T_{\min}{=}2$ and $T_{\max}{=}16$ encoder frames per iteration, yielding effective per-GPU batch sizes between 1 and 8. For intermediate-frame supervision, we draw the same number of auxiliary frames from the remaining timestamps to provide additional decoder supervision.

We optimize with AdamW (Loshchilov & Hutter, 2019) at a learning rate of $4{\times}10^{-4}$. The loss weights are set to $\lambda_{\mathrm{global}}{=}\lambda_{\mathrm{local}}{=}1$, $\lambda_{\mathrm{edge}}{=}0.5$, and $\lambda_{\mathrm{jac}}{=}0.05$. Training is conducted on 32 NVIDIA A100 GPUs and converges after 200K iterations.

E Additional Dataset Details

Training dataset. We use the animation subset of TexVerse (Zhang et al., 2025) as our training data. For each animated sequence, we extract a contiguous window of L_{seq}=32 frames. The per-frame structured shape latent $\mathbf{S}_t$ is produced by the TRELLIS.2 (Xiang et al., 2025a) data-processing pipeline, which voxelizes each mesh into an O-Voxel representation and encodes it into a compact sparse latent grid (sparse-tensor format). The per-frame image-conditioning input $\mathbf{I}_t$ is an RGB rendering from a fixed front view. We apply four quality filters to ensure training stability: (1) meshes with more than 150K vertices are discarded to bound memory; (2) static sequences (whose vertices remain unchanged across the sampled frames) are removed, as they provide no deformation signal; (3) shape latents exceeding 30K voxels are excluded; and (4) samples with missing files, corrupted meshes, or NaN vertex values are dropped. About 30K sequences remain for training.

Evaluation datasets. We evaluate on ActionBench (Sabathier et al., 2026), DeformingThings4D (DT4D) (Li et al., 2021), and 4D-DRESS (Wang et al., 2024). ActionBench contains 128 animation sequences spanning a broad range of object categories. For each sequence, the benchmark provides per-frame guidance images and ground-truth point clouds whose points correspond consistently across frames. Since the benchmark does not directly provide mesh sequences, we use the image-to-mesh network of ActionMesh (Sabathier et al., 2026) to generate the input mesh sequence to be registered. For the experiments, we linearly sample frames at different temporal intervals to construct variable-input settings: 2 frames with a large time gap, 4 frames with a moderate time gap, and 8 frames with denser temporal coverage.

DT4D contains 1,972 animation sequences spanning 56 animal identities and 54 humanoid identities. We randomly select one sequence per humanoid identity and two per animal identity (or the single available sequence when only one exists), yielding 158 evaluation sequences in total. Unlike ActionBench, DT4D provides ground-truth mesh sequences, from which we construct the registration inputs directly. We keep the first frame as the anchor and re-mesh each remaining frame at a different resolution, so that vertex count and connectivity vary across the sequence as in a practical registration setting; all methods receive the same inputs. Because the anchor retains the ground-truth connectivity and every registered output inherits the anchor topology, the outputs can be compared with the ground truth vertex by vertex, which is what enables the normalized geodesic error (Appendix F). Since most DT4D assets do not provide textures, we run our model in the image-free mode on this benchmark (see the random image dropout in Appendix C). Additionally, following the ActionBench protocol, we sample 100,000 surface points per mesh in synchronized barycentric coordinates so that the points correspond across frames, and use the resulting point-cloud sequences for the geometric metrics (CD-3D, CD-4D, and CD-Motion) described in Appendix F. Frames are sampled with the same 2-, 4-, and 8-frame settings as on ActionBench, and all registered outputs are expressed in the coordinate frame of the ground-truth sequences before evaluation.

4D-DRESS (Wang et al., 2024) contains real multi-view scans of 32 subjects in everyday clothing. We randomly sample 100 takes, and within each take sample frames at a fixed temporal stride so that consecutive inputs exhibit substantial motion, from which the 4-frame setting is drawn as on the other benchmarks. Since the frames are independent scans, the input sequences are natively heterogeneous in vertex count and connectivity, and no remeshing is applied. All methods receive the same input meshes, while the metrics are computed against the full-resolution scans; as no cross-frame correspondence exists, only the Chamfer distance and the rendering-based metrics are reported. As with DT4D, we run in the image-free mode.

F Evaluation Metric Details

For the visual metrics (LPIPS, CLIP Similarity, and DreamSim), we render normal maps of each registered frame and of its reference surface from four azimuth angles (0°, 90°, 180°, and 270°). The reference surface is the ground-truth mesh on DT4D and the full-resolution scan on 4D-DRESS. Since ActionBench provides no ground-truth meshes, we use the generated input mesh of each frame (Appendix E) as the reference, so on ActionBench these scores measure agreement with the generated target meshes. We first average the metric values across these four views, and then average over all frames in the sequence to obtain the final score.

For the geometric metrics (CD-3D, CD-4D, and CD-Motion) reported in the main paper and the appendix, we follow the ActionBench protocol (Sabathier et al., 2026). We first sample dense point

clouds from the predicted mesh sequences and align them with the ground-truth point clouds using Iterative Closest Point (ICP), which jointly optimizes rotation, translation, and scale by gradient descent on the Chamfer distance. ICP is performed on 10K-point subsamples, while the symmetric Chamfer distance is computed with unsquared Euclidean distances on denser 100K-point samples. All distances are measured in the ground-truth coordinates, in which each sequence is normalized as a whole to $[-1, 1]^3$ (ActionBench and DT4D) or $[-0.5, 0.5]^3$ (4D-DRESS).

CD-3D. We estimate an independent per-frame ICP alignment between the predicted and ground-truth point clouds at every frame. CD-3D is then the Chamfer distance between the per-frame-aligned prediction and the ground truth, averaged across all frames. As the alignment is recomputed at every frame, CD-3D factors out global pose drift and isolates per-frame shape accuracy.

CD-4D. We estimate a single ICP alignment from the first frame and apply it uniformly to every subsequent predicted frame. CD-4D is the Chamfer distance after this unified alignment, averaged across all frames. Holding the alignment fixed over time penalizes both shape error and accumulated temporal pose drift, jointly reflecting shape fidelity and temporal pose consistency.

CD-Motion. CD-Motion further probes motion coherence by *synchronizing* the surface sampling across frames. We compute face indices and barycentric weights once on the first predicted frame and reuse them on every subsequent frame, so that each sampled point traces a temporal trajectory on the surface. After applying the unified ICP transform, we establish bidirectional nearest-neighbor correspondences between the predicted and ground-truth point clouds at the first frame, propagate these correspondences across the entire sequence, and report the symmetric mean per-trajectory Euclidean distance. This penalizes temporally inconsistent surface motion even when per-frame shapes are individually accurate.

Normalized geodesic error (NGE). NGE follows the standard protocol of Blended Intrinsic Maps (Kim et al., 2011): the error of a correspondence is the geodesic distance on the target surface between the predicted and the ground-truth corresponding points, normalized by the square root of the target surface area. On DT4D, the registered meshes share the ground-truth connectivity, so vertex i of a registered frame should coincide with vertex i of the corresponding ground-truth frame. After the first-frame ICP alignment used for CD-Motion, we snap each registered vertex to its nearest ground-truth vertex (the standard nearest-neighbor conversion of a deformed shape into point-to-point correspondences (Groueix et al., 2018; Eisenberger et al., 2020)), measure the geodesic distance from there to its ground-truth counterpart as the shortest path along the mesh edges (Dijkstra's algorithm), normalize it by the square root of the ground-truth surface area, and average over all vertices and frames. NGE is not reported on ActionBench, whose ground truth consists of point-cloud trajectories without meshes.

G Additional Quantitative Results

Tables 4 and 5 complement Table 1 of the main paper with the per-setting results of every method on ActionBench and DT4D under the 2-, 4-, and 8-frame input settings. For completeness, we also include the other metrics of the ActionBench protocol, CD-3D and CD-4D, together with LPIPS (Zhang et al., 2018). Table 6 summarizes these three metrics averaged over the input settings. CD-3D and CD-4D are the Chamfer distances between the registered and ground-truth surfaces after per-frame and first-frame ICP alignment, respectively (Appendix F). We refer to them as surface-fit metrics, since they only quantify how closely the output surface fits the target and are blind to vertices sliding along that surface. Registration, however, is about where each vertex lands on the target, so a method can score well on CD-3D and CD-4D while its correspondences are wrong. CD-Motion and NGE measure correspondence directly and are therefore the metrics that the main paper focuses on, whereas CD-3D and CD-4D are included here only for completeness of the ActionBench protocol.

On the surface-fit metrics, DPF (Prokudin et al., 2023) attains the lowest values on both benchmarks. This is expected, since DPF fits each target by minimizing the Chamfer distance at inference, with only an as-isometric-as-possible regularizer added, so that metric and objective nearly coincide; NDP (Li & Harada, 2022), which also minimizes the Chamfer distance, is likewise strong on these metrics. Our feed-forward method, which never optimizes against the target at inference, is nonetheless second-best on CD-4D on ActionBench and on CD-3D on DT4D. Taken together

Table 4: **Detailed performance comparison on ActionBench.** Rows are grouped into non-neural methods (top; see the discussion at the end of Appendix G), neural baselines (middle), and our method (bottom). We report both geometric metrics (CD-3D, CD-4D, CD-Motion) and rendering-based visual metrics (CLIP, LPIPS, DreamSim) under each of the 2-, 4-, and 8-frame input settings. We additionally report inference time, measured on 8-frame inputs with a fixed 150K vertices and 300K faces per mesh frame. Best results are in **bold**, second-best are underlined.

Method	CD-3D ↓			CD-4D ↓			CD-M ↓			CLIP ↑			LPIPS ↓			DreamSim ↓			Infer.
	2	4	8	2	4	8	2	4	8	2	4	8	2	4	8	2	4	8	Time
Amberg NRICP (Amberg et al., 2007)	0.0633	0.0674	0.0691	0.0792	0.0910	0.0970	0.1484	0.1859	0.2043	0.9463	0.9312	0.9245	0.1031	0.1111	0.1147	0.0574	0.0715	0.0782	20.6
Sumner NRICP (Sumner et al., 2007)	0.0621	0.0659	0.0675	0.0793	0.0915	0.0975	0.1501	0.1893	0.2078	0.9389	0.9202	0.9126	0.1053	0.1146	0.1187	0.0607	0.0769	0.0837	31.6
FPFH (Rusu et al., 2009) + Amberg NRICP	0.0631	0.0660	0.0680	0.0783	0.0886	0.0948	0.1653	0.2061	0.2273	0.9340	0.9153	0.9057	0.1101	0.1222	0.1277	0.0697	0.0866	0.0946	22.4
BCPD (Hirose, 2020)	0.0554	0.0575	0.0581	0.0774	0.0867	0.0933	0.1538	0.1915	0.2134	0.9512	0.9361	0.9304	0.0883	0.0988	0.1035	0.0428	0.0550	0.0599	18.5
GBCPD (Hirose, 2022)	0.0561	0.0587	0.0598	0.0791	0.0916	0.0982	0.1547	0.1986	0.2192	0.9464	0.9314	0.9248	0.0941	0.1047	0.1096	0.0501	0.0619	0.0672	9.3
AMM (Yao et al., 2023)	0.0562	0.0582	0.0594	0.0776	0.0849	0.0900	0.1349	0.1636	0.1790	0.9672	0.9561	0.9517	0.0755	0.0839	0.0875	0.0320	0.0411	0.0446	11.5
ClusterReg (Zhao et al., 2024)	0.0541	0.0546	0.0548	0.0781	0.0844	0.0880	0.1572	0.1978	0.2155	0.9085	0.8688	0.8484	0.0998	0.1156	0.1237	0.0788	0.1097	0.1250	42.6
TransMatch (Trappolini et al., 2021)	0.0843	0.1000	0.1081	0.1410	0.1828	0.2047	0.6101	0.8939	1.0332	0.8283	0.7474	0.7072	0.1448	0.1715	0.1846	0.1785	0.2589	0.2991	14.2
NDP (Li & Harada, 2022)	0.0541	0.0546	0.0548	0.0799	0.0846	0.0885	0.1438	0.1761	0.1898	0.9736	0.9652	0.9610	0.0561	0.0638	0.0671	0.0265	0.0339	0.0370	5.9
LNDP (Li & Harada, 2022)	0.0610	**0.0487**	0.0658	0.0873	**0.0646**	0.1037	0.1450	0.1912	0.1917	0.9678	0.9584	0.9553	0.0983	0.1049	0.1078	0.0409	0.0500	0.0531	279.9
DPF (Prokudin et al., 2023)	**0.0530**	0.0531	0.0533	**0.0758**	0.0832	**0.0870**	0.1479	0.1831	0.2001	0.9785	0.9701	0.9659	0.0297	**0.0385**	**0.0426**	0.0157	0.0218	0.0248	70.2
OAReg (Zhao et al., 2025a)	0.0548	0.0565	0.0571	0.0784	0.0858	0.0899	0.1345	0.1649	0.1782	0.9748	0.9661	0.9617	0.0580	0.0655	0.0688	0.0263	0.0352	0.0387	25.3
ActionMesh (Sabathier et al., 2026)	0.0577	0.0570	**0.0532**	0.0785	0.0869	0.0873	0.1325	0.1615	0.1731	0.9714	0.9669	0.9738	0.0958	0.0951	0.0553	0.0319	0.0345	**0.0196**	7.7
Ours (w/o img)	0.0566	0.0571	0.0577	0.0769	0.0853	0.0897	0.1269	0.1545	0.1666	0.9846	0.9776	0.9737	0.0267	0.0440	0.0518	0.0141	0.0208	0.0242	**3.1**
Ours	0.0562	0.0567	0.0572	0.0765	0.0850	0.0888	**0.1245**	**0.1524**	**0.1642**	**0.9855**	**0.9781**	**0.9744**	**0.0254**	0.0435	0.0507	**0.0126**	**0.0197**	0.0230	

Table 5: **Detailed performance comparison on DeformingThings4D.** Rows are grouped into non-neural methods (top; see the discussion at the end of Appendix G), neural baselines (middle), and our method (bottom). We report the geometric metrics (CD-3D, CD-4D, CD-Motion), the rendering-based visual metrics (CLIP, LPIPS, DreamSim), and the normalized geodesic error (NGE) under each of the 2-, 4-, and 8-frame input settings. Since most DT4D assets are untextured, our results are obtained in the image-free mode (w/o img). Best results are in **bold**, second-best are underlined.

Method	CD-3D ↓			CD-4D ↓			CD-M ↓			CLIP ↑			LPIPS ↓			DreamSim ↓			NGE ↓		
	2	4	8	2	4	8	2	4	8	2	4	8	2	4	8	2	4	8	2	4	8
Amberg NRICP (Amberg et al., 2007)	0.0534	0.0726	0.0808	0.2025	0.2854	0.3305	0.3978	0.5560	0.6270	0.8607	0.8318	0.8207	0.1442	0.1376	0.1344	0.1649	0.1938	0.2042	0.1171	0.1636	0.1821
Sumner NRICP (Sumner et al., 2007)	0.0488	0.0678	0.0741	0.4074	0.6103	0.7014	0.4290	0.6387	0.7379	0.8623	0.8340	0.8230	0.1414	0.1404	0.1401	0.1580	0.1839	0.1931	0.1157	0.1618	0.1804
FPFH (Rusu et al., 2009) + Amberg NRICP	0.0383	0.0539	0.0583	0.0883	0.1387	0.1432	0.2469	0.3604	0.3843	0.9017	0.8666	0.8548	0.0885	0.0977	0.1009	0.1034	0.1404	0.1507	0.0836	0.1202	0.1313
BCPD (Hirose, 2020)	0.0331	0.0420	0.0453	0.0552	0.0734	0.0758	0.2786	0.3820	0.4104	0.8951	0.8670	0.8563	0.1038	0.1096	0.1117	0.1106	0.1337	0.1420	0.0956	0.1324	0.1437
GBCPD (Hirose, 2022)	0.0335	0.0444	0.0477	0.0580	0.0723	0.0797	0.2860	0.3803	0.4219	0.9021	0.8692	0.8570	0.1115	0.1177	0.1206	0.1029	0.1328	0.1428	0.0944	0.1301	0.1437
AMM (Yao et al., 2023)	0.0292	0.0401	0.0435	0.0668	0.1115	0.1274	0.2197	0.3295	0.3670	0.9182	0.8955	0.8879	0.1072	0.1102	0.1111	0.0869	0.1077	0.1136	0.0906	0.1281	0.1425
ClusterReg (Zhao et al., 2024)	0.0206	0.0282	0.0300	0.0289	0.0394	0.0427	0.2261	0.3171	0.3532	0.9009	0.8723	0.8619	0.1060	0.1116	0.1136	0.1010	0.1257	0.1339	0.0842	0.1192	0.1320
TransMatch (Trappolini et al., 2021)	0.0560	0.0805	0.0924	0.0929	0.1349	0.1548	0.4320	0.6495	0.7525	0.8779	0.8224	0.7936	0.0847	0.0986	0.1055	0.1168	0.1682	0.1934	0.1754	0.2611	0.3042
NDP (Li & Harada, 2022)	0.0171	0.0214	0.0224	0.0174	0.0216	0.0227	0.1236	0.1543	0.1636	0.9620	0.9524	0.9488	0.0461	0.0527	0.0548	0.0381	0.0480	0.0509	0.0547	0.0694	0.0749
LNDP (Li & Harada, 2022)	0.0357	0.0481	0.0519	0.0511	0.0640	0.0682	0.1560	0.1911	0.1989	0.9263	0.9119	0.9069	0.1110	0.1152	0.1162	0.0913	0.1072	0.1121	0.0600	0.0731	0.0799
DPF (Prokudin et al., 2023)	**0.0088**	**0.0102**	**0.0105**	**0.0092**	**0.0106**	**0.0110**	0.1864	0.2534	0.2794	0.9635	0.9508	0.9468	**0.0290**	**0.0357**	0.0384	0.0259	0.0350	0.0376	0.0789	0.1069	0.1160
OAReg (Zhao et al., 2025a)	0.0244	0.0316	0.0336	0.0269	0.0353	0.0379	0.1429	0.1871	0.2029	0.9561	0.9446	0.9402	0.0673	0.0731	0.0751	0.0488	0.0615	0.0658	0.0611	0.0822	0.0888
ActionMesh (Sabathier et al., 2026)	0.0260	0.0280	0.0141	0.0500	0.0757	0.0205	0.1412	0.1925	**0.0594**	0.9716	0.9694	0.9685	0.0363	0.0376	**0.0383**	0.0235	0.0270	0.0301	0.0489	0.0643	**0.0259**
Ours (w/o img)	0.0126	0.0111	0.0115	0.0403	0.0413	0.0440	**0.0867**	**0.0951**	0.1003	**0.9801**	**0.9754**	**0.9721**	0.0317	0.0495	0.0548	**0.0213**	**0.0252**	**0.0275**	**0.0292**	**0.0356**	0.0379

with Table 1, these numbers illustrate why surface fit alone is not a reliable indicator of registration quality: DPF combines the best surface fit with an NGE almost three times ours and a clearly worse CD-Motion, *i.e.*, its vertices reach the target surface but at the wrong locations, which is visible as vertex entanglement in Fig. 6 and Fig. 11. In the same vein, LPIPS, a low-level perceptual distance between rendered normal maps, rewards the tightly fitted surfaces of DPF, whereas the human-aligned DreamSim and the semantic CLIP similarity favor our registrations.

On DT4D, the per-setting results show that our method is particularly strong with sparse inputs and depends little on input density, whereas ActionMesh benefits substantially from dense temporal observations. Finally, the DT4D targets are clean artist-made character meshes whereas the ActionBench inputs are generated meshes, and the consistent average ranking across the two indicates that the deformation prior learned on TexVerse transfers across data sources.

Comparison with non-neural registration methods. Non-rigid registration methods fall into two families, non-neural methods that optimize an explicit deformation model for each instance, and neural methods that parameterize the deformation with a network, either optimized per instance or, as in ActionMesh and our method, predicted in a feed-forward pass. Among the baselines of the main paper, only ClusterReg (Zhao et al., 2024) belongs to the first family. For a complete picture, Tables 4, 5 and 6 additionally include six representative non-neural methods, namely the non-rigid ICP of Amberg et al. (2007), the embedded-deformation formulation of Sumner et al. (2007), the former initialized by FPFH (Rusu et al., 2009) feature matching, Bayesian coherent point drift (Hirose, 2020), its geodesic-kernel extension (Hirose, 2022), and the majorization–minimization solver AMM (Yao et al., 2023). Across all three tables, none of them matches the best neural baselines, let

Table 6: **Additional metrics averaged over the 2-, 4-, and 8-frame settings.** Rows are grouped into non-neural methods (top; see the discussion at the end of Appendix G), neural baselines (middle), and our method (bottom). CD-3D and CD-4D measure the Chamfer distance between the registered and ground-truth surfaces after per-frame and first-frame ICP alignment, respectively, and LPIPS compares rendered normal maps. Best results are in **bold**, second-best are underlined.

Method	ActionBench			DeformingThings4D		
	CD-3D ↓	CD-4D ↓	LPIPS ↓	CD-3D ↓	CD-4D ↓	LPIPS ↓
Amberg NRICP (Amberg et al., 2007)	0.0666	0.0891	0.1096	0.0689	0.2728	0.1387
Sumner NRICP (Sumner et al., 2007)	0.0652	0.0894	0.1129	0.0636	0.5730	0.1406
FPFH (Rusu et al., 2009) + Amberg NRICP	0.0657	0.0872	0.1200	0.0502	0.1234	0.0957
BCPD (Hirose, 2020)	0.0570	0.0858	0.0969	0.0401	0.0681	0.1084
GBCPD (Hirose, 2022)	0.0582	0.0896	0.1028	0.0419	0.0700	0.1166
AMM (Yao et al., 2023)	0.0579	0.0842	0.0823	0.0376	0.1019	0.1095
ClusterReg (Zhao et al., 2024)	0.0545	0.0835	0.1130	0.0263	0.0370	0.1104
TransMatch (Trappolini et al., 2021)	0.0975	0.1762	0.1670	0.0763	0.1275	0.0963
NDP (Li & Harada, 2022)	0.0545	0.0843	0.0623	0.0203	0.0206	0.0512
LNDP (Li & Harada, 2022)	0.0585	0.0852	0.1037	0.0452	0.0611	0.1141
DPF (Prokudin et al., 2023)	**0.0531**	**0.0820**	**0.0369**	**0.0099**	**0.0103**	**0.0344**
OAReg (Zhao et al., 2025a)	0.0562	0.0847	0.0641	0.0299	0.0334	0.0718
ActionMesh (Sabathier et al., 2026)	0.0560	0.0842	0.0821	0.0227	0.0487	0.0374
Ours (w/o img)	0.0571	0.0840	0.0408	0.0117	0.0419	0.0454
Ours	0.0567	0.0834	0.0399	–	–	–

alone our method, and they fall furthest behind on DT4D, where the large motion between frames must be recovered from geometry alone without a learned prior. They are also considerably slower than our method.

H Additional Ablation Studies

Table 7: **Comparison of anchor encoders.** We replace our GCN-style topology-aware encoder (Sec. 3.1) with DiffusionNet (Sharp et al., 2022) and PoissonNet (Maesumi et al., 2025), keeping every other component unchanged. Following the ablation protocol of Sec. 4.2, all variants are trained on TexVerse for 100K iterations and evaluated on 4-frame ActionBench sequences with all metrics of the ActionBench protocol (Appendix F). Best results are highlighted.

Anchor Encoder	CD-3D ↓	CD-4D ↓	CD-M ↓	CLIP ↑	LPIPS ↓	DreamSim ↓
DiffusionNet (Sharp et al., 2022)	0.0594	0.0873	0.1611	0.9713	0.0580	0.0301
PoissonNet (Maesumi et al., 2025)	0.0753	0.1152	0.2209	0.9587	0.0962	0.0541
Ours (GCN-style)	**0.0573**	**0.0853**	**0.1539**	**0.9753**	**0.0483**	**0.0231**

Choice of the anchor encoder. Our topology-aware encoder (Sec. 3.1) aggregates features over the mesh connectivity with a simple GCN-style scheme. A natural question is whether a more expressive intrinsic mesh network, which operates on the same connectivity, would serve better as the anchor encoder. To test this, we replace our encoder with two widely used intrinsic mesh networks, DiffusionNet (Sharp et al., 2022) and the recent PoissonNet (Maesumi et al., 2025), keeping all other components unchanged and following the ablation protocol of Sec. 4.2. Table 7 shows that our simpler encoder is the best on every metric. We attribute this to two likely factors. First, the anchor encoder only needs to provide connectivity-aware *local* descriptors, since global context is supplied downstream by the transformer-based motion encoder and the shape-latent prior; the global receptive field of intrinsic networks therefore brings little benefit here. Second, both networks inherit the conditioning of the discrete differential operators they build on (the cotangent Laplacian and its eigenbasis, gradient and divergence), which become ill-conditioned on imperfect meshes, *e.g.*, degenerate or non-manifold faces and many small disconnected components, each of which adds a null-space direction to the Laplacian. Such defects are common in the ActionBench inputs used for this ablation, which are generated by an image-to-mesh network rather than modeled by artists, so this weakness is particularly exposed here. Our aggregation instead uses only the row-normalized combinatorial adjacency, which is defined for any triangle soup and indifferent to triangle quality, and its purely local message passing has few numerical dependencies and is easy to optimize. Although both baselines are used with their default hyperparameters, the consistent ranking on all

six metrics indicates that stronger intrinsic operators do not translate into better registration in our setting.

Effect of the topology-aware representation. ActionBench contains few sequences that exhibit vertex entanglement. To quantify the impact of our topology-aware point representation, we therefore build a near-contact benchmark of 50 sequences across diverse object categories. Each sequence starts from a procedurally modeled object whose parts are in near contact and animates it by transforming its articulated part, *e.g.*, translating a cap or rotating a lid about its hinge, so that the touching surfaces separate. Such spatially close yet geodesically distant surfaces are exactly where vertex entanglement arises, which makes the benchmark a direct test of the topology-aware design.

Since all frames share the same vertices, the ground-truth correspondence of every vertex is known. We therefore report NGE together with the vertex error, *i.e.*, the Euclidean distance between each output vertex and its ground-truth position, normalized by the diagonal length of the first frame's bounding box. We further select near-contact pairs, *i.e.*, vertex pairs that are close in Euclidean distance yet geodesically distant, compute the vertex error on their vertices, and count a pair as unseparated if its predicted distance is below half of the true one.

Table 8 compares our final model with a variant without the topology-aware representation, which falls back to per-vertex positions and normals (Sec. 3.1). Our representation improves all metrics, reducing the near-contact vertex error by 42% and the fraction of unseparated pairs from 23.2% to 5.1%. Qualitative comparisons in Fig. 10 further confirm its effectiveness. Without the topology-aware representation, both sides of a contact move together and drag patches of one surface across the opening gap between them.



Figure 10: **Examples from the near-contact benchmark.** Left to right: the anchor, the target, and each variant's result, colored by per-vertex tracking error from white (low) to red (high).

Table 8: **Effect of the topology-aware representation on the near-contact benchmark.** Vertex errors are in percent of the bounding-box diagonal and averaged over all target frames and sequences.

Variant	Vertex error ↓		Unseparated ↓	NGE ↓
	All	Near contact	pairs (%)	
w/o Topo.	3.01	4.29	23.2	0.0333
Ours	**1.98**	**2.50**	**5.1**	**0.0209**

I Evaluation of Motion Interpolation

Since the decoder is queried by timestamp rather than by frame index (Sec. 3.3), it can generate frames that the encoder never sees. We test this on DT4D, whose ground-truth meshes give exact correspondences at every frame. The encoder receives only the keyframes $\{0, 15\}$ or $\{0, 5, 10, 15\}$ of a 16-frame sequence, and we evaluate the decoder's outputs at all remaining frames. We construct two baselines from our registered keyframes, which share the anchor's vertex indexing. The nearest-keyframe baseline copies the temporally closest keyframe into each held-out frame, and linear interpolation blends the vertices of the two neighboring keyframes. We evaluate on all 158 DT4D evaluation sequences (Appendix E). Since linear interpolation is exact for straight-line motion at constant speed, we additionally evaluate on the 50 sequences whose motion deviates most from it, *i.e.*, whose ground-truth held-out frames have the largest mean vertex distance to the linear blend of the neighboring ground-truth keyframes at $\{0, 5, 10, 15\}$. The same 50 sequences are used for both settings. As shown in Table 9, our continuous queries give the lowest CD-4D, CD-Motion, and NGE on the full set in both settings, and their margin over linear interpolation grows on the

Table 9: **Evaluation on held-out intermediate frames.** On all 158 DT4D evaluation sequences and on the 50 with the largest deviation from linear-in-time motion (see text), the encoder observes only the keyframes $\{0, 15\}$ or $\{0, 5, 10, 15\}$ of a 16-frame sequence and the decoder is queried at the remaining timestamps. All metrics are computed *only* on these held-out frames, which the encoder never observes. Both baselines are built from our registered keyframes. The nearest-keyframe baseline copies the temporally closest one, and linear interpolation blends the corresponding vertices of the two neighboring keyframes. Best results per setting are highlighted.

Sequences	Encoder Input	Method	CD-3D ↓	CD-4D ↓	CD-M ↓	NGE ↓	CLIP ↑	LPIPS ↓	DreamSim ↓
All (158)	2 keyframes	Nearest keyframe	0.0589	0.2471	0.5425	0.1601	0.9249	0.1341	0.1024
		Linear interpolation	0.0586	0.1864	0.4161	0.1410	**0.9295**	0.1212	0.0981
		Ours (continuous query)	**0.0575**	**0.1752**	**0.4072**	**0.1371**	0.9292	**0.1176**	**0.0972**
	4 keyframes	Nearest keyframe	0.0424	0.1241	0.3171	0.1172	0.9403	0.1037	0.0766
		Linear interpolation	**0.0413**	0.1024	0.2610	0.1024	**0.9441**	**0.0927**	**0.0727**
		Ours (continuous query)	0.0428	**0.0983**	**0.2558**	**0.1005**	0.9420	0.0929	0.0749
Nonlinear (50)	2 keyframes	Nearest keyframe	0.0714	0.3221	0.7120	**0.1837**	0.9071	0.1730	0.1284
		Linear interpolation	0.0719	0.2954	0.6818	0.1911	**0.9083**	0.1682	0.1289
		Ours (continuous query)	**0.0713**	**0.2690**	**0.6543**	0.1844	0.9074	**0.1605**	**0.1277**
	4 keyframes	Nearest keyframe	0.0608	0.1822	0.4882	0.1580	0.9172	0.1443	0.1076
		Linear interpolation	0.0612	0.1692	0.4525	0.1530	**0.9202**	0.1355	0.1060
		Ours (continuous query)	**0.0606**	**0.1535**	**0.4317**	**0.1475**	0.9199	**0.1344**	**0.1053**

nonlinear subset, where they reduce CD-4D by 9%. The rendering-based metrics of our queries and linear interpolation are close, and CD-3D is nearly tied because it re-aligns every frame and thereby absorbs most of the pose error. Continuous-time decoding thus improves correspondence over linear interpolation of the same registered keyframes, most clearly on nonlinear motion.

J Limitations and Future Work

Despite its strong performance across diverse scenarios, MeshLoom still has several limitations. First, although the network internalizes topology information, the registration fidelity depends on the quality of the anchor mesh. Since every output inherits the anchor connectivity, the model cannot represent topology changes such as tearing or merging, and if the anchor fails to faithfully capture the connectivity of the underlying object structure, it cannot, by itself, recover the missing connectivity during registration. For the latter, a promising remedy is to first reconstruct an A-pose anchor mesh whose connectivity is clearly expressed. Second, the network struggles under extreme motion intensities. When two consecutive frames differ by a large rigid transformation, *e.g.*, a near-symmetric human character rotating 180°, it becomes difficult to disambiguate correspondence, and the encoder loses track of the underlying motion. Incorporating stronger cues, such as multi-view renderings, is a promising direction for addressing this issue. Finally, since our current model is trained on object-centric 4D animation data, scaling to scene-level registration with multiple interacting objects remains an open direction, where the main bottleneck is likely to be data scarcity.

K Additional Qualitative Results

Fig. 11 provides additional visual comparisons with the baselines on ActionBench and DeformingThings4D, complementing Fig. 6 of the main paper. Figs. 12 and 13 show further registration results of our method across a broader range of motions and object identities on the two benchmarks; on DT4D, our results are obtained in the image-free mode. Please also refer to our supplementary webpage for interactive 3D visualizations.

Visualization. The colored coordinates in our correspondence figures (Figs. 1, 7, 8, 12, and 13) are obtained by coloring the anchor mesh by its geodesic distance to an extreme vertex, computed with Dijkstra's algorithm on the mesh edges and mapped to a rainbow colormap. Every registered frame inherits these per-vertex colors through the shared vertex indexing, so that each color marks the same surface point across frames. The error maps in Figs. 3, 5, 6, 10, and 11 instead color each registered vertex by its tracking error, *i.e.*, its distance to the ground-truth position of the same surface point as a fraction of the ground-truth bounding-box diagonal, on a white-to-red ramp that saturates at 2% (Figs. 3 and 5) or 10% (Figs. 6, 10, and 11). On DT4D and the near-contact benchmark, this position follows directly from the shared vertex indexing, whereas on ActionBench the ground-truth point trajectories are attached to the anchor by barycentric snapping. Error maps are not shown for 4D-DRESS (Fig. 7), whose independently reconstructed scans provide no ground-truth correspondence.

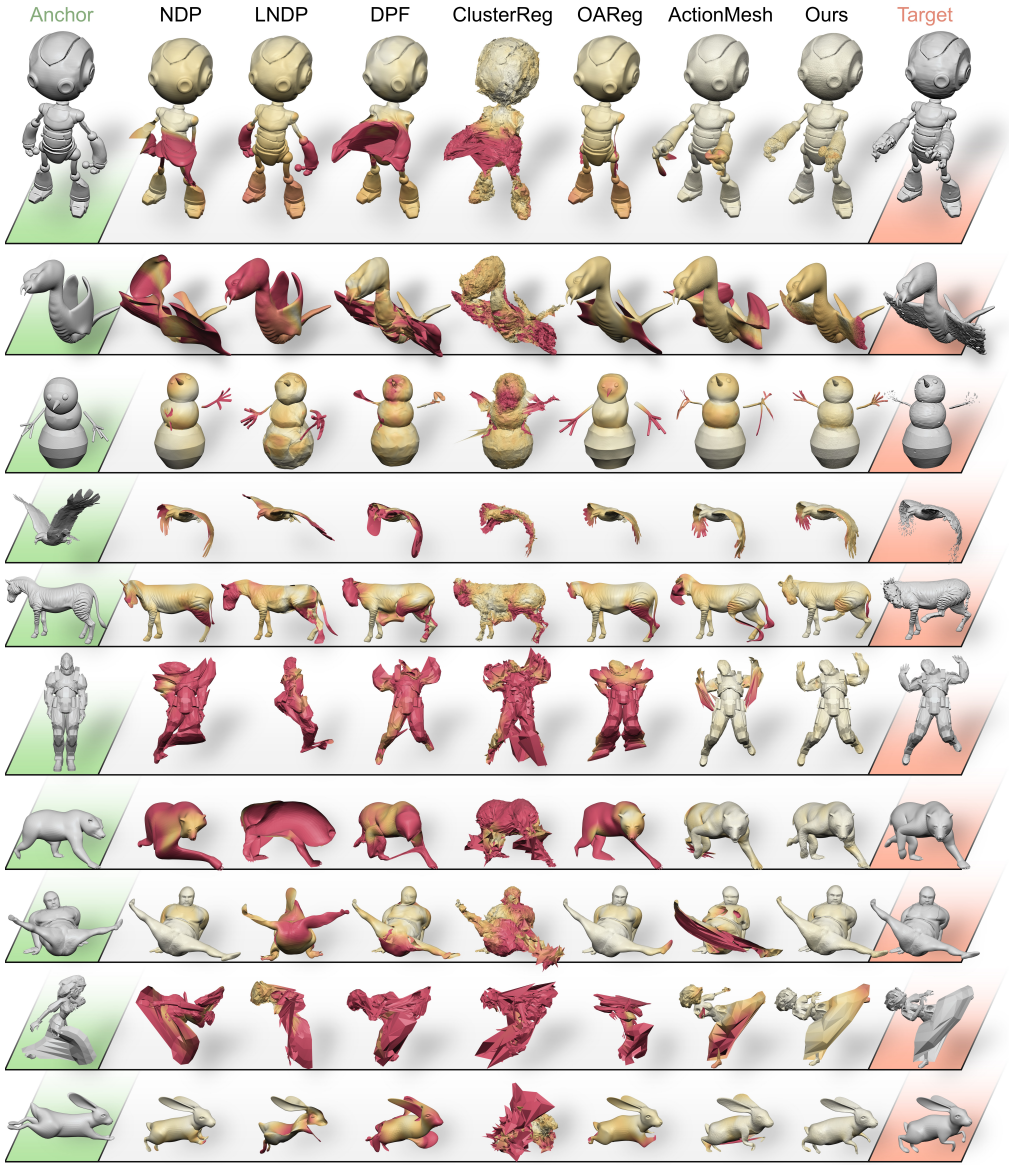

Figure 11: **Additional visual comparisons with baseline methods on ActionBench (top five rows) and DeformingThings4D (bottom five rows).** From left to right: the anchor mesh, the registered results of each method, and the target mesh. As in Fig. 6, registered meshes are colored by per-vertex tracking error, from white (low) to red (high). On DT4D, ours runs image-free.

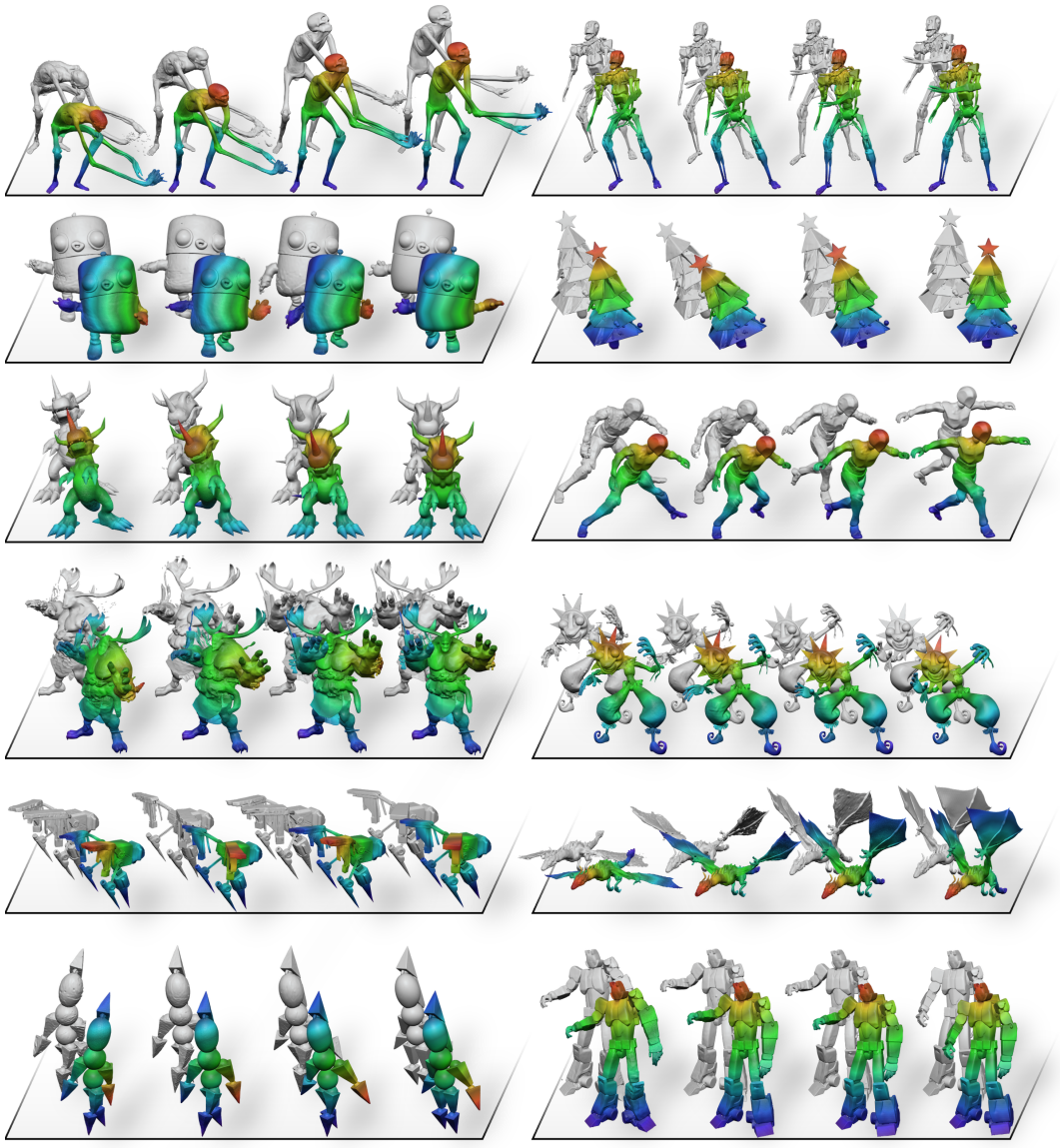

Figure 12: **Additional registration results of our method on ActionBench.** We show our registration results across a broader range of motions and object identities. Unregistered input meshes are rendered in pure white, while registered outputs are rendered with colored coordinates that visualize the vertex correspondence across mesh frames.



Figure 13: **Additional registration results of our method on DeformingThings4D.** Each panel shows one sequence of animal or humanoid motion. Unregistered input meshes are rendered in pure white, while the registered outputs, obtained in the image-free mode, are rendered with colored coordinates that visualize the vertex correspondence across mesh frames.