RfM: Zero-Shot Rigging from Video Diffusion Models

Shuohan Tao[1], Shunyuan Zheng[2], Boyao Zhou[2]

Georgia Institute of Technology[1], Ant Group[2]

Abstract

We present Rig-from-Motion (RfM), the first zero-shot framework that achieves automatic rigging and generates physically plausible free-view animation rendering. Unlike existing mesh rigging methods which are constrained by their reliance on large, manually-rigged datasets, our method completely avoids rigged training data and generalizes to any mesh topology. Guided by motion priors from a pre-trained video diffusion model, RfM animates a given static mesh from text prompts while generating an accurate, fine-grained skeleton. The optimization is driven by a Score Distillation Sampling (SDS) loss to ensure plausible motion. Our core innovation is a stochastic skeleton representation where each potential joint has an existence probability associated with its kinematics. Such representation allows us to apply the Hard Concrete distribution framework to optimize both skeletal topology and kinematics. During the optimization, we further propose regularization terms such as sparsity-inducing, self-intersection and gravity in order to discover a sparse, plausible skeletal structure. Qualitative and quantitative results demonstrate that RfM presents a compelling, data-free alternative to conventional rigging techniques.

1 Introduction

3D character animation plays a crucial role from filmmaking to video game industries (Peng et al. 2024; Holden, Saito, and Komura 2015). Within the pipeline, skeletal rigging, the process of creating an internal skeleton and attaching it to the mesh's surface, is one of the most significant steps. Traditionally, this is a manual, time-consuming task that requires significant expertise, posing a major labor bottleneck in the pipeline. Therefore, developing methods to automate rigging has been a long-standing goal in computer graphics research.

To automate this tedious process, early approaches relied on algorithmic and geometric techniques (Baran and Popović 2007; Tagliasacchi, Zhang, and Cohen-Or 2009). These methods operate by analyzing the input mesh's shape, often by calculating its medial axis or using geometric heuristics to place bones inside the limbs. Although these approaches have the advantage of not requiring any training data, they are often brittle. They typically rely on strong assumptions about the mesh, *e.g.* it being a clean, watertight, human-like model, and they frequently struggle with unconventional or stylized assets that do not fit a standard template.

To overcome these limitations, the field has shifted towards data-driven methods using deep learning for template-free rigging. As a pioneering work, RigNet (Xu et al. 2020) leverages graph neural network to predict joint position and establishes root to joint connection via traditional Minimum Spanning Tree algorithm. The following methods (Zhang et al. 2025; Liu et al. 2025; Song et al. 2025b,a) utilize auto-regressive model (Song et al. 2025b,a) or coarse-to-fine strategy (Deng et al. 2025; Guo et al. 2025) to hierarchically predict skeletons and skinning weights. Although such learning-based methods have demonstrated impressive results, they are constrained to skeleton rigging based solely on the geometric properties of the input mesh, unable to reflect **dynamic characteristics**. Apart from that, these methods learn a direct mapping from a given mesh to its skeleton by training on large-scale datasets (Deitke et al. 2023b,a; Song et al. 2025b,a), which are painstakingly rigged by human experts. This strong dependency results in **overfitting** to structural patterns of the training data, limiting generalization to novel topology.

In this work, we ask: can we rig general meshes without ever seeing a single rigged example? We find the answer by looking to a new source of knowledge: large-scale generative models. Recent advances in video diffusion models (Wang et al. 2023; Ho et al. 2022; Blattmann et al. 2023) have produced systems that hold a powerful, implicit understanding of dynamics and motion, learned from observing millions of diverse, real-world videos. The bridge that connects these powerful 2D video models to the world of 3D geometry is differentiable rendering. This technique allows us to calculate how a change in a 3D model's parameters—such as the position of a bone—will affect a rendered 2D image, enabling us to use image-based feedback to optimize a 3D scene. Furthermore, techniques like Score Distillation Sampling (SDS) (Poole et al. 2023) allow a pre-trained diffusion model to act as a loss function, guiding an optimization process to create plausible content. This approach has already shown remarkable success in generating dynamic 4D content from text prompts, as seen in works like 4D-fy (Bahmani et al. 2024) and Align Your Gaussians (Ling et al. 2024). Others have even used these priors to guide complex physics simulations (Lin et al. 2025). While these works generate motion, they typically do so for whole objects or assume the underlying articulated structure is already known. The most related

work to ours, AKD (Li et al. 2025), demonstrates that video priors can animate a character, but it requires a pre-defined skeleton as input. The fundamental challenge of discovering the skeleton and skinning weights in a fully data-free manner remains an open problem.

In this paper, we introduce Rig-from-Motion (RfM), the first framework, to generate a character's full skeletal rig and skinning weights without any prepared rigged 3D data. We achieve this by framing rigging as an optimization problem for each individual mesh, mainly guided by a pre-trained video diffusion model via SDS loss. A key component of our method is a novel stochastic skeleton representation, which allows the model to automatically discover a plausible bone structure during optimization. Specifically, we represent the input mesh with a discrete medial surface to obtain skeleton initialization, containing very dense joints which cannot guarantee robust animation in downstream tasks. We include a smoothing step on top of the original medial surface approximation algorithm to merge potential component meshes into a single watertight volume. We then assign each joint a Hard Concrete random variable to differentiable approximate a binary Bernoulli on-and-off random state. Furthermore, Hard Concrete distribution also serves as a gate mechanism to select necessary joints, along with our proposed sparsity-inducing L_0 regularization, to enforce completeness and correctness of the joint prediction. We also model the skinning weight with RBF (radial basis function) from each vertex to the joints, and jointly optimize radius value with joint rotation as forward kinematics to drive the mesh with realistic motion. To ensure geometric plausibility and dynamic characteristics, we finally propose self-intersection and gravity regularization terms to enforce physically consistent motion patterns. Experiments demonstrate that our method enables automatic rigging on meshes with arbitrary topological structures while generating physically consistent animations.

In summary, our main contributions are as follows:

- We introduce RfM, a data-free automatic rigging framework by distilling from generative video prior, without using any rigged 3D training data, establishing a new paradigm for character animation.
- We propose a stochastic skeleton representation that, when paired with a differentiable sampling scheme based on Hard Concrete distribution, enables the discovery of skeletal topology as part of the optimization process.
- We integrate sparsity-inducing L_0, self-intersection and gravity regularization terms with our optimization framework to output complete, plausible and animatable skeleton structures for a diverse set of meshes.

2 Related Work

2.1 Automatic Rigging

The automation of skeletal rigging has been a long-standing pursuit in research, evolving from early geometric algorithms to modern deep learning models.

Algorithmic and Geometric Methods. Early automatic rigging work relied on hand-crafted rules and geometric analysis of the input mesh (Tagliasacchi, Zhang, and Cohen-Or 2009; Wade 2000; Katz and Tal 2003). A foundational technique involves computing the mesh's medial axis or surface to approximate an internal skeleton, as demonstrated in Pinocchio (Baran and Popović 2007). Other methods have explored shape decomposition (Huang et al. 2009), graph-based algorithms (Au et al. 2008), or voxel-based skeletonization (Yan, Letscher, and Ju 2018) to determine bone placement and influence. While these methods are "data-free", they are often based on strong heuristics that limit their robustness. They typically require clean, manifold meshes and struggle with complex topologies, stylized characters, or assets that deviate from a standard bipedal or quadrupedal form factor.

Learning-Based Methods. To overcome the brittleness of geometric methods, the field has increasingly adopted deep learning (Liu et al. 2019a; Ma and Zhang 2023; Wang et al. 2024). This paradigm trains neural networks on large datasets of artist-rigged characters to learn a direct mapping from mesh geometry to a skeletal rig. RigNet (Xu et al. 2020) was a pioneering work in this area, using a graph convolutional network to predict both a skeleton and skinning weights. Subsequent works have significantly advanced the state-of-the-art. For instance, UniRig (Zhang et al. 2025) introduced a unified skeleton representation to handle diverse character morphologies within a single model. Most recently, RigAnything (Liu et al. 2025) proposed a template-free, autoregressive model capable of rigging a wide variety of 3D assets. While these learning-based methods (Xu et al. 2020; Zhang et al. 2025; Liu et al. 2025; Song et al. 2025b,a; Deng et al. 2025) demonstrate remarkable power and generalization, they share a fundamental dependency on large-scale, supervised datasets of mesh rig pairs, which not only makes the models expensive to train but also inherently limits their ability to rig out-of-distribution assets not seen during training.

2.2 Differentiable Rendering

Differentiable rendering enables image-space objectives to optimize 3D representations (Liu et al. 2019b; Peng et al. 2021). Recent representations such as NeRF (Mildenhall et al. 2020) and 3D Gaussian Splatting (Kerbl et al. 2023) have extended this capability to efficient novel-view synthesis. Their dynamic extensions model deforming scenes by attaching renderable primitives to lower-dimensional deformation or articulation structures (Luiten et al. 2024; Yang et al. 2024; Liu, Su, and Wang 2025; Li et al. 2024).

2.3 Score Distillation Sampling

The recent success of large-scale generative models, particularly diffusion models, has provided a powerful new source of prior knowledge about the natural world. Specifically, Score Distillation Sampling (SDS), first introduced in DreamFusion (Poole et al. 2023), is a powerful technique to harness this knowledge. SDS allows a pre-trained frozen diffusion model to be used as a loss function, guiding the optimization of a 3D representation (e.g., a NeRF or 3DGS) to be consistent with the model's learned prior. This concept has been rapidly adopted for 3D and 4D content creation. For example, 4D-fy (Bahmani et al. 2024) and Align Your Gaussians (Ling et al. 2024) leverage SDS with text-to-video dif-

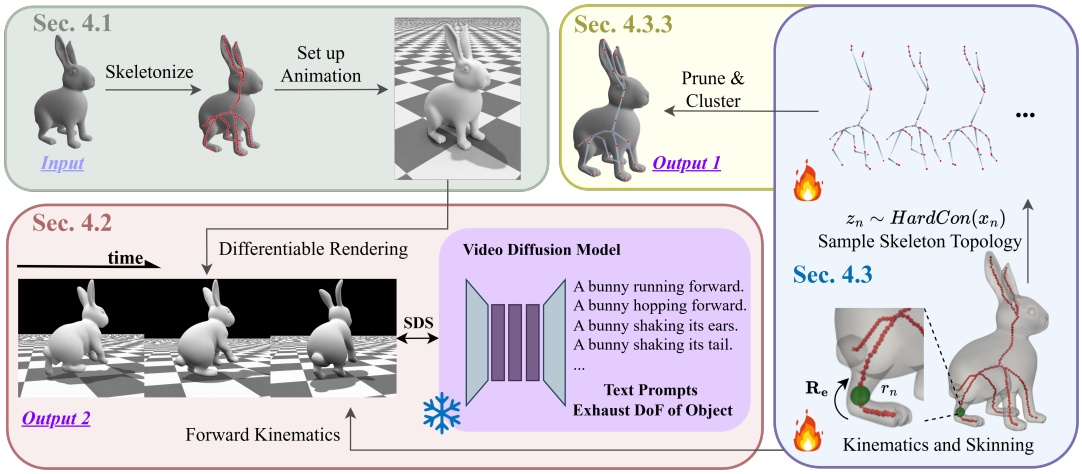


Figure 1: **Overview of our zero-shot rigging framework.** We initialize a dense skeleton and optimize its kinematics and skinning weights using motion priors distilled from a pretrained video diffusion model. A differentiable Hard Concrete representation then prunes redundant joints while preserving the recovered motion.

fusion models to generate dynamic 4D scenes of deforming objects from a text prompt. OmniPhysGS (Lin et al. 2025) further showed that these generative priors can be used to guide physics-based simulations, demonstrating their versatility. While these methods effectively generate motion, they typically deform a holistic 3D representation and do not infer a structured, reusable, articulated skeleton suitable for downstream animation pipelines. The work most adjacent to ours is AKD (Li et al. 2025), which generates plausible motion from video priors. Crucially, their method requires a pre-rigged 3D asset as input and focuses solely on optimizing motion for that fixed rig. On the contrary, our RfM, which integrates automatic rigging and animation as a unified, learnable output, addresses a fundamentally different problem.

3 Preliminaries

3.1 Video Score Distillation Sampling

We use video score distillation sampling (SDS) to transfer a motion prior from a frozen video diffusion model to the rig parameters. At each optimization step, the rendered video is encoded into a latent z, noised at diffusion timestep t, and denoised by the video diffusion model. The SDS gradient is

$$\nabla_z \mathcal{L}_{\text{SDS}} = \mathbb{E}_{t,\epsilon}\left[w(t)\big(z - \hat{z}\big)\right], \tag{1}$$

where $w(t)$ is the timestep-dependent weighting and $\hat{z}$ is the predicted clean latent. This gradient is propagated through the fixed renderer and linear-blend-skinning pipeline to the kinematic, skinning, and skeleton-gating parameters; the rendering appearance is kept fixed. The remaining SDS details are provided in the supplementary material.

3.2 Hard Concrete Joint Gates

To obtain a compact skeleton, each candidate joint is assigned a differentiable Hard Concrete gate (Louizos, Welling, and Kingma 2018). Given a learnable logit x_n and $u_n \sim \mathcal{U}(0,1)$, a gate is sampled as

$$s_n = \text{Sigmoid}\left(\frac{\log u_n - \log(1-u_n) + x_n}{T}\right), \tag{2}$$

$$z_n = \text{clip}_{[0,1]}\big(s_n(\zeta - \gamma) + \gamma\big), \tag{3}$$

where T is the temperature and $\gamma < 0 < 1 < \zeta$ stretches the Binary Concrete sample before clipping. Its nonzero probability gives a differentiable expected L_0 penalty:

$$\begin{aligned} \Pr(z_n \neq 0) &= \text{Sigmoid}\left(x_n - T\log\frac{-\gamma}{\zeta}\right), \\ \mathcal{L}_0 &= \sum_{n=1}^{N} \Pr(z_n \neq 0). \end{aligned} \tag{4}$$

The full mixed density, including its point masses and implementation details, is included in the supplementary material.

4 Method

The overall pipeline is illustrated in Figure 1, which includes initialization in Section 4.1, animation setup in Section 4.2, and skeleton distillation in Section 4.3.

4.1 Dense Skeleton Initialization

To obtain a discrete medial surface representation of meshes, we voxelize the mesh to construct a binary occupancy grid $V \in \{0,1\}^{N_x \times N_y \times N_z}$. We then apply a 3D Gaussian smoothing filter with standard deviation σ:

$$V_\sigma(\mathbf{x}) = V(\mathbf{x}) * G(\mathbf{x};\sigma), \tag{5}$$

where $*$ stands for convolution operator. This step dilates the volume to close small gaps in multi-shell meshes. Otherwise, some unwanted skeleton structures could appear as in Figure 2(a). We binarize V_σ at threshold 0.5 and apply a robust 3D medial surface thinning algorithm (Lee, Kashyap, and

Chu 1994) to skeletonize the volume, yielding voxel centers as the discrete medial surface representation.

These centers form an ideal solution space for optimization. We initialize joints on the voxel centers for dense skeletons, construct a minimum spanning tree via Prim's algorithm (Prim 1957). We choose the joint closest to the mesh center of mass as root and build hierarchy from the tree.

As the joint density is directly coupled to the voxel grid resolution, which can vary across meshes, we resample each skeletal chain based on a shared hyperparameter d_{min}. Starting from each leaf node, we traverse the chain and greedily prune intermediate joints, ensuring the spacing between any two consecutive joints along a bone is at least d_{min}. This step ensures the spatial resolution of joints is dictated by a shared hyperparameter d_{min} across different meshes.

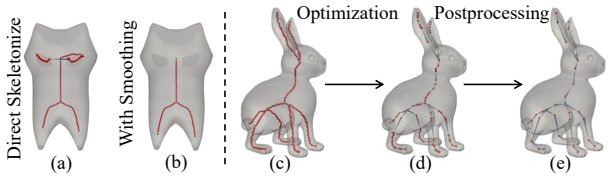


Figure 2: **Left: Comparison between (a) vanilla skeletonization algorithm and (b) our modified version.** Our version is robust when the input mesh is composed of several component meshes. **Right: Visualization of skeleton topology after each stage of our pipeline**, from (c) to (e).

4.2 Animation Setup

To expose the mesh geometry to the video prior without introducing an appearance objective, we render the deformed mesh using fixed grayscale colors derived from its surface normals. The Gaussian colors and other rendering attributes are initialized once and remain fixed throughout optimization; only the skeleton topology, kinematics, and skinning parameters receive gradients from the SDS objective. More detailed explanation is available in the supplementary material. With robust appearance representation under mesh deformation and plausible joint candidates, we can animate the mesh using the dense sets of skeletons. To begin, we initialize per-edge axis angle rotations $\mathbf{R_e} \in \mathbb{R}^{P\times T\times N\times M\times 3}$, which are gathered and stored by the parent nodes of each edge. N is the count of joints, M is the maximum number of child edges a node has in the set of skeleton, T is the total number of frames of each motion sequence, and P is the number of prompts. Each node has an optimizable radius r_n, the skinning weights from each vertex to the nodes are defined by RBF interpolation from K nearest nodes, and the mesh is deformed using LBS (Linear Blend Skinning):

$$w_{i,n} = \frac{\phi\left(\frac{\|\mathbf{v}_i - \mathbf{p}_n\|}{r_n}\right)}{\sum_{m\in N_k(i)} \phi\left(\frac{\|\mathbf{v}_i - \mathbf{p}_m\|}{r_m}\right)}. \tag{6}$$

In the formula above, $N_k(i)$ is the set of K nearest nodes to vertex vector $\mathbf{v}_i$, $\mathbf{p}_n$ is the corresponding node vector, ϕ is the Gaussian kernel for RBF, and $w_{i,n}$ is the skinning weight from node n to vertex i. This weight is normalized over the K nearest nodes for each vertex to get the final contribution from each node. We evaluate forward kinematics using the dense skeleton to warp the mesh to each individual frame and render the Gaussians from randomly sampled static cameras. SDS loss is calculated according to Section 3.1.

Following (Li et al. 2025), we use a checkerboard ground with dark skybox as background, a ground penetration regularization to distill physically consistent kinematics from video diffusion model, and a motion smoothness regularization to penalize magnitude of mean acceleration of vertices, shown in Eq. (7) and Eq. (8) respectively, where V is the set of all vertices in the mesh, and we approximate the second derivative in Eq. (8) with central finite difference method. $\langle . \rangle_{i,t}$ represents taking mean over vertices and timesteps. We avoid these setups for motions not plausible on the ground (swimming and flying):

$$L_{\text{ground}} = \frac{1}{|\{v \in V \mid v_y < 0\}|} \sum_{v\in V, v_y<0} |v_y|. \tag{7}$$

However, we observed that this regularization alone often results in floating meshes, and we introduce a simple gravity regularization as:

$$L_{\text{smooth}} = \left\langle \left\| \frac{d^2\mathbf{v}_i(t)}{dt^2} \right\| \right\rangle_{i,t}, L_{\text{gravity}} = \frac{1}{|V|}\sum_{v\in V} v_y. \tag{8}$$

To prevent self-intersections, we introduce a regularization that uses the dense skeleton as a proxy. Each node is assigned a spherical barrier with a radius, q_n, equal to its distance to the nearest vertex on the canonical mesh. A regularization proportional to the intrusion distance is applied when one node enters another's barrier, provided their distance on the skeletal tree is at least 10.

$$L_{\text{crossing},m,n} = \mathbb{I}\left(d_T(m,n) \geq 10\right) \cdot \text{ReLU}\left(q_n - \|\mathbf{p}_m - \mathbf{p}_n\|\right) \tag{9}$$

where $L_{\text{crossing},m,n}$ is the penalty received by node m for being within the energy barrier of n, and $d_T(m,n)$ is the graph distance between node m and n.

4.3 Stochastic Skeleton Modeling

To learn an optimal sparse skeleton, we treat its topology as a stochastically sampled version of a dense skeleton. Each joint is associated with the Hard Concrete distribution $HardCon(x_n)$ defined in Section 3.2. The logit x_n is optimized jointly through L_{total} in Eq. (15).

To sample skeletons, we gate each joint's dynamic motion using a Hard Concrete sample z_n. We reparameterize the rotation $\mathbf{R_e}$ into a static component $\langle \mathbf{R_e} \rangle_{P,T}$ (the mean over prompt and time) and a dynamic residual. The variable z_n scales only the residual:

$$\mathbf{R'_e} = \langle \mathbf{R_e} \rangle_{P,T} + \mathbf{z} \odot \left(\mathbf{R_e} - \langle \mathbf{R_e} \rangle_{P,T}\right). \tag{10}$$

Here, $\mathbf{z}$ is the vector of sampled gates $z_n \sim HardCon(x_n)$. This formulation locks a joint to its learnable mean pose when $z_n = 0$ and preserves its full motion when $z_n = 1$.

Metric	RigNet	UniRig	MagicArti.	Puppeteer	AnyMate	Ours
J2B ($\times 10^{-2}$) ↓	8.24	<u>2.96</u>	5.34	4.01	3.97	**2.86**
J2J ($\times 10^{-2}$) ↓	9.98	**4.21**	6.72	5.55	5.62	<u>5.50</u>
B2B ($\times 10^{-2}$) ↓	7.87	**2.87**	5.08	3.87	3.88	<u>3.08</u>
Completeness ↑	2.90	3.49	<u>3.83</u>	3.67	3.75	**4.11**
Conciseness ↑	2.86	3.75	3.74	**4.00**	2.15	<u>3.99</u>
Correctness ↑	2.61	3.63	3.22	<u>3.89</u>	2.21	**4.08**

Table 1: **Quantitative comparison on RigXL subset.** The **optimal** and <u>suboptimal</u> methods of each metric are highlighted. On the top, we evaluate methods with computational metrics, while we conduct user study on the bottom.

Sparsity-inducing Regularization We use the expected L_0 penalty in Eq. (4) (Louizos, Welling, and Kingma 2018) to ensure sparsity of joints. However, (Louizos, Welling, and Kingma 2018) focuses on the frequential sparsity of MLP connections, while our task also requires spatial sparsity. We thus propose a generalized sparsity regularization L_{sparse} that encourages both frequential and spatial sparsity. Our proposed regularization is formulated as:

$$L_{\text{sparse}} = \frac{1}{N^2} \sum_{m=1}^{N} \sum_{n=1}^{N} H_{mn} S_{mn} = \frac{L_0}{N^2} + L_{\text{spatial}}. \quad (11)$$

The symmetric interaction matrix $\mathbf{H} \in \mathbb{R}^{N \times N}$ is defined by the geometric mean of joint activation probabilities:

$$H_{mn} = \sqrt{P(z_m \neq 0) \cdot P(z_n \neq 0)}. \quad (12)$$

Matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ has element $S_{mn} = \exp(-d_{mn}^2 / r^2)$ measuring the proximity of joints m and n based on their distance d_{mn} and a radius hyperparameter r. The diagonal elements are $S_{mm} = 1$ as distances to selves are zero. The diagonal components of the sum recover the standard L_0 regularization in Eq. (4), which encourages frequential sparsity by penalizing the total number of active joints:

$$\sum_{m=1}^{N} H_{mm} S_{mm} = \sum_{m=1}^{N} P(z_m \neq 0) \equiv L_0. \quad (13)$$

The off-diagonal terms introduce our novel spatial regularization L_{spatial}, which penalizes the co-activation of nearby joints, resulting in large values in S_{mn}, and encourages a more spatially distributed sparsity pattern.

Symmetry Regularization Our symmetry regularization L_{sym} encourages plausible structures by penalizing activation differences between symmetric joint pairs. These pairs, $(n, M(n))$, are established by mirroring the skeleton along the axis of symmetry and defining $M(n)$ as the nearest neighbor to joint n in the mirrored version. We then minimize the squared L2 distance between their Hard Concrete logits:

$$L_{\text{sym}} = \frac{1}{N} \sum_{n=1}^{N} \left(x_n - x_{M(n)} \right)^2. \quad (14)$$

For unposed meshes, we augment the optimization by flipping the rendered videos, forcing the model to learn a symmetric activation pattern.

Overall, our final loss is:

$$\begin{aligned} L_{\text{total}} = & L_{\text{SDS}} + w_{\text{smooth}} L_{\text{smooth}} + w_{\text{sym}} L_{\text{sym}} + w_{\text{ground}} L_{\text{ground}} \\ & + w_{\text{gravity}} L_{\text{gravity}} + w_{\text{crossing}} L_{\text{crossing}} + w_{\text{sparse}} L_{\text{sparse}}. \end{aligned} \quad (15)$$

Skeleton Postprocess After optimization, we prune joints whose Hard Concrete logits are below zero and merge auxiliary or nearby junction joints using DBSCAN. Then we merge neighboring junction joints that lie within d_{junction} of each other to their mean positions. For skinning, since consecutive inactive joints between two active joints form a single rigid bone, their dense RBF skinning weights are summed to obtain the corresponding sparse bone weight. Thus, each output-bone influence is a superposition of locally optimized dense RBF weights, preserving expressive spatial variation while keeping the control skeleton concise.

5 Experiments

5.1 Implementation and Prompts

We use frozen ModelScope T2V (Wang et al. 2023) as the video prior and run our experiments on a single A100 40GB. For each mesh, GPT-o3 receives one static rendering and a fixed instruction to generate motion probes covering its major functional parts. Returned sentences are used directly as SDS prompts; multiple prompts share skeleton and skinning parameters but use separate motion sequences. Full details are provided in the supplement.

5.2 Main Results

Dataset and Baselines. In total, our evaluation dataset comprises 44 meshes: the 24 sampled from RigXL, plus an additional 20 Out-of-Distribution (OOD) meshes collected from Sketchfab to compare zero-shot rigging ability with baseline methods. In particular, RigXL dataset (Zhang et al. 2025) is used to train UniRig (Zhang et al. 2025). To ensure morphological diversity, we randomly sampled these meshes from categories including "biped", "quadruped", "bird & flyer", "static", and "water creature" in RigXL. To ensure the OOD assets are truly unseen by the learning-based baselines, we exclusively selected meshes published after those methods.

While static scene generation benchmarks (e.g., DreamFusion) can evaluate on hundreds of assets, our method composes dynamic scenes via video SDS, which demands significantly heavier computation. To balance computational feasibility with a rigorous evaluation, our 44-mesh sample size strictly surpasses the standard protocols of recent dynamic

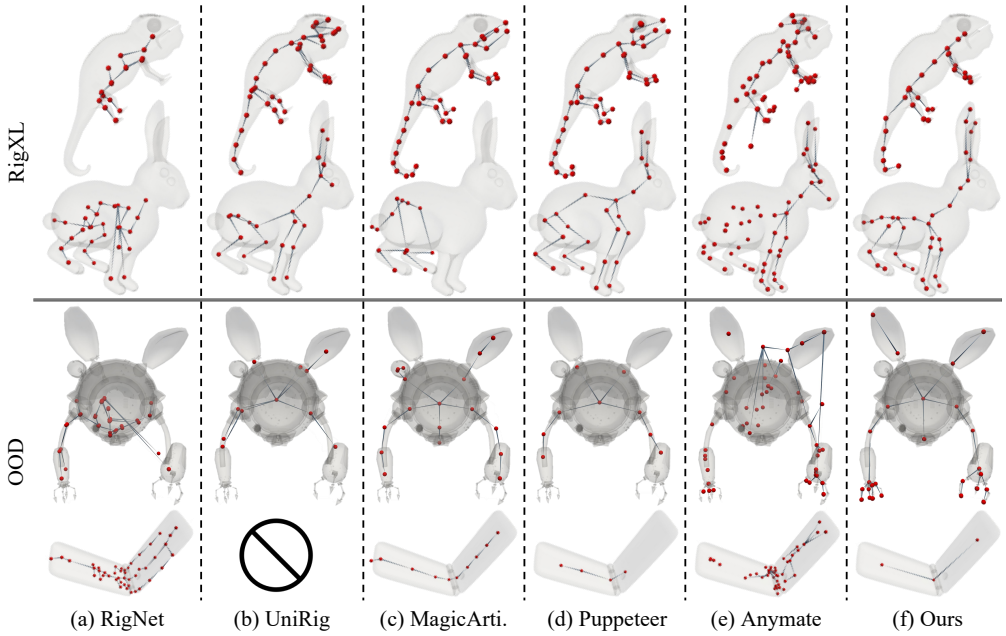


Figure 3: **Qualitative comparison with baseline methods.** In the first two rows, we show rigging results on RigXL data, and the bottom two rows illustrate the robustness of our method on Out-of-Distribution (OOD) data, with respect to learning-based baselines. Further comparison can be found in the supplementary material.

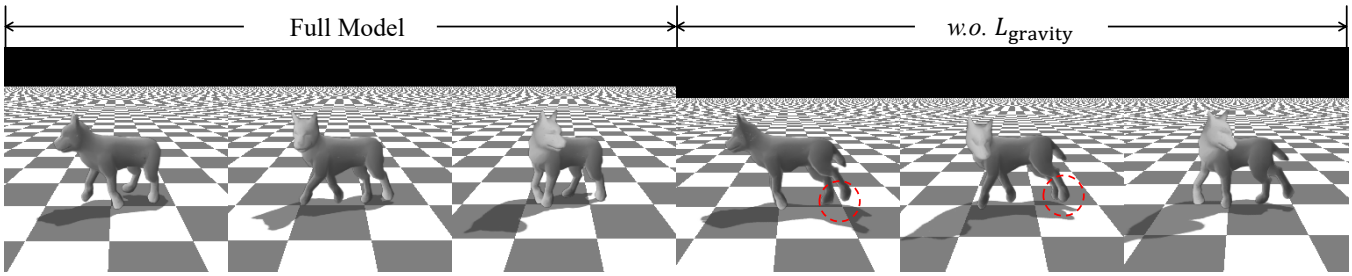


Figure 4: **Ablation study on L_{gravity}.** Our method maintains more plausible ground contact with gravity regularization. The red circles indicate regions where character would float unnaturally without gravity regularization.

video SDS-based methods (e.g., AKD, which evaluates on 29 meshes). Our rigging was benchmarked against five learning-based models: UniRig (Zhang et al. 2025), RigNet (Xu et al. 2020), Puppeteer (Song et al. 2025a), MagicArticulate (Song et al. 2025b), and AnyMate (Deng et al. 2025). Our animation is also compared against Puppeteer and AKD. We additionally evaluate skinning quality by animation reconstruction on a DeformingThings4D (Li et al. 2021) subset against UniRig and Puppeteer. Each method's joint rotations are optimized to fit the animation sequences.

Metrics & Analysis. Quantitative comparisons using J2B (Joint to Bone), J2J (Joint to Joint), and B2B (Bone to Bone) Chamfer distances are presented in Table 1. However, assessing skeleton quality by comparing against fixed GT labels presents a fundamental limitation. We notice that the rigging GT is inherently flawed due to subjective biases or technical limitations, as shown in supplementary Figure S3. To evaluate skeleton quality independently of potentially flawed GT labels, we conduct a user study in Table 1. To test zero-shot rigging ability on OOD data, we extend this user study to our 20 collected OOD meshes in Table 2. In total 18 users evaluated the completeness, conciseness, and anatomical correctness of the skeletons on a scale from 1 to 5. Furthermore, following AKD (Li et al. 2025), we utilize Semantic Adherence (SA) and Physical Commonsense (PC) scores from VideoPhy (Bansal et al. 2025) to evaluate animation quality. We also evaluate skinning quality by measuring animation reconstruction error on DeformingThings4D using Chamfer Distance (CD) and Mean Absolute Error (MAE).

Rigging Comparison. In Table 1, our method outperforms all baselines on J2B metric, and achieves suboptimal results on J2J and B2B metrics to UniRig, which is trained on RigXL dataset. We also achieved almost the best results on completeness, conciseness, and correctness in user study, as shown in Table 1 and Table 2. On DeformingThings4D, our method's skinning weight reconstructs animation more

Metric	RigNet	UniRig	MagicArti.	Puppeteer	AnyMate	Ours
Completeness ↑	2.69	2.73	3.45	3.32	<u>3.54</u>	**3.97**
Conciseness ↑	2.53	3.23	3.27	<u>3.58</u>	1.84	**4.13**
Correctness ↑	2.50	2.89	3.24	<u>3.48</u>	1.90	**4.15**

Table 2: **User study result on OOD data.** The **optimal** and <u>suboptimal</u> methods of each metric are highlighted.

	Ours	Puppeteer	AKD
SA ↑ / PC ↑	**0.37 / 0.84**	0.21 / 0.55	0.34 / 0.79
Time per Sample ↓	**1 h**	**1 h**	25 h

Table 3: **Animation quality and speed comparison.** All reported runtimes were measured on a single 40GB A100 GPU. Our method achieved better animation quality than Puppeteer at the same speed.

accurately than UniRig and Puppeteer.

The results of our OOD evaluation (Figure 3 and supplementary Figures S4 to S6) demonstrate a critical drawback in existing baseline methods: learning-based models (such as UniRig and RigNet) regressing on GT labels tend to overfit to dataset-specific artifacts (supplementary Figure S3) and lack robustness against in-the-wild, OOD data. In contrast, our method prioritizes functionality correctness, resulting in superior zero-shot performance.

Animation Comparison. Shown in Table 3, RfM achieves stronger SA and PC scores than AKD due to its physical consistency regularization. Puppeteer uses a video diffusion model to generate a video from one static view of the mesh and optimizes the motion under supervision from deep priors like model estimated optical flow and depth maps. This makes it more vulnerable to persistent occlusion, resulting its worse animation quality than RfM and AKD, both of which optimizes from SDS gradient directly from different randomly sampled views.

Speed Comparison. RfM and Puppeteer are substantially faster than AKD. Although still slower than other baseline feed-forward methods, we note here that RfM and Puppeteer additionally output animation compared to them.

	Full Model	*w.o.* L_{gravity}	*w.o.* L_{crossing}
SA ↑ / PC ↑	**0.37 / 0.84**	0.31 / 0.82	0.28 / 0.79

Table 4: **Quantitative ablation study on L_{gravity} and L_{crossing}** on RigXL subset using VideoPhy metrics.

5.3 Ablation Studies

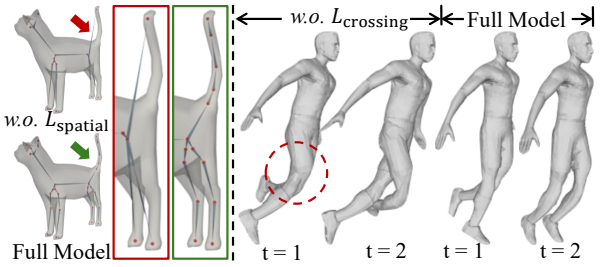
We prove the effectiveness of L_{sparse}, L_{crossing}, and L_{gravity} through ablation. Additional ablation studies are available in the supplementary material.

Effectiveness of Sparsity-Inducing Regularization. Our L_{spatial} avoids suppressing subtle motions at nearby joints by eliminating redundant clusters to save limited joint budget, preserving the knee articulation in Figure 5(a).

Effectiveness of Self-Intersection Regularization. As shown in Figure 5(b) and Table 4, the meshes self-intersect

	Ours	UniRig	Puppeteer
CD ↓	$\mathbf{1.52 \pm 0.11}$	1.86 ± 0.18	1.69 ± 0.14
MAE ↓	$\mathbf{1.17 \pm 0.06}$	1.29 ± 0.09	1.25 ± 0.12

Table 5: **Quantitative skinning comparison on DeformingThings4D, values reported in units of 10^{-2}.** Our method achieved better skinning quality than UniRig and Puppeteer.



(a) Ablation on L_{spatial} (b) Ablation on L_{crossing}

Figure 5: **Qualitative ablation study on (a) L_{spatial} and (b) L_{crossing}.** L_{spatial} maintains crucial joint motions when conducting the sparsification, as shown in (a). L_{crossing} mitigates the self-intersection phenomenon as circled in red in (b).

after ablating L_{crossing}. The model struggles to execute the alternating leg motion due to the video diffusion model's learned blurry limb, resulting in right leg crossing the left to synthesize a deceptive alternating gait. The full model successfully improved on the flawed knowledge from the video diffusion prior and created correct walking pattern. The SA and PC scores are also lower compared to full model.

Effectiveness of Gravity Regularization. Demonstrated in Table 4, removing L_{gravity} results in feet floating above the ground and lower SA and PC scores. Qualitatively, it causes the feet to float above the ground and produces less plausible kinematics, as shown in Figure 4.

6 Conclusion

This paper introduces RfM, a novel framework that generates a character's rigging from a static mesh without any pre-rigged 3D training data. By reformulating rigging as a zero-shot optimization problem, RfM bypasses the issues of data dependency and questionable evaluation benchmarks, whereas learning-based methods can be affected by inconsistent rigging quality in their training data.

Limitation. RfM inherits prompt-selection errors from the VLM and motion biases from the video prior, while stronger priors incur substantially greater runtime and memory costs.

References

Au, O. K.-C.; Tai, C.-L.; Chu, H.-K.; Cohen-Or, D.; and Lee, T.-Y. 2008. Skeleton extraction by mesh contraction. *ACM TOG*, 27(3): 1–10.

Bahmani, S.; Skorokhodov, I.; Rong, V.; Wetzstein, G.; Guibas, L.; Wonka, P.; Tulyakov, S.; Park, J. J.; Tagliasacchi, A.; and Lindell, D. B. 2024. 4d-fy: Text-to-4d generation using hybrid score distillation sampling. In *CVPR*, 7996–8006.

Bansal, H.; Lin, Z.; Xie, T.; Zong, Z.; Yarom, M.; Bitton, Y.; Jiang, C.; Sun, Y.; Chang, K.-W.; and Grover, A. 2025. VideoPhy: Evaluating Physical Commonsense for Video Generation. In *ICLR*.

Baran, I.; and Popović, J. 2007. Automatic rigging and animation of 3d characters. *ACM TOG*, 26(3): 72–es.

Blattmann, A.; Dockhorn, T.; Kulal, S.; Mendelevitch, D.; Kilian, M.; Lorenz, D.; Levi, Y.; English, Z.; Voleti, V.; Letts, A.; et al. 2023. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*.

Deitke, M.; Liu, R.; Wallingford, M.; Ngo, H.; Michel, O.; Kusupati, A.; Fan, A.; Laforte, C.; Voleti, V.; Gadre, S. Y.; et al. 2023a. Objaverse-xl: A universe of 10m+ 3d objects. *NeurIPS*, 36: 35799–35813.

Deitke, M.; Schwenk, D.; Salvador, J.; Weihs, L.; Michel, O.; VanderBilt, E.; Schmidt, L.; Ehsani, K.; Kembhavi, A.; and Farhadi, A. 2023b. Objaverse: A universe of annotated 3d objects. In *CVPR*, 13142–13153.

Deng, Y.; Zhang, Y.; Geng, C.; Wu, S.; and Wu, J. 2025. Anymate: A dataset and baselines for learning 3d object rigging. In *SIGGRAPH*, 1–10.

Guo, Z.; Xiang, J.; Ma, K.; Zhou, W.; Li, H.; and Zhang, R. 2025. Make-it-animatable: An efficient framework for authoring animation-ready 3d characters. In *CVPR*, 10783–10792.

Ho, J.; Salimans, T.; Gritsenko, A.; Chan, W.; Norouzi, M.; and Fleet, D. J. 2022. Video diffusion models. *NeurIPS*, 35: 8633–8646.

Holden, D.; Saito, J.; and Komura, T. 2015. Learning an inverse rig mapping for character animation. In *SIGGRAPH*, 165–173.

Huang, Q.; Wicke, M.; Adams, B.; and Guibas, L. 2009. Shape Decomposition Using Modal Analysis. In *Computer graphics forum*, volume 28, 407–417. North Holland.

Katz, S.; and Tal, A. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM TOG*, 22(3): 954–961.

Kerbl, B.; Kopanas, G.; Leimkuehler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG*, 42(4): 1–14.

Lee, T.-C.; Kashyap, R. L.; and Chu, C.-N. 1994. Building skeleton models via 3-D medial surface axis thinning algorithms. *CVGIP: graphical models and image processing*, 56(6): 462–478.

Li, X.; Ma, Q.; Lin, T.-Y.; Chen, Y.; Jiang, C.; Liu, M.-Y.; and Xiang, D. 2025. Articulated Kinematics Distillation from Video Diffusion Models. In *CVPR*, 17571–17581.

Li, Y.; Takehara, H.; Taketomi, T.; Zheng, B.; and Nießner, M. 2021. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 12706–12716.

Li, Z.; Zheng, Z.; Wang, L.; and Liu, Y. 2024. Animatable gaussians: Learning pose-dependent gaussian maps for high-fidelity human avatar modeling. In *CVPR*, 19711–19722.

Lin, Y.; Lin, C.; Xu, J.; and MU, Y. 2025. OmniPhysGS: 3D Constitutive Gaussians for General Physics-Based Dynamics Generation. In *ICLR*.

Ling, H.; Kim, S. W.; Torralba, A.; Fidler, S.; and Kreis, K. 2024. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. In *CVPR*, 8576–8588.

Liu, I.; Su, H.; and Wang, X. 2025. Dynamic Gaussians Mesh: Consistent Mesh Reconstruction from Dynamic Scenes. In *ICLR*.

Liu, I.; Xu, Z.; Yifan, W.; Tan, H.; Xu, Z.; Wang, X.; Su, H.; and Shi, Z. 2025. Riganything: Template-free autoregressive rigging for diverse 3d assets. *ACM TOG*, 44(4): 1–12.

Liu, L.; Zheng, Y.; Tang, D.; Yuan, Y.; Fan, C.; and Zhou, K. 2019a. Neuroskinning: Automatic skin binding for production characters with deep graph networks. *ACM TOG*, 38(4): 1–12.

Liu, S.; Li, T.; Chen, W.; and Li, H. 2019b. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, 7708–7717.

Louizos, C.; Welling, M.; and Kingma, D. P. 2018. Learning Sparse Neural Networks through L_0 Regularization. In *ICLR*.

Luiten, J.; Kopanas, G.; Leibe, B.; and Ramanan, D. 2024. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 800–809.

Ma, J.; and Zhang, D. 2023. TARig: Adaptive template-aware neural rigging for humanoid characters. *Computers & Graphics*, 114: 158–167.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 405–421. Springer.

Peng, H.-Y.; Zhang, J.-P.; Guo, M.-H.; Cao, Y.-P.; and Hu, S.-M. 2024. Charactergen: Efficient 3d character generation from single images with multi-view pose canonicalization. *ACM TOG*, 43(4): 1–13.

Peng, S.; Jiang, C.; Liao, Y.; Niemeyer, M.; Pollefeys, M.; and Geiger, A. 2021. Shape as points: A differentiable poisson solver. *NeurIPS*, 34: 13032–13044.

Poole, B.; Jain, A.; Barron, J. T.; and Mildenhall, B. 2023. DreamFusion: Text-to-3D using 2D Diffusion. In *ICLR*.

Prim, R. C. 1957. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6): 1389–1401.

Song, C.; Li, X.; Yang, F.; Xu, Z.; Wei, J.; Liu, F.; Feng, J.; Lin, G.; and Zhang, J. 2025a. Puppeteer: Rig and Animate Your 3D Models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Song, C.; Zhang, J.; Li, X.; Yang, F.; Chen, Y.; Xu, Z.; Liew, J. H.; Guo, X.; Liu, F.; Feng, J.; et al. 2025b. Magicarticulate: Make your 3d models articulation-ready. In *CVPR*, 15998–16007.

Tagliasacchi, A.; Zhang, H.; and Cohen-Or, D. 2009. Curve skeleton extraction from incomplete point cloud. *ACM TOG*, 28(3): 1–9.

Wade, L. 2000. *Automated generation of control skeletons for use in animation*. The Ohio State University.

Wang, J.; Yuan, H.; Chen, D.; Zhang, Y.; Wang, X.; and Zhang, S. 2023. Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*.

Wang, R.; Mao, W.; Lu, C.; and Li, H. 2024. Towards high-quality 3d motion transfer with realistic apparel animation. In *ECCV*, 35–51.

Xu, Z.; Zhou, Y.; Kalogerakis, E.; Landreth, C.; and Singh, K. 2020. RigNet: neural rigging for articulated characters. *ACM TOG*, 39(4): 1–14.

Yan, Y.; Letscher, D.; and Ju, T. 2018. Voxel cores: Efficient, robust, and provably good approximation of 3d medial axes. *ACM TOG*, 37(4): 1–13.

Yang, Z.; Gao, X.; Zhou, W.; Jiao, S.; Zhang, Y.; and Jin, X. 2024. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 20331–20341.

Zhang, J.-P.; Pu, C.-F.; Guo, M.-H.; Cao, Y.-P.; and Hu, S.-M. 2025. One model to rig them all: Diverse skeleton rigging with unirig. *ACM TOG*, 44(4): 1–18.

Supplementary Material

The supplementary material provides rendering and implementation details, additional mechanism and robustness ablations, and further qualitative results.

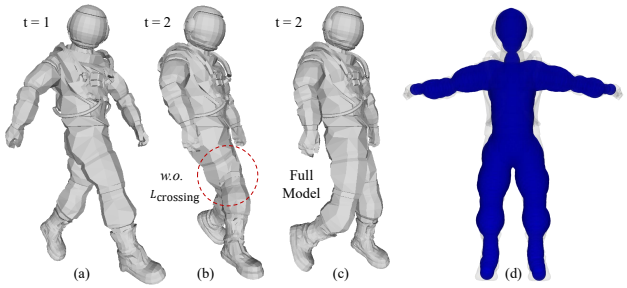


Figure S1: **Illustration of self-crossing proxy.** (a) The mesh at previous timestep. (b) The mesh at next timestep without self-crossing regularization. (c) The mesh at next timestep with self-crossing regularization. (d) The blue orbs centered on dense joints form a joint energy barrier for each mesh. A self-intersection loss will incur if the blue region intersects itself after deformation.

A Fixed Normal-Based Rendering

We use a deterministic normal-derived grayscale rendering to provide surface-orientation cues to the video diffusion model. We sample Gaussian centers on the mesh surface and associate each Gaussian with the interpolated unit surface normal $\mathbf{n}_i = [n_{ix}, n_{iy}, n_{iz}]^T$ at its sampled location. Its grayscale value is initialized as

$$s_i = \frac{n_{ix} + n_{iy} + n_{iz}}{3}, \qquad \mathbf{c}_i = (0.5 s_i + 0.5)\mathbf{1}, \quad \text{(S1)}$$

where $\mathbf{1}$ is a 3D vector of ones.

This deterministic mapping produces a neutral surface-orientation cue similar to viewport shading. It is computed once in the canonical pose and held fixed during skeleton optimization. Gaussian scale, orientation, opacity, and color are not updated by the SDS loss; Gaussian positions move only according to the optimized LBS/FK deformation.

B Additional Preliminaries

B.1 Video SDS Details

The epsilon-prediction model used by video SDS reconstructs the clean latent as

$$\hat{z} = \frac{z_t - \sqrt{1-\alpha_t}, \epsilon_\theta(z_t, t, c)}{\sqrt{\alpha_t}}, \quad \text{(S2)}$$

where z_t is the noised video latent, α_t is the diffusion noise schedule, and c is the text condition. Substituting this reconstruction into Eq. (1) produces the gradient used to optimize the rig. Gradients pass through the video encoder, the fixed normal-based renderer, forward kinematics, and linear blend skinning, while the video diffusion model remains frozen.

B.2 Hard Concrete Density

The construction in Eqs. (2) and (3) produces a mixed distribution with continuous density inside $(0, 1)$ and point masses at the endpoints. Let Q_s and q_s denote the CDF and PDF of the stretched Binary Concrete variable before clipping. The resulting Hard Concrete density is

$$q_{z_n}(z) = \begin{cases} Q_s(0 \mid x_n, T), & z = 0, \\ q_s(z \mid x_n, T), & 0 < z < 1, \\ 1 - Q_s(1 \mid x_n, T), & z = 1. \end{cases} \quad \text{(S3)}$$

The stretch parameters γ and ζ permit samples to cross both clipping boundaries, so a joint can be represented by an exact zero or one while retaining a differentiable reparameterized sample during optimization. Integrating the nonzero portion of this density gives the activation probability and expected L_0 penalty in Eq. (4).

C Self-Crossing Regularization

As mentioned in Section 4.2 of our paper, we introduced self-intersection regularization to prevent mesh self-intersection, which is quite common in generated videos of video diffusion models, not to mention in our SDS optimization case. Shown in Figure S1, we have plotted the collision volume together with meshes. The blue orbs centering at dense skeleton joints form a good proxy to the object's actual volume. When a joint enters the blue orb region of another joint and they are not close neighbors on the skeleton, a penalty will incur to prevent self-intersection. This regularization conveniently re-utilized the dense joints in our stochastic skeleton mentioned in Section 4.3 for a computationally efficient self-intersection regularization.

D Additional Ablation and Dataset Analysis

All experiments in this section use humanoid meshes from the RigXL subset. They evaluate whether the discovered skeleton depends on particular prompts, random initialization, or the canonical input pose, and whether the proposed skeleton distillation preserves animation quality.

D.1 Automatic Prompt Generation, Coverage, and Held-Out Motions

For every evaluated mesh, we submit one static rendering and the same fixed instruction to GPT-o3 before RfM optimization. The VLM returns one or more newline-separated motion sentences. Each line is used directly as one video-SDS text condition, and the number of returned lines determines P. The VLM receives no skeleton, skinning weights, animation, baseline output, or downstream RfM result. We use the first syntactically valid response and perform no semantic editing or result-based prompt selection. The exact instruction, query settings, input renderings, and per-mesh outputs are provided below and in the HTML supplement. Prompts act as motion probes that expose different articulated degrees of freedom. For the humanoid ablation, the generated motion probes are walking and punching, which respectively excite lower- and upper-body articulation. As shown in Table S2, removing either generated probe increases J2B, demonstrating

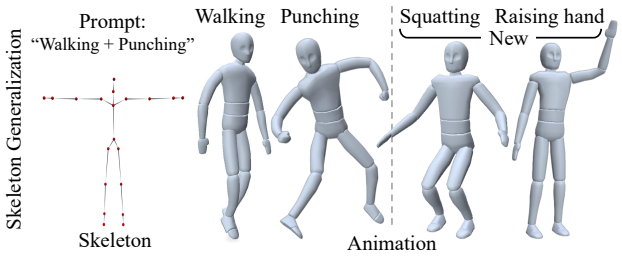


Figure S2: **Prompt coverage.** A skeleton discovered using walking and punching prompts supports held-out optimized motions, including squatting and raising the hands.

that they provide complementary motion evidence. After discovering a skeleton with both prompts, we keep that skeleton fixed and optimize held-out motions such as squatting and raising the hands. The "Held-out" column of Table S1 and Figure S2 show that the skeleton supports these motions. We use the following instruction unchanged for every mesh:

> For each image, identify every distinct articulated character and output the smallest set of motion sentences that together significantly animate every major functional body-part group. Begin with the simplest natural, species-appropriate locomotion at a normal pace. Add one complementary motion whenever locomotion leaves a major functional group insufficiently animated. For paired appendages, choose a dynamic action that moves both members through a large range, preferably in an alternating and independent pattern. When several actions qualify, prefer an alternating forward-strike action over a one-sided gesture or a motion in which the paired appendages move together. Keep motions anatomically plausible and nonredundant. Each sentence must use exactly this structure: the <character> is <verb in present-participle form>. Use a singular, generic character name. Return only the motion sentences, one per line. Prefer the same verb for characters of the same category.

D.2 Skeleton Distillation, Gating, and Robustness

Table S1 compares the final distilled skeleton with the dense candidate skeleton. Their similar SA and PC scores show that distillation preserves the recovered animation quality while removing redundant joints. Table S2 further shows that replacing the Hard Concrete gates and expected L_0 penalty with L_1 regularization substantially worsens J2B because L_1 couples joint-motion magnitude to sparsification. Removing symmetry regularization also worsens J2B, supporting its role in balanced joint selection.

Across three optimization seeds, RfM obtains a J2B of 3.53 ± 0.14, indicating robustness to random initialization. We additionally rerun the pipeline from unposed meshes sampled from existing animations. Both the rigging and animation scores remain comparable to the canonical-input setting, as shown in Table S1 and Table S2.

E Comparison with Geometry-based Methods

We additionally provide skeleton quality comparison against geometry-based methods: Pinocchio (Baran and Popović 2007) and Mesh Contraction (Au et al. 2008). RfM outperforms the two classical geometry-based baselines across all three structural metrics, shown in Table S4. These results indicate that motion priors from the video diffusion model produce skeletons that align more closely with the reference rigs than those obtained from geometry alone. Notably, Pinocchio requires a predefined humanoid skeleton template, whereas RfM discovers the skeletal structure directly from the input mesh and motion prior without relying on category-specific templates.

	Final	Dense	Unposed	Held-out
SA ↑	0.46	0.48	0.46	0.44
PC ↑	0.75	0.72	0.73	0.77

Table S1: **Animation evaluation on humanoid RigXL meshes.** "Final" uses the distilled skeleton, "Dense" uses the dense candidate skeleton, "Unposed" starts from an unposed input, and "Held-out" optimizes new motions using the fixed skeleton discovered from walking and punching.

	Ours	w.o. L_{sym}	L_1	Walk	Punch	Unposed
J2B ↓	3.53 ± 0.14	3.92	10.36	4.54	6.22	3.68

Table S2: **Skeleton evaluation on humanoid RigXL meshes, J2B reported in units of 10^{-2}.** Ours is reported as mean $\pm$ standard deviation over three optimization seeds; the remaining columns report single-run ablations. Walk and Punch use only the indicated prompt; Unposed starts from an unposed input mesh.

F User-study Protocol

We recruited 18 participants to evaluate the quality of the generated skeletons. For each input mesh, participants were shown skeletons produced by all compared methods using identical rendering settings and the same standardized viewpoints. Method identities were hidden, and the order of methods was randomized independently for each evaluation. Participants rated each result on a five-point Likert scale according to three criteria: completeness, whether the skeleton represented all visibly articulating parts of the object; conciseness, whether it avoided redundant or unnecessary joints; and correctness, whether the joint locations and connectivity were anatomically or structurally plausible for the input shape. A score of 1 indicated very poor performance and a score of 5 indicated very good performance under the corresponding criterion.

G Implementation Details

We run our method on a single A100 GPU with 40GB VRAM. We set N_{warmup} to be 2000 steps, scaling with the number of prompts, and N_{refine} to be fixed 3000 steps. During training, we jointly optimize Hard Concrete logit x_n, joints rotation sequences $\mathbf{R_e}$, and RBF skinning radius r_n. We optimize our model using Adam optimizer with learning rate of $\mathbf{R_e}$ and r_n being 0.001 and 0.2 for x_n. We linearly decay the kinematics and skinning model learning rates from 0.001

Hyperparameter	Symbol	Value
Voxel resolution	N_x, N_y, N_z	256
Gaussian smoothing standard deviation	σ	3.0
Minimum consecutive-joint spacing	$d_{\min}$	2×10^{-2}
Number of nearest joints for RBF skinning	K	3.0
Hard Concrete temperature	T_{HC}	1.0
Hard Concrete lower stretch limit	γ	-1×10^{-1}
Hard Concrete upper stretch limit	ζ	1.1
Spatial sparsity radius	r	1×10^{-1}
Junction-merging distance	d_{junction}	3×10^{-1}
Motion-smoothness weight	w_{smooth}	1×10^{4}
Symmetry weight	w_{sym}	1×10^{4}
Ground-penetration weight	w_{ground}	1×10^{5}
Gravity weight	w_{gravity}	1×10^{2}
Self-crossing weight	w_{crossing}	1×10^{5}
Sparsity weight	w_{sparse}	5.0

Table S3: **Hyperparameter values used by our pipeline.**

	Ours	Pinocchio	Mesh Contraction
J2B ↓	3.53 ± 0.14	4.98	4.99
J2J ↓	6.24 ± 0.23	7.38	7.19
B2B ↓	3.79 ± 0.11	4.66	5.04

Table S4: **Quantitative comparison with geometry-based methods on humanoid RigXL meshes.** Our method greatly surpasses the performance of the two compared geometry-based methods due to its introduction of a motion prior.

to 0.0001 during warmup phase. Their learning rates are set to a constant 0.001 during the refinement phase to allow for quick adaptation to changed skeleton topology. We use ModelScope T2V (Wang et al. 2023) to extract the SDS gradient. Each optimized video contains 20 frames, corresponding to 2 seconds at 10 frames per second. At every optimization step, we render the animated mesh at a resolution of 256×256 from three randomly sampled camera views. The cameras are placed on a sphere around the character, with a random azimuth and an elevation sampled between -20° and 5°. We use a 45° field of view and a camera distance of three to four normalized units, depending on the geometry of the input asset. Each camera remains fixed relative to the character throughout the video, preventing camera motion from being interpreted as object motion by the video prior.

We use classifier-free guidance with a guidance coefficient of 100. The negative prompt is set to "`rubber, worst quality, inconsistent motion, blurry, jittery, distorted`". At each iteration, the diffusion timestep is uniformly sampled from the middle 96% of the noise schedule, excluding the lowest and highest 2% to avoid unstable gradients at extreme noise levels. The rendered videos are normalized to $[-1, 1]$ and encoded using the frozen VAE of the video diffusion model. Both the VAE and denoising network remain frozen throughout optimization.

H Further Results

Figure S3 shows flawed GT skeletons. The frog, dinosaur, baby dragon, and dolphin lack skeleton for toe, foot, wings, and back fin respectively. Our method was unaffected by the flawed GT, whereas learning-based methods sometimes share the same flaw patterns as the ground-truth skeleton.

Figure S4, Figure S5, and Figure S6 show further qualitative comparisons of OOD meshes. We found that learning-based methods could fail for very simplistic input meshes, e.g. the mesh of a lamp, while providing detailed skeletons for in-distribution data. Our method, on the other hand, exhibits consistent robustness to general meshes.

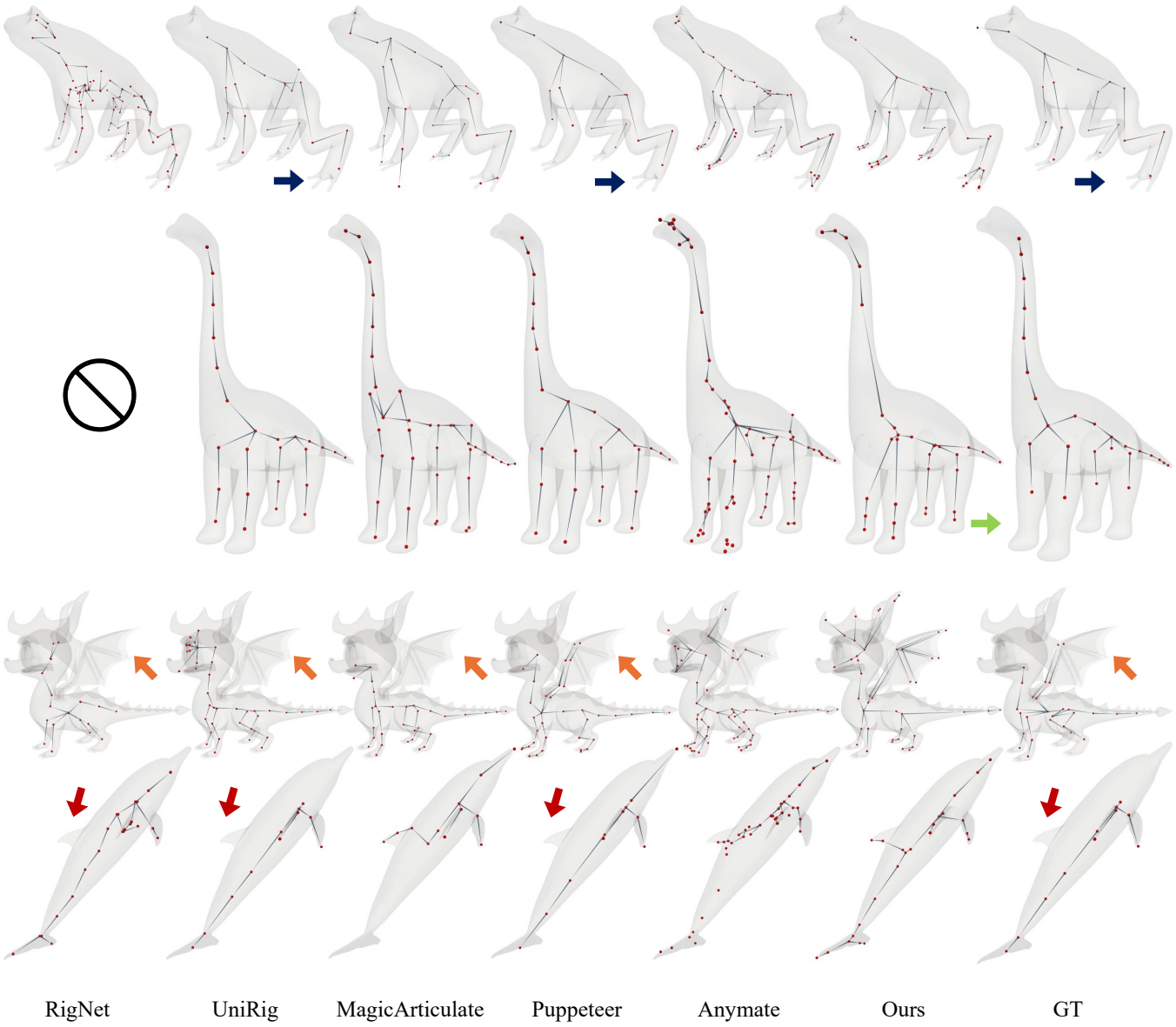


Figure S3: **Examples of imperfect ground-truth skeletons in RigXL.** Arrows identify missing or anatomically inconsistent joints and corresponding errors produced by supervised baselines. A stop symbol indicates that a method failed to produce a plausible skeleton.

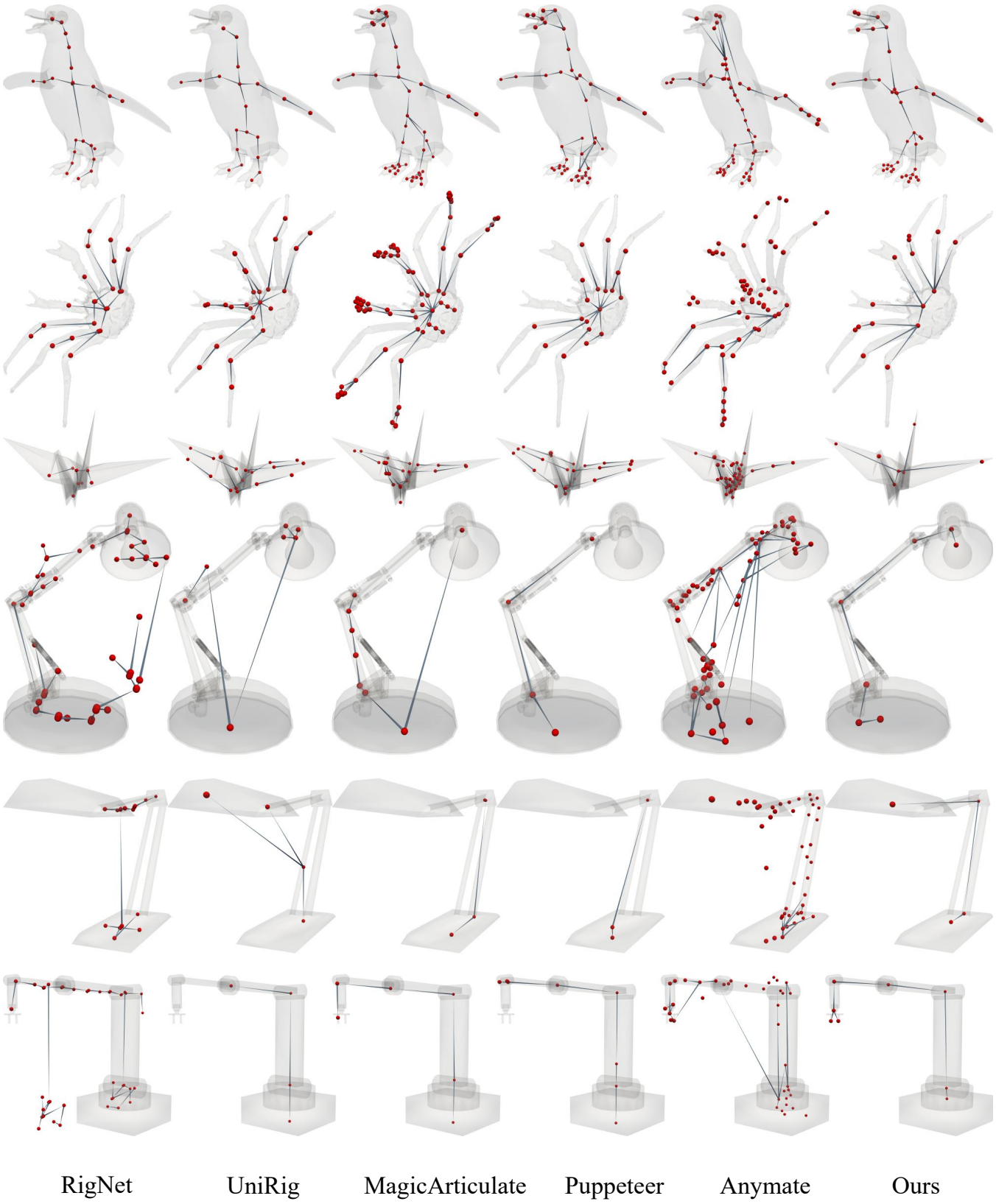


Figure S4: Further zero-shot qualitative comparison on OOD data.

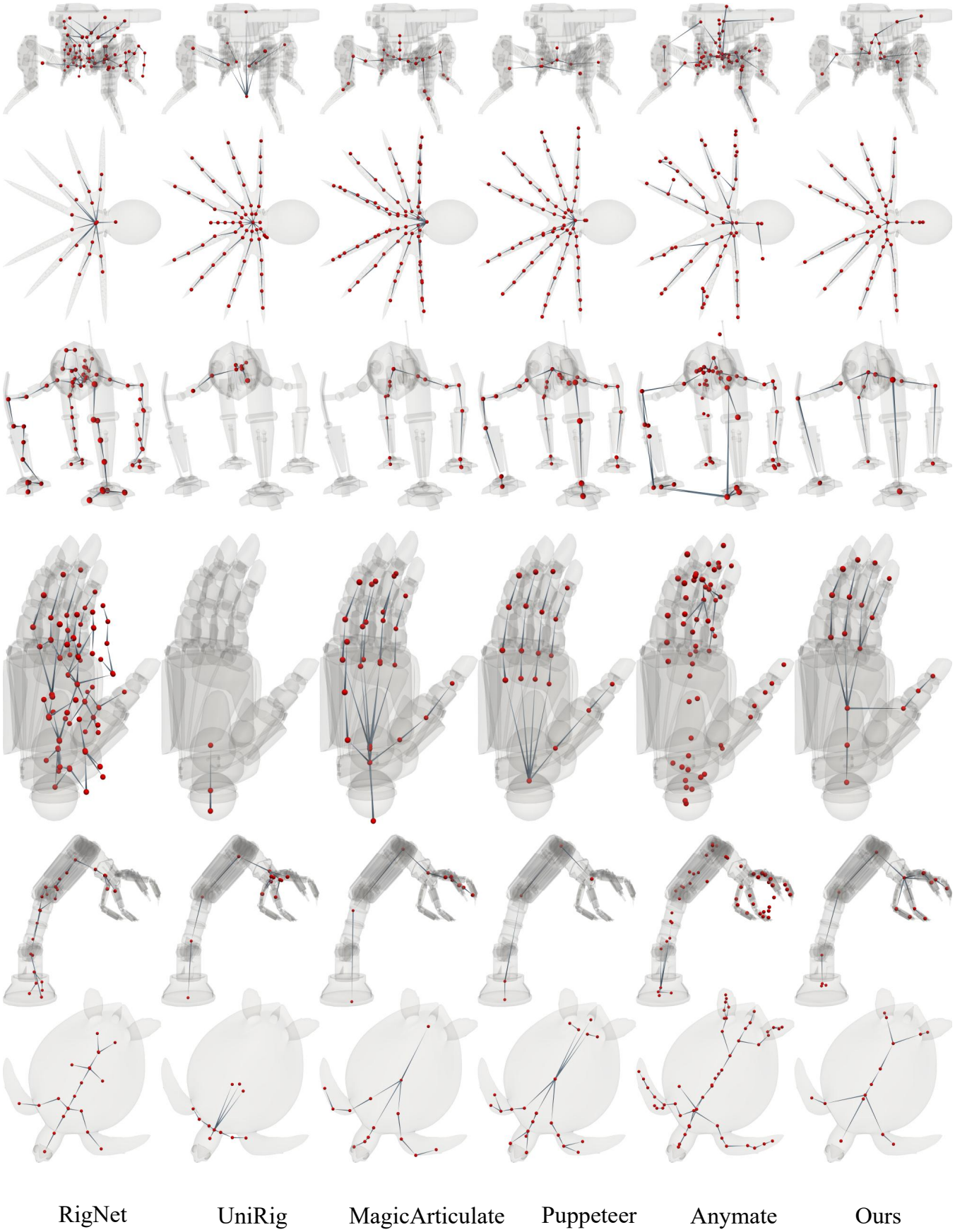


Figure S5: Further zero-shot qualitative comparison on OOD data.

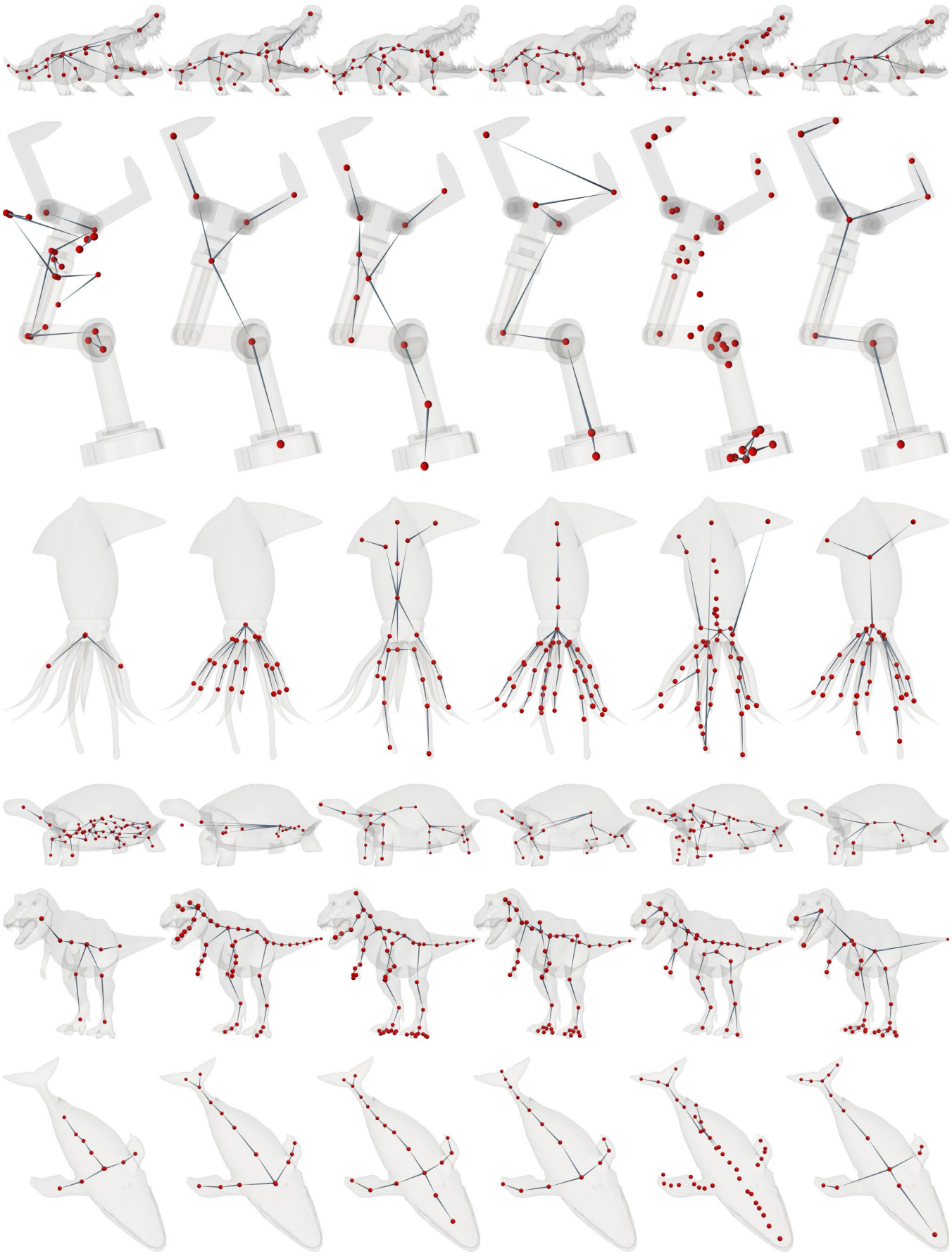


Figure S6: Further zero-shot qualitative comparison on OOD data.