ANIRIG: RECOVERING ANIMATION-READY RIGGED ASSETS FROM ARBITRARY-TOPOLOGY 3D CHARACTERS

Anonymous authors
Paper under double-blind review

ABSTRACT

High-quality character meshes are increasingly produced by generation, reconstruction, scanning, and artist workflows. Skeletal animation requires a rest mesh, a skeleton, and skinning weights that work together. Recovering this rigged asset from a single observed mesh remains a bottleneck, especially when the input is already posed. Existing rigging and skinning methods often assume rest-pose inputs or evaluate skeletons and weights separately. Component-level accuracy alone does not establish whether these predictions support coherent rest recovery and subsequent animation. We introduce AniRig, a semantic-hierarchy-conditioned framework for recovering an animation-ready rigged asset from a single observed character mesh with arbitrary topology. The hierarchy specifies joint names and parent-child connectivity, and can be user-provided or inferred from rendered mesh views by a Vision-Language Model. AniRig jointly predicts observed and rest skeletons in a single diffusion model under this hierarchy, and estimates structured sparse skinning weights by jointly evaluating a compact set of candidate joints. A differentiable pose-to-rest construction composes these predictions with the observed mesh to recover the canonical rest mesh and enables coupled fine-tuning with asset-level surface supervision. On Articulation-XL2.0, AniRig improves skeleton and skinning weight accuracy. On an independent real-motion posed-mesh benchmark, asset-level fine-tuning improves canonical rest recovery and held-out re-posing consistency. Qualitative results show that the recovered assets support skeletal animation reuse across matched hierarchies.



Figure 1: **AniRig recovers animation-ready rigged assets from single observed character meshes with arbitrary topologies.** Given a posed or non-canonical input mesh and an input semantic skeletal hierarchy, AniRig predicts a canonical rest mesh, a rest skeleton, and vertex-level skinning weights. The recovered asset can be re-posed and driven by compatible skeleton-animation curves under a matched hierarchy.

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

1 INTRODUCTION

High-quality character meshes are increasingly available from generation, reconstruction, scanning, and artist workflows (Poole et al., 2023; Sun et al., 2024; Zhang et al., 2024). However, turning a single observed mesh into a reusable rigged asset remains a major bottleneck. A standard animation asset requires a canonical rest mesh, a named rest skeleton, and vertex-level skinning weights. These quantities are not specified by surface geometry alone, particularly for arbitrary-topology characters that cannot be reduced to a fixed human template such as SMPL (Loper et al., 2015). When the input mesh is already posed or non-canonical, the observed surface further entangles the current articulation with the underlying rest-pose structure.

Automatic rigging and skinning methods have made substantial progress in predicting skeletons and surface bindings (Xu et al., 2020; Deng et al., 2025; Zhang et al., 2025a; Song et al., 2025a;b). Many assume rest-pose inputs or evaluate skeletons and skinning weights separately. Motion-aware rigging uses observations across poses (Xu et al., 2022), while recent 4D approaches animate objects using video or generative priors (Mou et al., 2025). AnimaX (Huang et al., 2025) generates skeletal motion for an already articulated mesh. For single-mesh recovery, locally plausible skeletons or skinning weights may still fail to compose into a coherent asset for canonicalization, re-posing, or skeletal animation reuse under a matched hierarchy.

A skeleton that only explains the observed pose does not provide the canonical rest structure required for animation reuse. Pairing the observed and rest skeletons establishes the correspondence between these configurations. Composing them with skinning weights lets rest and re-posed surface losses jointly refine joint placement and surface binding during fine-tuning.

We introduce AniRig, a semantic-hierarchy-conditioned framework for pose-to-rest asset recovery. The hierarchy specifies joint names and parent-child connectivity, and can be user-provided or inferred from rendered mesh views using a Vision-Language Model (VLM) (Comanici et al., 2025). Conditioned on this hierarchy, a diffusion model predicts corresponding observed and rest skeletons using shared bone tokens. Structured sparse skinning uses coverage-supervised candidate selection to retain deformation-relevant influences within a compact joint set, then predicts their support and weights through candidate interaction for rest recovery and re-posing. A differentiable pose-to-rest construction composes the predicted skeletons and skinning weights with the observed mesh to recover the canonical rest mesh. This construction preserves vertex indexing and mesh connectivity between the observed and rest meshes.

We evaluate AniRig on large-scale articulated assets and a real-motion posed-mesh benchmark. Component comparisons show improved skeleton generation and skinning weight prediction, while asset-level ablations demonstrate that fine-tuning through the differentiable pose-to-rest construction improves rest recovery and held-out re-posing consistency. Qualitative results further show that the recovered assets can reuse skeletal animation across matched hierarchies.

Our contributions are as follows:

- **Animation-ready asset recovery.** We recover a reusable rigged asset from a single observed character mesh, conditioned on joint names and parent-child connectivity. Differentiable pose-to-rest construction enables coupled fine-tuning: errors on reconstructed rest and re-posed surfaces refine skeleton placement and skinning weights together. The explicit rest mesh, rest skeleton, and normalized LBS weights support downstream editing and compatible skeletal animation reuse.
- **Paired skeleton prediction.** We jointly predict skeletons in the observed and rest poses within a single diffusion model to separate articulation from canonical structure. Each bone token couples the two configurations during denoising, preserving bone-wise correspondence without post-hoc matching. Learning the two configurations together improves joint localization and bone alignment in the observed pose.
- **Structured sparse skinning.** We formulate skinning as sparse support reasoning, using full-joint coverage supervision to train candidate selection to retain active influences. Structured interaction compares the selected joints, while separate support and weight heads distinguish influence membership from magnitude. These components improve support recovery and weight accuracy, and the compact candidate set reduces the computational and memory costs of detailed interaction.

2 Related Work

2.1 Automatic Rigging for Diverse 3D Characters

Automatic rigging predicts a skeletal structure that can articulate a character mesh. Template-fitting systems such as Pinocchio (Baran & Popović, 2007) deform a predefined skeleton to the input shape, while curve-skeleton methods extract medial structures through geometric operators (Au et al., 2008; Cao et al., 2010; Tagliasacchi et al., 2012; Huang et al., 2013). Template fitting depends on an appropriate skeletal template, whereas geometric skeleton extraction does not directly specify the named joints and hierarchy needed for animation.

Data-driven methods learn skeletal regularities from rigged assets. RigNet (Xu et al., 2020) predicts joints, connectivity, and skinning with graph features, while Point2Skeleton (Lin et al., 2021) and Neural Blend Shapes (Li et al., 2021b) learn skeletal abstractions or articulations from point clouds and deformations. Recent large-scale systems improve coverage over diverse topologies: Anymate (Deng et al., 2025) uses modular predictors; UniRig (Zhang et al., 2025a) and RigAnything (Liu et al., 2025a) tokenize skeleton structures with Transformers; MagicArticulate (Song et al., 2025b) and DRiVE (Sun et al., 2025) formulate articulation recovery as generation or denoising. ACT (Long et al., 2026) combines semantic hierarchy conditioning, diffusion-based rigging, and motion generation. Drawing on its model design and semantic hierarchy conditioning, AniRig jointly predicts observed and rest skeletons and couples them with structured sparse skinning through asset-level surface supervision for single-mesh pose-to-rest recovery.

2.2 Skinning Weight Prediction

Skinning assigns per-vertex skeletal influences for Linear Blend Skinning (LBS). Classical methods compute weights from geometric or physical priors, including bounded biharmonic weights (Jacobson et al., 2011), geodesic voxel binding (Dionne & de Lasa, 2013), and mesh-free optimization (Dodik et al., 2025). These approaches provide principled geometric baselines, but geometric proximity can differ from skeletal influence on posed meshes.

Neural methods learn surface–skeleton associations from local mesh, graph, or skeleton features (Liu et al., 2019; Pan et al., 2021; Mosella-Montoro & Ruiz-Hidalgo, 2022). Attention-based systems including Anymate, UniRig, and Puppeteer (Song et al., 2025a) model global surface–joint interactions. SkinTokens (Zhang et al., 2026) compresses skinning weights into discrete tokens and jointly models skeletons and skinning autoregressively. For dense surface–joint scorers, detailed interaction over all joints allocates computation to joints outside each vertex's sparse support. AniRig first selects candidate joints for each surface query, then models their interactions and uses separate heads to predict influence support and weight magnitude.

2.3 Pose-to-Rest Rigged Asset Recovery

Prior animatable-representation methods address different parts of asset recovery. Neural Blend Shapes learns skeletal articulation with corrective deformation, MoRig (Xu et al., 2022) uses motion observations to infer motion-aware rigs, and Puppeteer predicts a skeleton before synthesizing animation. Recent 4D approaches (Mou et al., 2025; Wu et al., 2025) produce animated representations rather than rigged assets editable through named joints and skinning weights. AnimaX (Huang et al., 2025) generates skeletal motion for an articulated input mesh, whereas AniRig recovers the rigged asset needed for subsequent animation.

Make-It-Animatable (Guo et al., 2025) recovers animation-ready humanoid assets by predicting bones, blend weights, and pose-to-rest transformations. Its models are trained for predefined humanoid hierarchies, whereas AniRig conditions on a character-specific hierarchy. For generation, AnyTop (Gat et al., 2025) introduces topology-aware tokenization and attention for arbitrary-topology character animation diffusion, while AniGen (Huang et al., 2026) jointly generates shape, skeleton, and skinning fields conditioned on an image. In contrast, AniRig focuses on single-mesh pose-to-rest rigged asset recovery for arbitrary-topology characters.

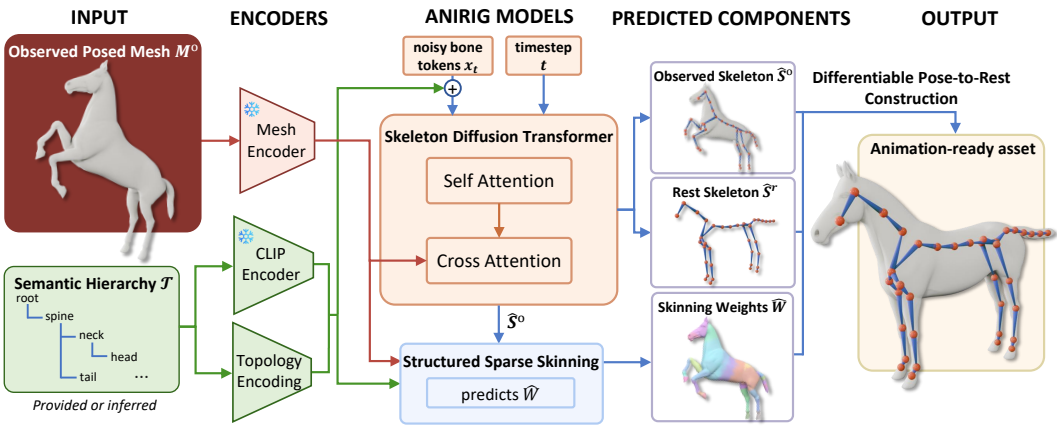


Figure 2: **Overview of AniRig.** Given an observed posed mesh and a provided or inferred semantic hierarchy, AniRig predicts the observed and rest skeletons together with structured sparse skinning weights, then composes them through a differentiable approximate pose-to-rest construction to recover a canonical rest mesh, rest skeleton, and vertex-level skinning weights.

3 METHOD

Given an observed character mesh $M^o = (V^o, F)$ and a semantic skeletal hierarchy $\mathcal{T} = (\mathcal{J}, \mathcal{B}, \eta)$, where $\mathcal{J}$, $\mathcal{B}$, and η specify joints, parent-child edges, and joint names, AniRig predicts

$$\hat{A} = (\hat{S}^o,\ \hat{S}^r,\ \hat{W},\ \hat{M}^r), \tag{1}$$

where $\hat{S}^o = (\hat{J}^o, \mathcal{B})$ and $\hat{S}^r = (\hat{J}^r, \mathcal{B})$ are the observed- and rest-pose skeletons, $\hat{W} \in \mathbb{R}^{N_v \times K}$ is the vertex-level skinning weight matrix for $N_v = |V^o|$ vertices and $K = |\mathcal{J}|$ joints, and $\hat{M}^r = (\hat{V}^r, F)$ is the recovered rest mesh. The rest target is each asset's dataset-provided rest configuration, normalized as in Sec. 4.1. Mesh connectivity F is preserved; skeletal connectivity follows $\mathcal{B}$.

The hierarchy can be user-provided or inferred from six rendered mesh views using a VLM, following ACT (Long et al., 2026). Given this structure, we predict paired skeletons, estimate skinning weights from the mesh and predicted observed skeleton, and combine both predictions through differentiable pose-to-rest construction (Fig. 2). Appendix D.1 describes hierarchy construction and editing, including user control over joint coverage and connectivity.

3.1 PAIRED SKELETON PREDICTION

Observed-pose geometry alone does not specify a rest skeleton. We therefore jointly predict the paired skeletons within a single diffusion model under the same hierarchy. Each bone token encodes endpoints in both poses. Shared joint indexing preserves correspondence without post-hoc matching, allowing the skeletons and skinning weights to be composed for rest recovery.

Drawing on ACT's model design, we use semantic hierarchy conditioning and topology-aware tokenization inspired by AnyTop (Gat et al., 2025).

A frozen TripoSG mesh encoder (Li et al., 2025) extracts surface tokens from M^o. Following endpoint-based skeleton representations (Song et al., 2025b), we encode each non-root joint i, with parent $p(i)$, by concatenating endpoints in the observed and rest poses:

$$z_i = [j^o_{p(i)},\ j^o_i,\ j^r_{p(i)},\ j^r_i] \in \mathbb{R}^{12}. \tag{2}$$

The clean sequence $x_0 \in \mathbb{R}^{K \times 12}$ stacks tokens in joint order, including a root token with zero parent endpoints. Following ACT's topological positional encoding (TPE) design, we frequency-encode the joint–parent coordinates in each bone token. Frozen CLIP joint-name embeddings (Radford et al., 2021) provide semantic conditioning, while token-index and parent-index positional encodings identify each joint and its parent in the input hierarchy. A conditional Diffusion Transformer

(DiT) (Peebles & Xie, 2023) predicts x_0 from mesh tokens, joint names, and topology. Predicted bone endpoints specify the joint positions in $\hat{S}^o$ and $\hat{S}^r$: the observed skeleton conditions skinning, and the paired skeletons define the pose-to-rest transforms. Appendix B.1 details the denoiser, conditioning, root-token handling, and joint reconstruction.

3.2 Structured Sparse Skinning Prediction

Skinning is naturally sparse: each vertex typically depends on only a few joints. We exploit this structure by selecting candidates from the full skeleton before detailed interaction and weight prediction, concentrating the scorer's computation and memory on likely influences.

Inputs and geometric priors. The module combines surface, observed-skeleton, and geometric-prior tokens. Following AnyTop's topology-aware attention and Puppeteer's topology-aware skinning formulation, the skeleton encoder combines joint-name embeddings with graph-distance attention biases, so attention reflects skeletal connectivity rather than pose-induced spatial proximity. The selector and scorer combine bone-relative distances with surface heat diffusion, voxel heat diffusion, and soft accessibility. Heat fields describe influence propagation over the surface and volume; accessibility measures occupancy along point-to-bone segments. Appendix B.3 details field construction and use during training and inference.

Candidate selection and interaction. The candidate selector scores all query–joint pairs, retaining the top-ranked joints for each query q as $\mathcal{C}_q \subseteq \mathcal{J}$, with $|\mathcal{C}_q| = \min(K_c, K)$. Queries aggregate candidate-joint features; joints receive messages from queries that select them. Candidate attention compares competing influences jointly, conditioning each score on the other candidates.

Separate prediction heads. The support head supplies soft influence-membership logits; the weight head predicts influence-strength logits. Each head is supervised separately, and their logits are fused into s_{qj} before normalization. Appendix B.2 details the architecture and fusion.

Coverage supervision and sparse normalization. An omitted influence cannot be recovered by the scorer. We train the discrete Top-K_c selector on full-joint scores: support-push raises active-joint scores, while coverage pushes the weakest active-joint score above the selection boundary. Invalid candidates are masked, and the logits are normalized only inside $\mathcal{C}_q$:

$$\hat{w}_{qj} = \begin{cases} \dfrac{\exp(s_{qj}/\tau_q)}{\sum_{k\in\mathcal{C}_q}\exp(s_{qk}/\tau_q)}, & j \in \mathcal{C}_q, \\ 0, & j \notin \mathcal{C}_q, \end{cases} \tag{3}$$

where $\tau_q > 0$ is a learned point-wise temperature. The resulting weights are sparse, non-negative, and sum to one for each query. At inference, we query every vertex to obtain $\hat{W}$.

3.3 Pose-to-Rest Mesh Construction

We recover the rest mesh with a single-pass approximate inverse-articulation operator. At inference, hierarchical alignment of corresponding child-bone directions constructs joint rotations; predicted observed and rest joint positions anchor the transform translations. We blend the resulting rigid transforms $T_j^{r\leftarrow o}$ to approximate the rest position of a homogeneous observed vertex $\bar{v}_i^o = [(v_i^o)^\top, 1]^\top$:

$$\hat{v}_i^r = \Pi\left(\sum_{j\in\mathcal{J}} \hat{w}_{ij}\, T_j^{r\leftarrow o}\bar{v}_i^o\right), \tag{4}$$

where $\Pi(\cdot)$ converts homogeneous coordinates to Euclidean coordinates. Applying Eq. 4 to all vertices yields $\hat{M}^r = (\hat{V}^r, F)$. This differentiable construction couples skeletons, weights, and recovered geometry without per-vertex iterative optimization. During fine-tuning, ground-truth observed bone directions and known observed-pose augmentation rotations guide this construction. Appendix B.4 details both settings; Appendix C.5 evaluates the approximation error.

3.4 Training and Inference

We pretrain the skeleton and skinning modules separately. Skeleton supervision encourages accurate placement and geometric consistency across paired poses (Appendix B.5). Skinning uses annotated observed skeletons, with support and weight supervision and mesh smoothness regularization. Full-joint losses train candidate selection; candidate-level losses train support and weight heads.

We jointly fine-tune the pretrained modules on training assets using asset-level surface supervision. A single denoiser evaluation at a random timestep predicts clean paired skeletons; the observed skeleton conditions skinning. We optimize the composed asset with:

$$\mathcal{L} = \mathcal{L}_{skel} + \mathcal{L}_{skin} + \lambda_{\text{rest}}\mathcal{L}_{rest} + \lambda_{\text{repose}}\mathcal{L}_{repose}. \tag{5}$$

The rest and re-posing losses compare predicted and ground-truth surface samples using MSE; target-pose rotations are sampled independently of the observed input. The surface losses update the skeleton predictor and candidate scorer through normalized candidate weights; selector losses train the selector. Appendices B.5 and B.6 provide the full objectives and training schedule.

At inference, Eq. 4 recovers the rest mesh, completing the editable LBS asset $(\hat{S}^r, \hat{W}, \hat{M}^r)$. Compatible animations use identity-rest local rotations under matched hierarchies (Appendix B.4).

4 Experiments

4.1 Experimental Setup

Datasets. All three training stages use the same Articulation-XL2.0 training split (Song et al., 2025a): over 40K annotated assets spanning humanoid and non-humanoid characters. Its approximately 2K test assets are used for component evaluation. We clean and re-annotate noisy joint names with Gemini 2.5 Flash (Comanici et al., 2025) for semantic conditioning.

For asset-level evaluation, we curate an independent real-motion benchmark of 100 humanoid and non-humanoid characters from Objaverse-XL (Deitke et al., 2023), disjoint from Articulation-XL2.0 data. We uniformly sample 10 frames per character: nine independent inputs for rest recovery and one held-out frame for re-posing. Re-posing evaluation uses the benchmark hierarchy for direct joint correspondence (benchmark details: Appendix C.1).

Training and implementation. We follow UniRig (Zhang et al., 2025a) for normalization and train on paired observed and rest poses. Both modules use a frozen TripoSG encoder (Li et al., 2025), with 8192 surface samples for skeleton prediction and 1024 queries for skinning training. The skeleton DiT uses a clean-state x_0 objective with a 1000-timestep DDPM training schedule (Ho et al., 2020) and 50-step DDIM sampling (Song et al., 2021) at inference. Skinning uses $K_c = 8$ candidate joints per query. Appendix B.6 details augmentation and training schedules.

4.2 Comparisons

We first compare skeleton and skinning predictions separately, following Puppeteer's metrics and evaluation settings. Appendix C.2 explains the comparison scope.

Skeleton generation. On the Articulation-XL2.0 test set in its original poses, we compare predicted skeletons with ground truth using J2J, J2B, and B2B distances (Xu et al., 2020), all reported in units of 10^{-2}.

We compare against five methods. RigNet predicts joints and connectivity from graph-based features. Anymate (Deng et al., 2025) separates joint and connectivity prediction. UniRig generates skeleton trees autoregressively; Puppeteer uses target-aware positional encoding for autoregressive generation. SkinTokens (Zhang et al., 2026) jointly generates skeletons and skinning tokens autoregressively. We use official implementations and available checkpoints. Topology-conditioned methods receive the dataset hierarchy; others use their native inputs. AniRig achieves the lowest joint and bone errors (Table 1). In Fig. 3, AniRig aligns joints and bones more accurately along thin, branching structures, mirroring its quantitative gains across all three distance metrics. This alignment preserves the articulated structure needed to animate these parts.

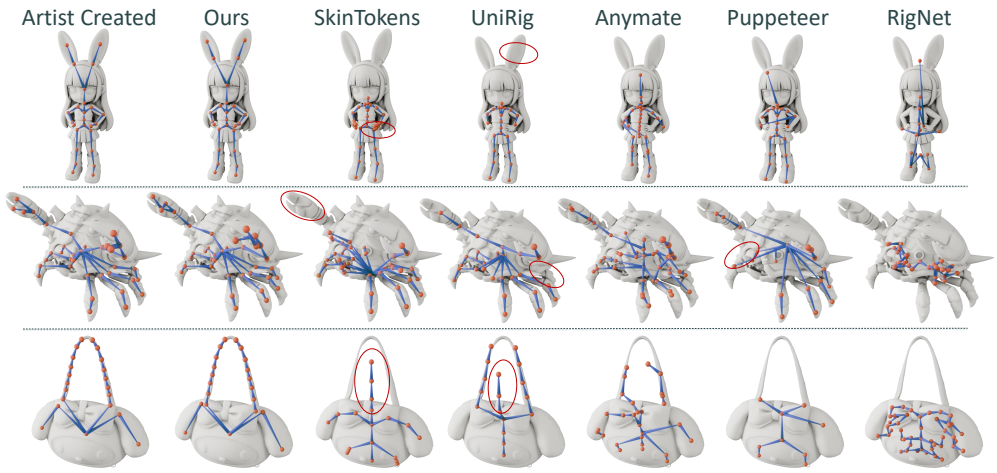


Figure 3: **Qualitative comparison of skeleton generation.** Artist-created skeletons and predictions are overlaid on the same observed meshes. Red ellipses highlight errors in joint placement and bone geometry. Across topologies, AniRig aligns predicted bones more closely with observed geometry.

Table 1: **Quantitative comparison of skeleton generation.** Joint-to-joint (J2J), joint-to-bone (J2B), and bone-to-bone (B2B) distances evaluate joint placement and bone alignment on Articulation-XL2.0. AniRig achieves the lowest errors across all three metrics.

Method	J2J ↓	J2B ↓	B2B ↓
RigNet	7.618	6.076	5.279
Anymate	4.093	2.998	2.928
UniRig	3.305	2.611	2.180
Puppeteer	3.033	2.300	1.923
SkinTokens	2.485	1.599	1.463
Ours	**1.574**	**1.143**	**0.967**

Skinning weight prediction. Skinning comparisons use Articulation-XL2.0 test meshes, with all methods conditioned on the same ground-truth skeletons, isolating weight prediction from skeleton accuracy. We report L1 Distance, Recall, and Precision.

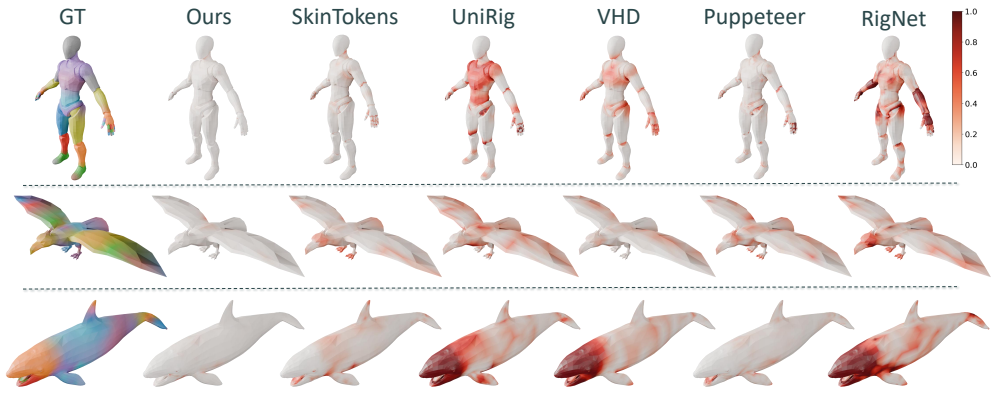


Figure 4: **Qualitative comparison of skinning weight prediction.** The first column shows GT skinning weights; the remaining columns show per-vertex errors against GT. AniRig shows lower errors around limb boundaries and thin structures across humanoid and non-humanoid examples, including closely spaced human fingers, bird feet, and shark teeth.

Table 2: **Quantitative comparison of skinning weight prediction.** L1 Distance measures weight error; Recall and Precision assess influence support. AniRig achieves the lowest L1 Distance and the highest Recall and Precision. Best results are highlighted in bold.

Method	L1 Distance ↓	Recall ↑	Precision ↑
VHD	0.526	76.9%	75.4%
RigNet	0.729	66.1%	73.7%
UniRig	0.524	70.5%	83.1%
Puppeteer	0.335	74.0%	87.6%
SkinTokens	0.293	75.5%	89.9%
Ours	**0.233**	**83.2%**	**90.0%**

We compare against Voxel Heat Diffuse Skinning (VHD), a voxel-based automatic skinning method available as a Blender add-on, and four learned methods. RigNet uses geodesic features in a graph network; UniRig uses bone–point cross-attention; Puppeteer combines topology-aware attention with part features; and SkinTokens decodes weights from discrete tokens. AniRig achieves the lowest L1 Distance and the highest Recall and Precision among compared methods, indicating more accurate weight allocation and fewer missed influences while maintaining high precision (Table 2). Fig. 4 shows lower errors around limb boundaries and thin structures across humanoid and non-humanoid examples. Human fingers illustrate this challenge: nearby surface regions require distinct joint influences, and AniRig more accurately assigns weights across these adjacent articulated parts.

4.3 Ablation Studies

Paired skeleton prediction. The observed-skeleton-only variant uses the ground-truth rest skeleton for asset-level fine-tuning. Its higher errors across all three metrics (Table 3) support learning the paired rest structure to improve both joint localization and bone alignment in the observed pose. Asset-level fine-tuning further improves these predictions through surface supervision of the composed asset, linking skeletal accuracy to the geometry it deforms.

Table 3: **Paired skeleton prediction ablation.** Lower J2J, J2B, and B2B are better.

Variant	J2J ↓	J2B ↓	B2B ↓
Ours w/o rest-skeleton prediction	1.699	1.193	1.011
Ours w/o asset-level fine-tuning	1.660	1.208	1.020
Ours	**1.574**	**1.143**	**0.967**

Structured sparse skinning. Removing coverage loss alone degrades support recovery and weight accuracy (Table 4), highlighting the importance of retaining true influences during candidate selection. Replacing structured interaction with a pointwise MLP increases error, supporting explicit interaction among competing influences. Removing heat and accessibility priors also weakens support recovery, showing the value of surface and volume cues beyond spatial proximity. Separate support and weight heads with their respective supervision improve support recovery and weight allocation over the single-head variant. Asset-level fine-tuning further refines weights through supervision of the rest and re-posed surfaces, coupling weight prediction to the resulting geometry.

Top-K_c candidate selection. We vary K_c to assess inference-time trade-offs in candidate coverage, weight accuracy, and computational cost. At $K_c = 8$, most true influences are retained, and final-weight L1 error is lowest in the sweep (Table 6, Appendix C.3). At $K_c = 4$, omitted true influences limit weight prediction. Beyond eight candidates, coverage approaches saturation: additional candidates increase final-weight Recall but lower Precision and raise L1 error while increasing computational and memory costs. On the RTX 3090, both $K_c = 24$ and dense all-joint inference run out of memory. We therefore retain $K_c = 8$ at inference (cost analysis: Appendix C.4).

Table 4: **Structured sparse skinning ablation.** OOM denotes out of memory.

Variant	L1 Distance ↓	Recall ↑	Precision ↑
Ours w/ dense all-joint scorer	OOM	OOM	OOM
Ours w/o coverage loss	0.349	75.6%	88.2%
Ours w/o candidate interaction	0.324	78.8%	89.1%
Ours w/o geometric priors	0.261	80.3%	89.2%
Ours w/o separate prediction heads	0.246	81.9%	89.5%
Ours w/o asset-level fine-tuning	0.242	82.1%	89.8%
Ours	**0.233**	**83.2%**	**90.0%**

4.4 Composed Asset Evaluation

We evaluate rest recovery and held-out re-posing using Chamfer Distance (CD) and F-Score, following 3DShape2VecSet (Zhang et al., 2023). Each recovered asset is re-posed with the held-out frame's ground-truth local rotations (Appendix C.1). Asset-level fine-tuning improves rest recovery and held-out re-posing under the benchmark hierarchy (Table 5), supporting coherent deformation of the recovered rest mesh under a new pose. With unedited VLM hierarchies, rest recovery also improves over the observed input; Appendix D.1 explains the gap to the benchmark hierarchy.

Table 5: **Rest recovery and re-posing consistency.** Inferred hierarchies are VLM-predicted and unedited. The fine-tuning ablation uses the benchmark hierarchy. Repose is reported only when direct joint correspondence is available. Lower CD and higher F-Score are better.

	Rest		Re-posing	
Variant / Protocol	CD ↓	F-Score ↑	CD ↓	F-Score ↑
Observed input	0.176	0.417	–	–
Ours (inferred hier.)	0.074	0.659	–	–
Ours (benchmark hier., w/o fine-tuning)	0.070	0.655	0.103	0.553
Ours (benchmark hier.)	**0.064**	**0.692**	**0.089**	**0.621**

Using unedited VLM-inferred hierarchies, AniRig recovers rigged rest-pose assets from in-the-wild meshes produced by 3D generative models, including both humanoid and non-humanoid characters. The examples in Appendix A (Fig. 5) show the recovered assets animated under compatible hierarchies, illustrating full-pipeline recovery across diverse character structures.

5 Conclusion

We presented AniRig, which recovers an editable rigged asset from a single observed character mesh with arbitrary topology, conditioned on a semantic hierarchy that specifies joint coverage and connectivity. Paired skeletons and structured sparse skinning are coupled through differentiable approximate pose-to-rest construction. Component comparisons show improved skeleton and skinning accuracy, while asset-level fine-tuning improves rest recovery and held-out re-posing. Qualitative results show compatible animation reuse across matched hierarchies.

Limitations and future work. Complex garments and hairstyles remain challenging for precise joint placement and localized skinning of fine secondary structures. Mechanical assemblies can require precise hinge alignment and rigid-part constraints, which our tree-structured skeleton and soft LBS formulation do not explicitly enforce. Rest recovery is limited by axial twist: endpoint-based skeletons and direction-based rotations do not recover twist that leaves skeletal geometry unchanged. Such twist can persist in the recovered mesh, while large relative rotations between influencing joints can amplify errors in approximate inverse articulation. Future work includes finer supervision for secondary structures, constraint-aware rigging, and explicit twist modeling (Appendix E).

Reproducibility Statement

We will release the source code, pretrained model checkpoints, cleaned data and annotations, data-processing scripts, and evaluation code. Section 3 describes the method, with network architectures, geometric priors, and component objectives detailed in Appendix B. Section 4 specifies the evaluation protocols; Appendices C.1 and B.6 provide the benchmark composition and training schedules. The full prompt for VLM-based hierarchy construction is in Appendix D.3.

AI Use Statement

In this work, we used Gemini 2.5 Flash for dataset preparation, specifically to clean and re-annotate joint names, and for VLM-assisted semantic hierarchy inference from six orthographic mesh views. The hierarchy construction procedure is described in Appendix D.1, with the full prompt provided in Appendix D.3. We did not use generative AI for any other tasks requiring disclosure under the ICLR 2027 AI policy. All AI-assisted joint annotations and inferred hierarchies were reviewed by the authors before use. We take responsibility for the final content of this work, including all AI-assisted artifacts.

References

Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. *ACM Transactions on Graphics*, 27(3):1–10, 2008. ISSN 0730-0301. doi: 10.1145/1360612.1360643.

Autodesk. Maya 2022 user guide: Aim constraints. Official documentation, 2022. URL `https://help.autodesk.com/cloudhelp/2022/ENU/Maya-CharacterAnimation/files/GUID-6E2297A1-D8EA-452B-80C5-A692F24CF427.htm`. Accessed September 2026.

David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. *ACM Transactions on Graphics*, 22(3):862–870, 2003. ISSN 0730-0301. doi: 10.1145/882262.882357.

Ilya Baran and Jovan Popović. Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics*, 26(3):72–es, 2007. ISSN 0730-0301. doi: 10.1145/1276377.1276467.

Junjie Cao, Andrea Tagliasacchi, Matt Olson, Hao Zhang, and Zhinxun Su. Point Cloud Skeletons via Laplacian Based Contraction. In *2010 Shape Modeling International Conference*, pp. 187–197. IEEE, 2010. doi: 10.1109/SMI.2010.25.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-XL: A universe of 10M+ 3D objects. In *Advances in Neural Information Processing Systems*, volume 36, pp. 35799–35813, 2023.

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

Olivier Dionne and Martin de Lasa. Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 173–180, 2013.

Ana Dodik, Vincent Sitzmann, Justin Solomon, and Oded Stein. Robust biharmonic skinning using geometric fields. *ACM Transactions on Graphics*, 45(2), 2025. ISSN 0730-0301. doi: 10.1145/3771928.

Inbar Gat, Sigal Raab, Guy Tevet, Yuval Reshef, Amit Haim Bermano, and Daniel Cohen-Or. AnyTop: Character animation diffusion with any topology. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–10, 2025.

Zhiyang Guo, Jinxu Xiang, Kai Ma, Wengang Zhou, Houqiang Li, and Ran Zhang. Make-It-Animatable: An efficient framework for authoring animation-ready 3D characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10783–10792, June 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Hui Huang, Shihao Wu, Daniel Cohen-Or, Minglun Gong, Hao Zhang, Guiqing Li, and Baoquan Chen. L1-medial skeleton of point cloud. *ACM Transactions on Graphics*, 32(4):1–8, 2013. ISSN 0730-0301, 1557-7368. doi: 10.1145/2461912.2461913.

Yi-Hua Huang, Zi-Xin Zou, Yuting He, Chirui Chang, Cheng-Feng Pu, Ziyi Yang, Yuan-Chen Guo, Yan-Pei Cao, and Xiaojuan Qi. AniGen: Unified S^3 fields for animatable 3D asset generation. *ACM Transactions on Graphics*, 45(4):1–16, 2026. doi: 10.1145/3811297.

Zehuan Huang, Haoran Feng, Yang-Tian Sun, Yuan-Chen Guo, Yan-Pei Cao, and Lu Sheng. AnimaX: Animating the inanimate in 3D with joint video-pose diffusion models. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–13, 2025.

Hayley Iben, Mark Meyer, Lena Petrovic, Olivier Soares, John Anderson, and Andrew Witkin. Artistic simulation of curly hair. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 63–71, 2013. doi: 10.1145/2485895.2485913.

Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics*, 30(4):78:1–78:8, 2011. ISSN 0730-0301. doi: 10.1145/2010324.1964973.

W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, 1976. doi: 10.1107/S0567739476001873.

Jiefeng Li, Chao Xu, Zhicun Chen, Siyuan Bian, Lixin Yang, and Cewu Lu. HybrIK: A hybrid analytical-neural inverse kinematics solution for 3D human pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3383–3393, 2021a.

Lin Li, Haoran Feng, Zehuan Huang, Haohua Chen, Wenbo Nie, Shaohua Hou, Keqing Fan, Pan Hu, Sheng Wang, Buyu Li, and Lu Sheng. SegviGen: Repurposing 3D generative model for part segmentation. *ACM Transactions on Graphics*, 45(4):68:1–68:12, 2026. doi: 10.1145/3811399.

Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. Learning skeletal articulations with neural blend shapes. *ACM Transactions on Graphics*, 40(4): 130:1–130:15, July 2021b. ISSN 0730-0301. doi: 10.1145/3450626.3459852.

Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, et al. TripoSG: High-fidelity 3D shape synthesis using large-scale rectified flow models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–18, 2025. doi: 10.1109/TPAMI.2025.3633512.

Cheng Lin, Changjian Li, Yuan Liu, Nenglun Chen, Yi-King Choi, and Wenping Wang. Point2Skeleton: Learning Skeletal Representations from Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4275–4284, 2021. doi: 10.1109/CVPR46437.2021.00426.

Isabella Liu, Zhan Xu, Wang Yifan, Hao Tan, Zexiang Xu, Xiaolong Wang, Hao Su, and Zifan Shi. RigAnything: Template-free autoregressive rigging for diverse 3D assets. *ACM Transactions on Graphics*, 44(4):1–12, 2025a. doi: 10.1145/3731149.

Lijuan Liu, Youyi Zheng, Di Tang, Yi Yuan, Changjie Fan, and Kun Zhou. NeuroSkinning: Automatic skin binding for production characters with deep graph networks. *ACM Transactions on Graphics*, 38(4):114:1–114:12, 2019. ISSN 0730-0301. doi: 10.1145/3306346.3322969.

Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. PartField: Learning 3D feature fields for part segmentation and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9704–9715, 2025b. doi: 10.1109/ICCV51701.2025.00905.

Pengyu Long, Weirui Wang, Qingcheng Zhao, Xiaoyang Guo, Xiaoyu Pan, Qixuan Zhang, Jiaqing Zhou, Tianlei Hu, Wei Yang, Lan Xu, and Jingyi Yu. ACT: A unified framework for rigging and animating characters with arbitrary topologies. *ACM Transactions on Graphics*, 45(4):1–20, 2026. doi: 10.1145/3811392.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: a skinned multi-person linear model. *ACM Transactions on Graphics*, 34(6), 2015. ISSN 0730-0301. doi: 10.1145/2816795.2818013.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Yanwen Guo, Zhuo Chen, and Chunchao Guo. P3-SAM: Native 3D part segmentation. In *European Conference on Computer Vision*, 2026. URL `https://eccv.ecva.net/virtual/2026/poster/4483`.

Ziqi Ma, Yisong Yue, and Georgia Gkioxari. Find any part in 3D. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7818–7827, 2025. doi: 10.1109/ICCV51701.2025.00733.

A. Milliez, R.W. Sumner, M. Gross, and B. Thomaszewski. HairControl: A tracking solution for directable hair simulation. *Computer Graphics Forum*, 37(8):115–123, 2018. doi: 10.1111/cgf.13517.

Albert Mosella-Montoro and Javier Ruiz-Hidalgo. SkinningNet: Two-Stream Graph Convolutional Neural Network for Skinning Prediction of Synthetic Characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18593–18602, 2022.

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

Xiaoyu Pan, Jiancong Huang, Jiaming Mai, He Wang, Honglin Li, Tongkui Su, Wenjun Wang, and Xiaogang Jin. HeterSkinNet: A Heterogeneous Network for Skin Weights Prediction. *Proc. ACM Comput. Graph. Interact. Tech.*, 4(1):10:1–10:19, 2021. doi: 10.1145/3451262.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *International Conference on Learning Representations*, 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.

Chaoyue Song, Xiu Li, Fan Yang, Zhongcong Xu, Jiacheng Wei, Fayao Liu, Jiashi Feng, Guosheng Lin, and Jianfeng Zhang. Puppeteer: Rig and animate your 3D models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, volume 38, pp. 72152–72184, 2025a.

Chaoyue Song, Jianfeng Zhang, Xiu Li, Fan Yang, Yiwen Chen, Zhongcong Xu, Jun Hao Liew, Xiaoyang Guo, Fayao Liu, Jiashi Feng, et al. MagicArticulate: Make your 3D models articulation-ready. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15998–16007, 2025b.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. DreamCraft3D: Hierarchical 3D generation with bootstrapped diffusion prior. In *International Conference on Learning Representations*, 2024.

Mingze Sun, Junhao Chen, Junting Dong, Yurun Chen, Xinyu Jiang, Shiwei Mao, Puhua Jiang, Jingbo Wang, Bo Dai, and Ruqi Huang. DRiVE: Diffusion-based rigging empowers generation of versatile and expressive characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21170–21180, 2025.

Andrea Tagliasacchi, Ibraheem Alhashim, Matt Olson, and Hao Zhang. Mean Curvature Skeletons. *Computer Graphics Forum*, 31(5):1735–1744, August 2012. ISSN 0167-7055, 1467-8659. doi: 10.1111/j.1467-8659.2012.03178.x.

Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2Motion: Joint analysis of motion parts and attributes from 3D shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8876–8884, 2019.

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

Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. RigNet: neural rigging for articulated characters. *ACM Transactions on Graphics*, 39(4), 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392379.

Zhan Xu, Yang Zhou, Li Yi, and Evangelos Kalogerakis. MoRig: Motion-aware rigging of character meshes from point clouds. In *SIGGRAPH Asia 2022 conference papers*, pp. 1–9, 2022.

Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y. Lam, Yan-Pei Cao, and Xihui Liu. SAMPart3D: Segment any part in 3D objects. *arXiv preprint arXiv:2411.07184*, 2024. doi: 10.48550/arXiv.2411.07184.

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*, 42(4), 2023. ISSN 0730-0301. doi: 10.1145/3592442.

Jia-Peng Zhang, Cheng-Feng Pu, Meng-Hao Guo, Yan-Pei Cao, and Shi-Min Hu. One model to rig them all: Diverse skeleton rigging with UniRig. *ACM Transactions on Graphics*, 44(4):1–18, 2025a. doi: 10.1145/3730930.

Jia-Peng Zhang, Cheng-Feng Pu, Meng-Hao Guo, Yan-Pei Cao, and Shi-Min Hu. Skin Tokens: A learned compact representation for unified autoregressive rigging. *arXiv preprint arXiv:2602.04805*, 2026.

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. CLAY: A controllable large-scale generative model for creating high-quality 3D assets. *ACM Transactions on Graphics*, 43(4):1–20, 2024. doi: 10.1145/3658146.

Longwen Zhang, Qixuan Zhang, Haoran Jiang, Yinuo Bai, Wei Yang, Lan Xu, and Jingyi Yu. BANG: Dividing 3D assets via generative exploded dynamics. *ACM Transactions on Graphics*, 44(4):62:1–62:21, 2025b. doi: 10.1145/3730840.

Appendix A presents the result gallery, and Appendix B gives model and training details. Appendix C covers evaluation protocols, baseline scope, and additional analyses, including computational and memory costs; Appendix D describes VLM-assisted hierarchy construction and provides the full prompt. Appendix E discusses limitations and future directions.

A Additional Qualitative Results



Figure 5: **Result gallery on in-the-wild generated meshes.** Columns show the input mesh, predicted observed skeleton, skinning weights, recovered rest-pose asset, and compatible animation under a matched hierarchy. Hierarchies are inferred from rendered views without manual editing. The examples span animals, stylized characters, mechanical creatures, and multi-limbed characters, illustrating complete asset recovery and animation reuse.

B METHOD AND TRAINING DETAILS

B.1 PAIRED SKELETON REPRESENTATION AND CONDITIONING

The paired sequence has one token per joint, ordered by joint index with the root at index 0. Each non-root token represents its incoming edge as in Eq. 2. The root has no incoming edge: its parent coordinates are zero placeholders, while its child coordinates encode its position in each pose. To batch characters with different joint counts, we pad the sequences and exclude padded rows from attention and losses.

The skeleton predictor is a Diffusion Transformer (DiT) (Peebles & Xie, 2023) with 12 blocks of width 512 and eight attention heads. Following ACT's topological positional encoding (TPE) design (Long et al., 2026), we frequency-encode the paired joint–parent coordinates and project the 12D endpoint representation into this feature space. Separately, frozen CLIP joint-name embeddings (Radford et al., 2021) provide semantic conditioning, while token-index and parent-index positional encodings identify each joint and its parent in the input hierarchy. Padded features are masked after each block.

A frozen TripoSG encoder (Li et al., 2025) maps sampled positions and normals to surface features, projected into the denoiser's feature space. Each block combines bone-token self-attention, surface cross-attention, and a feed-forward layer. Adaptive layer normalization conditions the updates on the diffusion timestep and pooled shape and joint-name features. The output head predicts a clean bone-token sequence encoding both poses, with shared joint indexing across the observed and rest skeletons for skinning and pose-to-rest construction.

During skeleton pretraining, x_0 is the clean paired sequence assembled from the augmented observed skeleton and its corresponding rest skeleton. We sample uniformly from the 1000 training timesteps and corrupt the sequence by

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I),$$

where β_s is the forward noise variance at step s and $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$. Mesh features, joint-name embeddings, and hierarchy encodings condition the denoiser rather than being part of the corrupted coordinate sequence. Root parent placeholders are zero in the clean target but receive noise in x_t, as do padded rows; the joint-count mask excludes padding from attention and losses. A denoiser call predicts the clean paired sequence, supervised by the losses in Appendix B.5. Asset-level fine-tuning uses the same forward-noising process on ground-truth paired skeletons and evaluates surface losses on the clean prediction from a single denoiser call at a sampled timestep (Sec. 3.4). At inference, 50-step DDIM sampling (Song et al., 2021) starts from Gaussian noise and repeatedly uses clean-sequence predictions to update the noisy state.

The root token is retained for endpoint supervision but removed, together with padded rows, before skeleton reconstruction. Real edges are ordered by child index, beginning with an edge from the root, so the remaining tokens already match the edge sequence. In each pose, the root uses the first real edge's parent-endpoint prediction; every other joint uses the child-endpoint prediction of its incoming edge.

B.2 STRUCTURED SPARSE SKINNING ARCHITECTURE

Fig. 6 expands the skinning module in Fig. 2.

Skeleton conditioning and feature representations. The four-layer skeleton encoder is conditioned on $\hat{S}^o$, joint-name embeddings, and graph distances from $\mathcal{B}$. Its geometric features describe each bone's endpoints, midpoint, direction, and length, together with root-relative coordinates. These features are frequency-encoded and combined with projected joint-name embeddings. Bucketed graph distances enter joint self-attention as learnable biases (Gat et al., 2025; Song et al., 2025a). A pooled skeleton feature also conditions the surface-query tokens, giving local influence prediction access to the character's overall skeletal structure.

For N_q surface queries and K joints, the encoders produce query features $P \in \mathbb{R}^{N_q \times d}$ and skeleton features $H \in \mathbb{R}^{K \times d}$. The selector produces scores $U \in \mathbb{R}^{N_q \times K}$, from which it gathers the candidate set $\mathcal{C}_q$ for each query. Sparse interaction and bilinear fusion produce a representation

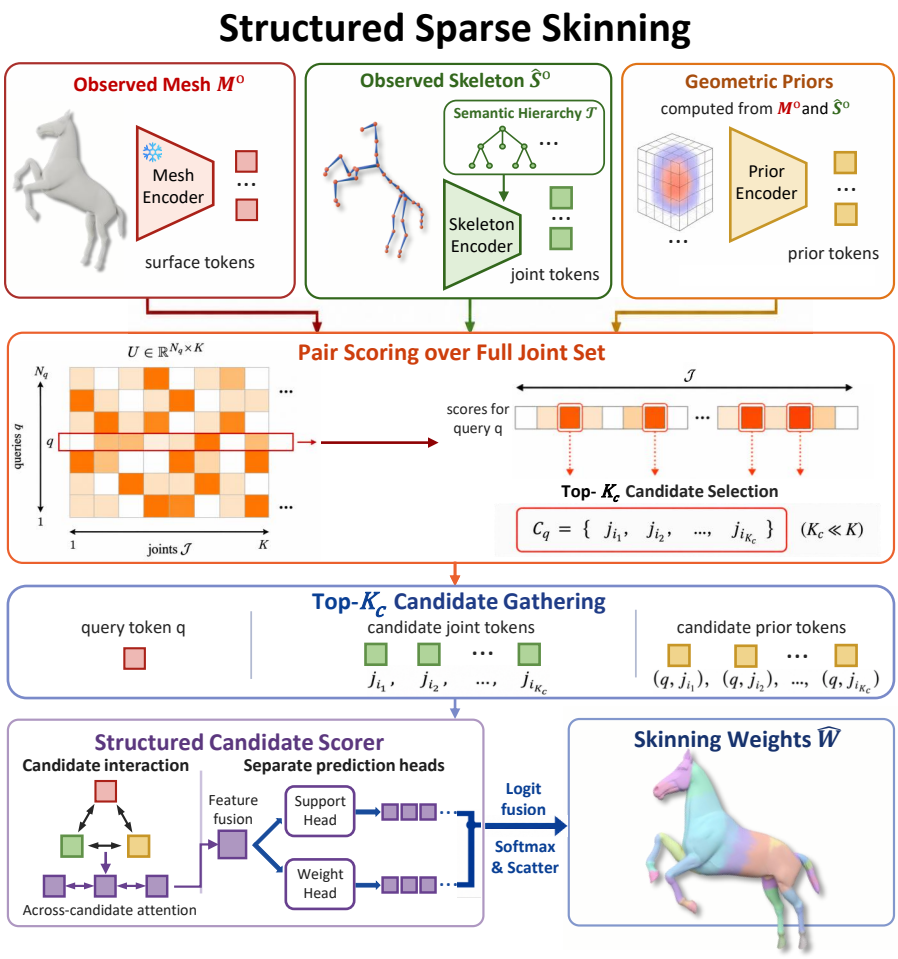


Figure 6: **Structured sparse skinning prediction.** AniRig scores query–joint pairs over the full joint set using surface tokens, topology-aware skeleton tokens, and geometric-prior tokens computed from M^o and $\hat{S}^o$. A Top-K_c selector gathers candidate influencing joints, and the candidate scorer uses separate support and weight heads within this compact set. The normalized sparse weights are scattered back to the full joint space to form $\hat{W}$.

$f_{qj} \in \mathbb{R}^d$ for each selected pair. The two prediction heads map this representation to scalar support and weight logits, denoted by a_{qj} and b_{qj}, respectively. Batch dimensions are omitted. Padded joints are excluded throughout; $\mathcal{C}_q$ contains only valid joints, retaining all of them when $K < K_c$.

Candidate selection. The candidate selector combines geometric proximity, precomputed priors, and learned surface–joint compatibility. The geometric cues include distance to the joint, distance to the bone segment, and distance falloff. Surface heat diffusion, voxel heat diffusion, and soft accessibility provide complementary point–joint fields; their values enter learned scoring branches alongside the geometric cues. The selector also compares projected surface and skeleton features through normalized feature similarity. The geometric-prior ablation zeros all three fields and removes their scoring branches, retaining distance-based and learned feature cues.

Candidate interaction. The candidate scorer uses a more detailed description of each selected pair. Alongside distance and prior values, it encodes position along the bone, surface-normal alignment, and signed coordinates in a bone-local basis. Two sparse mutual-interaction blocks exchange information between surface queries and their candidate joints. Queries aggregate candidate-joint features, joints aggregate messages from the queries that selected them, and the pair features are updated from both. A four-head candidate-attention block then compares the candidates for each query. Relative geometric cues provide attention biases and gate an additional pairwise relative-

feature message, allowing the model to compare spatial relationships among candidate joints before predicting their support and influence strength. The interaction ablation replaces both mutual-interaction blocks, candidate attention, and bilinear fusion with a pointwise MLP on concatenated features; the prediction heads and residual scoring terms remain.

Support and weight prediction. The updated query, joint, and pair features undergo rank-64 bilinear fusion, followed by support and weight heads with hidden widths 128 and 64. The support head contributes soft influence-membership logits, while Top-K_c selection restricts nonzero weights to the candidate set. The head ablation replaces the separate support and weight heads and their logit combination with a single head. The resulting skinning weights receive the final-weight supervision described in Appendix B.5. The remaining architecture and optimization schedule are unchanged.

For the full model, the fused logits and point-wise temperature are

$$s_{qj} = b_{qj} + \alpha_s a_{qj} + r_{qj}^{\mathrm{dist}} + \alpha_p r_{qj}^{\mathrm{prior}} + r_{qj}^{\cos} + r_{qj}^{\mathrm{coarse}},$$
$$\tau_q = \mathrm{softplus}\big(h_\tau\big(\mathrm{mean}_{j\in\mathcal{C}_q} f_{qj}\big)\big) + \tau_{\min}.$$

Here h_τ is a learned scalar-output MLP applied to the mean valid-candidate feature. We use $\alpha_s = 0.75$, $\alpha_p = 1.15$, and $\tau_{\min} = 0.5$. The distance and coarse residuals are MLP predictions modulated by feature-dependent sigmoid gates; their inputs include point-to-joint distance and the selector score, respectively. The prior residual encodes heat, accessibility, validity indicators, and the selector score. The cosine residual compares normalized projections of the updated query and joint features, with a learned non-negative scale. Selector scores are detached wherever they enter the scorer.

The fused logits s_{qj} and temperature τ_q determine the final weights through Eq. 3.

B.3 Geometric Prior Construction

We use three vertex–joint fields: surface heat diffusion, voxel heat diffusion, and soft accessibility. We precompute the three geometric-prior fields for each training asset in its dataset rest configuration using the annotated rest skeleton. During pose augmentation, these values remain associated with the corresponding vertices and are sampled together with the posed surface queries. This amortizes field construction across randomly generated training poses. The cached fields provide auxiliary binding cues alongside current-pose surface and skeletal features. At inference, all three fields are recomputed from the observed mesh and the skeleton supplied to the skinning module; complete-asset recovery uses the predicted observed skeleton.

We repair zero-length bones and add temporary outgoing segments at terminal joints before diffusion. Bone influences are accumulated at parent joints, temporary joints are removed, and nonzero rows are normalized across joints to give a relative influence distribution for each vertex.

Soft accessibility measures occupancy along the segment from a vertex to its nearest point on a joint's incoming bone; the root uses its joint position. Gaussian-weighted samples along this segment aggregate filled-volume and surface-shell occupancy into I_{qj} and S_{qj}. We define

$$A_{qj} = \exp(-\alpha_I I_{qj} - \alpha_S S_{qj}),$$

with $\alpha_I = 4$ and $\alpha_S = 1.5$. The three fields are sampled alongside surface queries and supplied to the candidate selector and scorer. Column j in all three fields corresponds to joint j, as in W: heat aggregates outgoing-segment influences at that joint, whereas accessibility describes its incoming segment.

For prior generation, meshes are centered at their bounding-box centers and normalized to $[-1, 1]$. Both heat solvers use resolution 128, five diffusion loops, 64 samples, four influences, falloff 0.2, and sharpness 3, with solidification disabled. Accessibility uses voxel pitch equal to the bounding-box diagonal divided by 192 and a one-voxel dilated surface shell. We sample each segment at 20 evenly spaced positions from 0.08 to 0.92 of its length, using normalized Gaussian weights with mean 0.5 and standard deviation 0.22.

B.4 Joint Orientations and Animation Transforms

Inference-time direction alignment. We estimate rotations from corresponding outgoing bones in the predicted observed and rest skeletons. For joint j and child c, let $d_{jc}^s = (\hat{j}_c^s - \hat{j}_j^s)/\|\hat{j}_c^s - \hat{j}_j^s\|_2$,

where $s \in \{o, r\}$; pairs with a numerically zero length are excluded. We traverse the hierarchy from the root. Given the parent's accumulated rotation $R_{p(j)}$, we align $R_{p(j)} d^o_{jc}$ to d^r_{jc} with a residual rotation ΔR_j, then set $R_j = \Delta R_j R_{p(j)}$. The root uses identity as its inherited rotation.

With multiple valid child directions, we minimize their summed squared alignment error with equal weight, using an SVD-based rotation fit (Kabsch, 1976) with determinant $+1$. With one direction, we use the minimum-angle Rodrigues rotation. For antiparallel directions, its 180° axis is the normalized projection of the world x-axis onto the plane orthogonal to the source direction; we use the world y-axis instead when the absolute dot product with the x-axis exceeds 0.9. A joint with no valid outgoing directions, including a leaf, inherits its parent's rotation; a root with no valid directions uses identity.

Position-anchored transforms. The global rotations R_j and predicted joint positions define rigid transforms:

$$T_j^{r \leftarrow o} = \begin{bmatrix} R_j & \hat{j}_j^r - R_j \hat{j}_j^o \\ \mathbf{0}^\top & 1 \end{bmatrix}.$$

Each transform maps its predicted observed joint to its predicted rest joint without bone scaling. Eq. 4 blends the transforms using predicted skinning weights.

Training-time orientation guidance. During asset-level fine-tuning, we first align predicted observed directions to the ground-truth observed skeleton using the same direction-alignment rule, giving C_j. Let R_j^{aug} be the global rest-to-observed rotation obtained by accumulating the known observed-pose augmentation rotations through forward kinematics (FK). We use $R_j = (R_j^{\text{aug}})^\top C_j$ in the transform above. Predicted observed and rest joint positions and skinning weights remain in surface reconstruction.

Re-posing the recovered asset. We use the identity-rest convention of pose augmentation: all rest joint frames are aligned with the mesh coordinate axes. For non-root joint j, the local bind translation is $\hat{\delta}_j = \hat{j}_j^r - \hat{j}_{p(j)}^r$; the root uses $\hat{\delta}_0 = \hat{j}_0^r$. Let $Q_j^u \in \mathrm{SO}(3)$ denote the target local rotation in this convention. The predicted skeleton's global bind and posed transforms are

$$\hat{G}_j^r = \begin{bmatrix} I & \hat{j}_j^r \\ \mathbf{0}^\top & 1 \end{bmatrix}, \qquad \hat{G}_j^u = \hat{G}_{p(j)}^u \begin{bmatrix} Q_j^u & \hat{\delta}_j \\ \mathbf{0}^\top & 1 \end{bmatrix}.$$

The root's parent transform is identity, so its position remains $\hat{j}_0^r$ during re-posing. LBS uses $T_j^{u \leftarrow r} = \hat{G}_j^u (\hat{G}_j^r)^{-1}$ to deform the recovered rest surface with the predicted skinning weights.

B.5 Training Objectives

We detail the component losses appearing in Eq. 5.

Skeleton prediction. The clean paired sequence is supervised by endpoint reconstruction, shared-joint consistency, and bone-length losses. For pose $s \in \{o, r\}$, let $e_i^s = [a_i^s, c_i^s] \in \mathbb{R}^6$ denote the parent and child coordinates in token i. Let $\hat{j}_i^s$ be the joint coordinate reconstructed by the decoder in Appendix B.1. For each non-root joint, the predicted bone length is $\hat{\ell}_i^s = \|\hat{j}_i^s - \hat{j}_{p(i)}^s\|_2$, and $\ell_i^s = \|a_i^s - c_i^s\|_2$ is its ground-truth length. The objective is

$$\begin{aligned} \mathcal{L}_{\text{skel}} = \lambda_e \sum_{s \in \{o,r\}} \mathrm{MSE}(\hat{e}^s, e^s) \\ + \lambda_b \sum_{s \in \{o,r\}} \mathrm{MSE}(\hat{\ell}^s, \ell^s) + \lambda_p \,\mathrm{MSE}(\hat{\ell}^o, \hat{\ell}^r) \\ + \lambda_{\text{conn}} \mathcal{L}_{\text{conn}}. \end{aligned}$$

The hierarchy fixes parent–child connectivity. The length losses penalize deviations from ground-truth bone lengths in each pose and encourage consistency between the paired predictions. We set $\lambda_e = 10$ and $\lambda_b = \lambda_p = \lambda_{\text{conn}} = 20$. Endpoint MSE averages over all coordinates of valid tokens, including the root and its zero parent placeholders. All three bone-length MSE terms average over real bones only, excluding the root token. Padding is excluded from every term.

Shared-joint consistency. A joint can appear in several endpoint tokens. We align these repeated predictions with the decoded coordinates separately in each pose. For an asset with K valid joints and root index zero, we use

$$\mathcal{L}_{\text{conn}} = \frac{1}{2K} \sum_{s\in\{o,r\}} \left[\sum_{i=1}^{K-1} \|\hat{a}_i^s - \hat{j}_{p(i)}^s\|_2^2 + \|\hat{c}_0^s - \hat{j}_0^s\|_2^2 \right].$$

This term is averaged across assets, with gradients through both sides. The root term aligns its token's child endpoint with the decoded root from the first real edge; root parent placeholders and padding are excluded. Bone-length supervision acts directly on the decoded skeletons used by skinning and pose-to-rest construction.

Skinning prediction. During component training, skinning receives annotated observed skeletons. Surface-query positions, normals, target skinning weights, and prior values are sampled together on the mesh, keeping the geometric inputs and supervision spatially aligned.

The loss supervises weights, selection, and candidate heads, with mesh smoothness regularization:

$$\begin{aligned}\mathcal{L}_{\text{skin}} = {} & \lambda_1\mathcal{L}_1 + \lambda_{\text{root}}\mathcal{L}_{\text{root}} + \lambda_{\text{Dice}}\mathcal{L}_{\text{Dice}} \\ & + \lambda_{\text{inactive}}\mathcal{L}_{\text{inactive}} + \lambda_{\text{push}}\mathcal{L}_{\text{push}} + \lambda_{\text{cov}}\mathcal{L}_{\text{cov}} \\ & + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}} + \lambda_s\mathcal{L}_{\text{support}} + \lambda_w\mathcal{L}_{\text{weight}} + \lambda_\tau\mathcal{L}_{\text{temp}}.\end{aligned}$$

The loss weights are $\lambda_1 = 20$, $\lambda_{\text{root}} = 5$, $\lambda_{\text{Dice}} = 3$, $\lambda_{\text{inactive}} = 6$, $\lambda_{\text{push}} = 1.5$, $\lambda_{\text{cov}} = 4$, $\lambda_{\text{smooth}} = 1.5$, $\lambda_s = 1$, $\lambda_w = 0.5$, and $\lambda_\tau = 0.02$.

Final-weight supervision. We use an active-support threshold of $\epsilon = 10^{-4}$. $\mathcal{L}_1$ is mean absolute weight error over valid joints and queries, and $\mathcal{L}_{\text{root}}$ restricts that error to root influences. We use the Dice discrepancy $D(x, y) = 1 - (2x^\top y + \epsilon)/(\|x\|_2^2 + \|y\|_2^2 + \epsilon)$. Full-field Dice compares the predicted and target weight vectors across queries for each joint, then averages over joints. Inactive-support suppression averages $[\hat{w}_{qj} - 0.05]_+$ where $w_{qj} \le \epsilon$. Here $[x]_+ = \max(x, 0)$ denotes the positive part.

Selector supervision. The candidate scorer operates only on $\mathcal{C}_q$, so it cannot recover influences discarded during selection. Support-push raises scores of active joints, while coverage places the weakest active joint above the selection boundary. Let u_{qj} denote the corresponding entry of the selector score matrix U, and $\mathcal{S}_q = \{j : w_{qj} > \epsilon\}$ the target support. $\mathcal{L}_{\text{push}}$ averages $\text{softplus}(-u_{qj})$ over active query–joint pairs. For $m_q = |\mathcal{S}_q|$, let ν_q be the $(K_c - m_q + 1)$-th highest non-support score, using the masked value -10^4 if fewer valid non-support joints exist. Then $\mathcal{L}_{\text{cov}}$ averages $[\delta + \nu_q - \min_{j\in\mathcal{S}_q} u_{qj}]_+$ over queries with $0 < m_q \le K_c$, with margin $\delta = 0.05$. This encourages retaining active influences while allowing unused candidate slots to contain non-support joints. The Top-K_c indices are non-differentiable, so these full-joint losses train the selector directly. Candidate losses train the scorer and shared surface projection and skeleton encoder, without gradients through the selected indices or detached selector scores. TripoSG remains frozen. The coverage ablation removes only $\mathcal{L}_{\text{cov}}$, retaining $\mathcal{L}_{\text{push}}$ and all other losses.

Candidate heads and smoothness. For candidate supervision, set $y_{qj} = \mathbf{1}[w_{qj} > \epsilon]$ and, for sets with target mass greater than ϵ, normalize the target weights within the selected set as $\tilde{w}_{qj} = w_{qj} / \sum_{k\in\mathcal{C}_q} w_{qk}$. The head losses for the full model are

$$\begin{aligned}\mathcal{L}_{\text{support}} &= \text{BCE}(a, y) + \gamma_D \left\langle D(\sigma(a_q), y_q) \right\rangle_q, \\ \mathcal{L}_{\text{weight}} &= -\left\langle \sum_{j\in\mathcal{C}_q} \tilde{w}_{qj} \log \text{softmax}_{\mathcal{C}_q}(b_q)_j \right\rangle_q, \\ \mathcal{L}_{\text{temp}} &= \left\langle (\log \tau_q)^2 \right\rangle_q.\end{aligned}$$

Here $\gamma_D = 0.5$, σ is the sigmoid, and BCE is binary cross-entropy with logits over valid candidate entries. Support Dice is computed per query over valid candidates. The weight loss includes only queries with selected target mass above ϵ; other query-wise averages exclude empty candidate

sets. The weight-head softmax uses b_{qj} alone; final weight supervision uses the fused logits and temperature from Eq. 3.

For smoothness, we group surface queries by their sampled face indices. Faces are sampled in proportion to area, with uniform sampling within each triangle; face means average the queries assigned to that face. Adjacent-face supervision uses mesh-adjacency pairs only when both faces contain sampled queries, without additional neighbor sampling. Mesh-topology-aware smoothness combines squared deviations from detached within-face mean weights with twice the mean absolute difference between adjacent-face means. Each adjacent-face difference is weighted by $\exp[-(\mu_f^{\mathrm{gt}} - \mu_g^{\mathrm{gt}})^2/0.05]$, applied per joint, where μ^{gt} is the mean target weight on a sampled face. This reduces smoothing across influence boundaries.

Asset-level fine-tuning. We independently sample two sets of local joint rotations with the same pose-augmentation procedure: one generates the observed input and the other the re-posing target. We use 8192 corresponding surface samples across the rest, observed, and augmented target poses. Eq. 4 maps observed samples to $\hat{P}^r$ using predicted skeletons and sample weights, with the training-time orientation guidance in Appendix B.4. To obtain $\hat{P}^u$, we apply the target-pose local joint rotations to the predicted rest skeleton through FK under the same rest-frame convention, then deform $\hat{P}^r$ by LBS with the predicted sample weights. Ground-truth targets use the same target-pose rotations with annotated skeletons and weights. Sample indices remain matched across these operations. For ground-truth rest and target samples P^r and P^u, we define $\mathcal{L}_{rest} = \mathrm{MSE}(\hat{P}^r, P^r)$ and $\mathcal{L}_{repose} = \mathrm{MSE}(\hat{P}^u, P^u)$. Both MSE terms average over points and coordinates, with $\lambda_{\mathrm{rest}} = \lambda_{\mathrm{repose}} = 0.1$.

B.6 Training Schedule

Both modules use 512-dimensional features. Pose augmentation randomly rotates each joint by at most 45° and applies FK and LBS to the rest mesh; vertices and joints are normalized to $[-1, 1]$. Using four NVIDIA L20 GPUs, the stages in Sec. 3.4 take 7 days for skeleton training (global batch size 1024), 6 days for skinning (batch size 64), and 1 day for asset-level fine-tuning (batch size 32). All three stages use the same training split described in Sec. 4.1. We optimize all stages with AdamW (Loshchilov & Hutter, 2019), using an initial learning rate of 10^{-4} and cosine decay (Loshchilov & Hutter, 2017) to 10^{-5}.

C Evaluation Details and Additional Analyses

C.1 Benchmark and Evaluation

The real-motion benchmark, curated from Objaverse-XL (Deitke et al., 2023) (Sec. 4.1), provides rest meshes, rest skeletons, skinning weights, local joint rotations, and observed skeletons. Its 100 test characters have 20–50 joints and animation clips longer than 30 frames, comprising 40% humanoids, 25% animals, 20% fantasy creatures, 5% mechanical characters, and 10% other types. All are disjoint from Articulation-XL2.0 data, and excluded from model selection. Sampling and evaluation follow Secs. 4.1 and 4.4. The animation data have global root translation removed, so evaluation measures articulation without a global motion trajectory. We average results over assets recovered independently from nine frames per character. For Table 5, the ground-truth local rotations are converted to the identity-rest convention in Appendix B.4 before being applied to the predicted rest skeleton through FK and LBS. Predicted and ground-truth surfaces share the observed mesh's normalization scale. CD sums the two directional means of unsquared Euclidean nearest-neighbor distances; F-Score is the harmonic mean of surface precision and recall. Both metrics use $50{,}000$ sampled surface points, with an F-Score threshold of $\tau = 0.02$.

For skinning evaluation, we retain points whose ground-truth weights sum to one within 10^{-2}. L1 Distance sums absolute weight errors over joints and averages over valid points. Precision and Recall pool active point–joint entries within each test case, using a 10^{-4} weight threshold. Each metric is then averaged equally across test cases.

C.2 Baseline Selection and Comparison Scope

Skeleton comparisons use each method's native conditioning. Skinning comparisons condition every method's weight-prediction component on the same ground-truth dataset skeleton. AniRig's semantic hierarchy specifies joint coverage and connectivity, giving users direct control over the target rig and animation hierarchy. ACT (Long et al., 2026) is not included in the quantitative comparisons because its code is not publicly available.

Classical rigging and skinning. Template-based rigging methods such as Pinocchio (Baran & Popović, 2007) fit predefined geometric skeletons to meshes, requiring compatible templates and a selection protocol for heterogeneous characters. Our skeleton comparisons focus on learned predictors, including RigNet (Xu et al., 2020), that, like AniRig, recover character-specific skeletons without predefined geometric templates. Each method retains its native conditioning.

For skinning, classical approaches compute binding weights from geometric distances, diffusion, or smoothness-based optimization (Baran & Popović, 2007; Jacobson et al., 2011; Dionne & de Lasa, 2013). As a practical non-learning baseline, we include Voxel Heat Diffuse Skinning (VHD), a voxel-based automatic skinning method available as a Blender add-on. All methods receive the same ground-truth dataset skeletons, so the comparison isolates skinning-weight prediction from skeleton accuracy. As shown in Table 2, AniRig achieves lower L1 Distance and higher Recall and Precision than VHD, showing that its gains extend beyond comparisons with learned predictors.

Component and complete-asset evaluation. AniRig recovers complete rigged assets across humanoid and non-humanoid characters using a single trained model conditioned on character-specific joint counts, semantic joint names, and parent–child connectivity, without topology-specific fine-tuning. Its jointly predicted observed and rest skeletons establish the correspondence needed to compose skeleton predictions and skinning weights into a recovered rest mesh through Eq. 4. UniRig (Zhang et al., 2025a), Puppeteer (Song et al., 2025a), and SkinTokens (Zhang et al., 2026) predict input-pose rigs; evaluating their outputs for rest recovery would require an additional canonicalization stage to supply the rest skeleton and its correspondence. Make-It-Animatable (Guo et al., 2025) performs pose-to-rest recovery with a predefined kinematic tree and extends its humanoid models to additional predefined bone configurations through fine-tuning. A comparison across the benchmark's character-specific hierarchies would therefore require topology-specific adaptation, while evaluation restricted to its supported humanoid configurations would cover only a subset of the single-model, variable-hierarchy setting studied here. We therefore compare shared skeleton and skinning outputs in Sec. 4.2, and evaluate complete-asset recovery against ground truth across the heterogeneous benchmark in Sec. 4.4 (Table 5). Rest recovery measures canonical geometry; held-out re-posing evaluates whether the recovered rest mesh, skeleton, and weights jointly reproduce the target surface under the held-out frame's ground-truth local rotations. The asset-level fine-tuning ablation quantifies its effect on both rest recovery and re-posing consistency.

C.3 Candidate-Set Size

We vary K_c at inference using a fixed checkpoint trained with $K_c = 8$ and evaluate on the Articulation-XL2.0 test set with 8192 surface queries per sample. Candidate Recall measures the fraction of ground-truth influences retained per query; All-Hit measures whether all are retained, using a 10^{-4} weight threshold. Both average over valid queries with non-empty ground-truth support, then equally across test cases. Final-weight metrics follow Appendix C.1.

Candidate-scorer time is measured using CUDA events, excluding warmup, and is part of the model-forward time. Memory is the peak allocated by PyTorch over the entire run, including model loading, warmup, inference, and GPU postprocessing.

Table 6a distinguishes selection coverage from final-weight quality. All-Hit checks whether the entire target support survives pruning, making it more sensitive to omitted influences than average Candidate Recall. At $K_c = 4$, the smaller candidate set omits true influences that cannot be recovered during final weight prediction. Beyond $K_c = 8$, additional candidates increase final-weight Recall but reduce Precision and raise L1 error as coverage saturates. Retaining more true influences therefore does not by itself ensure more accurate weight assignment.

Table 6: **Effect of candidate-set size.** (a) Candidate coverage and final-weight quality. (b) Model-forward time, candidate-scorer time, and peak allocated memory on an RTX 3090. $K_c = 8$ is the default. OOM denotes out of memory. Best results are highlighted in bold.

(a) Coverage and weight quality

K_c	Candidate Recall ↑	All-Hit ↑	L1 Distance ↓	Recall ↑	Precision ↑
4	93.98%	81.78%	0.274	76.0%	**91.0**%
8	99.38%	97.89%	**0.233**	83.2%	90.0%
12	99.81%	99.38%	0.237	85.6%	88.7%
16	99.93%	99.76%	0.246	86.9%	87.4%
20	**99.97**%	**99.91**%	0.256	**87.6**%	86.1%
24	OOM	OOM	OOM	OOM	OOM

(b) Measured computational and memory costs

K_c	Model forward (ms) ↓	Candidate scorer (ms) ↓	Peak allocated memory (GiB) ↓
4	**283.6**	**68.2**	**1.91**
8	377.1	140.2	3.78
12	463.6	217.1	6.25
16	468.0	257.5	10.08
20	591.6	356.4	15.05
24	OOM	OOM	OOM

The additional candidates also increase candidate-scorer cost (Table 6b). Relative to eight, twenty candidates require $1.57\times$ the model-forward time and $3.98\times$ the peak allocated memory, while adding only 0.59 percentage points of Candidate Recall. The scorer accounts for an increasing share of forward time, consistent with the cost of processing more candidate pairs. The $K_c = 24$ run fails with OOM. These results support retaining $K_c = 8$ at inference to balance weight accuracy and computational efficiency.

C.4 Theoretical Computational and Memory Costs

For a reference workload of $N_q = 8192$ and $d = 512$, the same scorer's linear layers, attention, and weighted aggregation require 1069.7 versus 12690.0 GMACs for 8 versus 100 candidates, a $11.9\times$ reduction. Candidate features have $N_q K_c d$ elements and explicit pairwise relative-feature tensors have $N_q K_c^2 d$ elements, giving storage reductions of $100/8 = 12.5$ and $(100/8)^2 = 156.25$, respectively. In FP32, a single pairwise relative-feature tensor occupies 1.0 versus 156.25 GiB at these two candidate counts. Encoding and full-joint selection retain their shared costs. Linear per-candidate projection and fusion account for much of the arithmetic, while pairwise relative-feature storage grows quadratically, making large candidate sets particularly memory-intensive.

C.5 Inverse-Articulation Error

To isolate the approximation error of Eq. 4, we randomly sample 100 characters from the Articulation-XL2.0 training split and generate poses with random joint rotations. Forward LBS deforms the original rest-mesh vertices using ground-truth transforms and skinning weights; the inverse operator then uses the same weights and the inverses of the same joint transforms to recover their rest positions. This round-trip test complements the complete-asset evaluation in Table 5.

We measure Euclidean distance between corresponding recovered and original rest vertices, expressed as a percentage of each character's fixed rest-pose bounding-box diagonal. For each group, we average over its vertices within each frame, then over frames within each character, and finally equally across characters represented in that group. Grouping uses the number of active ground-truth

influences or their maximum pairwise relative rotation, measured between the rotation components of the rest-to-observed joint transforms.

Table 7: **Inverse-articulation error on 100 randomly selected training characters.** Mean normalized vertex error is grouped by active influence count and maximum pairwise relative rotation between influencing joints.

Grouping	Group	Error (%)
Active influences	1	0.000
	2	2.222
	≥ 3	3.405
Maximum relative rotation	0–30°	0.326
	30–45°	0.860
	45–60°	1.485
	$> 60°$	3.614

With a single influence, Eq. 4 reduces to a rigid inverse, consistent with the zero error reported in Table 7. With multiple influences, blending inverse transforms generally differs from inverting their blend. The larger errors in the higher relative-rotation groups reflect the difficulty of undoing a blend of differing joint motions.

D VLM-Assisted Hierarchy Construction

D.1 Construction and Practical Use

Following ACT (Long et al., 2026), we use Gemini 2.5 Flash (Comanici et al., 2025) to infer joint names and parent–child relations from six orthographic views. The prompt in Appendix D.3 requests a compact hierarchy of major articulated structures. Fig. 7 pairs these visual inputs with a generated hierarchy for a quadruped. Whereas RigAnything (Liu et al., 2025a) and SkinTokens (Zhang et al., 2026) predict topology from meshes, AniRig accepts an editable hierarchy. Users can add, remove, or reconnect finger, hair, or other joints; the observed and rest skeletons both follow the edited hierarchy, retaining correspondence for pose-to-rest construction.

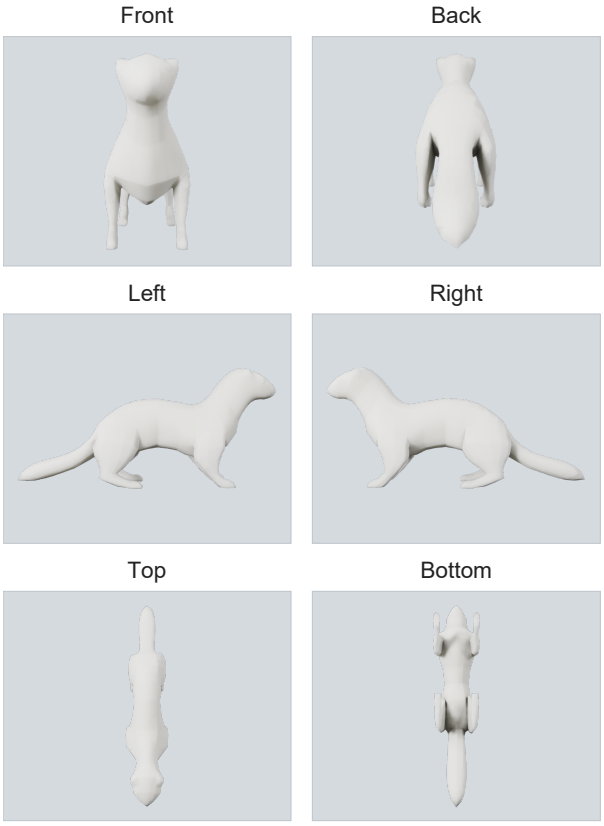
Inferred hierarchies can be revised through VLM feedback or manual editing before rerunning AniRig. The animation-ready asset consists of an explicit rest mesh, a rest skeleton, and normalized LBS weights. These outputs support downstream editing; compatible joint rotations under the specified hierarchy drive the asset using the convention in Appendix B.4.

Inferred-hierarchy evaluation. Table 5 (inferred hierarchy) and Fig. 5 use unedited VLM hierarchies. In the inspected failures, inferred hierarchies omit joints on small limbs, tails, wings, or complex branches, removing articulated degrees of freedom needed for rest recovery.

D.2 Six-View Hierarchy Example

From views to topology. The side views expose the trunk and tail profile, while the front, back, top, and bottom views reveal the bilateral layout of the limbs (Fig. 7). The resulting hierarchy organizes the character into a spine–neck–head chain, paired front and back leg chains, and a tail branch.

(a) Six-view mesh input



(b) VLM-generated hierarchy

```
root
- spine_01
- - spine_02
- - - neck
- - - - head
- - - front_leg_left_01
- - - - front_leg_left_02
- - - - - front_foot_left
- - - front_leg_right_01
- - - - front_leg_right_02
- - - - - front_foot_right
- back_leg_left_01
- - back_leg_left_02
- - - back_foot_left
- back_leg_right_01
- - back_leg_right_02
- - - back_foot_right
- tail_01
- - tail_02
- - - tail_tip
```

Figure 7: **Six-view input and VLM-generated hierarchy.** (a) Orthographic renders of the same character, arranged in the prompt's order: Front/Back, Left/Right, and Top/Bottom. (b) The corresponding joint names and parent–child hierarchy, displayed using the prompt's indentation format. The hierarchy conditions paired skeleton prediction in AniRig; three-dimensional joint positions are predicted by the skeleton model. The full prompt is given in Appendix D.3.

D.3 Full Topology Prompt

Full prompt for the six-view hierarchy construction in Appendix D.1.

```
You are an expert technical rigger.

You will be given one image containing six labeled orthographic
   views of the same 3D mesh:
Front, Back, Left, Right, Top, Bottom.

Your task is to infer a clean, practical, animatable skeleton
   topology directly from the mesh views and output only the
   final topology text.

Rules:
- Build a usable rig topology, not a description of the shape.
- Use the minimum sufficient number of joints.
- Keep only major articulated structures.
- Ignore tiny decorations, wrinkles, cloth folds, and surface
   noise.
```

```
- If the character is symmetric, make the topology symmetric.
- Use exactly one root.
- Every non-root joint must have exactly one parent.
- Chains must go from proximal to distal.
- Do not hallucinate unnecessary fingers, facial bones, ear
    bones, or dense tail segments unless they are clearly
    justified by the mesh.
- If a part is visible but does not need articulation, you may
    omit bones for it.
- Prefer a simple valid rig over an over-complicated one.

Naming rules:
- Use lower_snake_case only.
- Use full words left and right, never use l or r.
- Use names like: root, spine, neck, head, arm_left,
    forearm_left, hand_left, leg_right, shin_right, foot_right,
    tail_01, tail_tip, ear_left, ear_right.
- Use _left and _right for symmetric counterparts.
- Use _01, _02, _03 only for real sequential chains.
- Use _tip only for clear chain endpoints.
- For unusual non-humanoid parts, use neutral names like
    appendage_front_left_01, appendage_back_right_01,
    branch_top_01, extension_left_01.

Output requirements:
- Output only the topology text.
- No explanations.
- No JSON.
- No markdown code fence.
- No comments.

Use exactly this format:
root
- spine
- - neck
- - - head
- arm_left
- - forearm_left
- - - hand_left
- arm_right
- - forearm_right
- - - hand_right

Now analyze the six labeled mesh renders and output the final
    topology text.
```

E LIMITATIONS AND FUTURE DIRECTIONS

Axial twist and rest recovery. Recovering complete joint orientations from bone directions alone involves an inherent geometric ambiguity. A single direction determines the bone axis but leaves rotation about that axis unconstrained. Non-collinear child directions add orientation constraints at branching joints; single-child or collinear configurations require additional information to determine axial twist. Animation tools supply such information through explicit orientation controls, including the up vectors used with Maya's aim constraints (Autodesk, 2022). Similarly, HybrIK (Li et al., 2021a) recovers swing analytically from joint positions and predicts twist from visual features. AniRig uses hierarchical direction alignment at inference (Appendix B.4), so residual twist can remain when bone directions do not fully constrain orientation. Improving joint localization

alone cannot recover this missing degree of freedom. Surface-conditioned twist prediction could exploit additional orientation cues while retaining the paired-skeleton construction. This ambiguity is distinct from the inverse-blending approximation evaluated in Appendix C.5, which uses known joint transforms.

Fine-scale garment and hair structures. Production pipelines for garment and hair animation use specialized representations and dynamics, including cloth simulation with contact handling (Baraff et al., 2003), elastic-strand hair models (Iben et al., 2013), and guide-driven simulation that combines artistic control with physical motion details (Milliez et al., 2018). Our fixed-weight LBS formulation provides skeletal deformation; material response, collision handling, and inertial secondary motion require additional dynamics models. Fine-scale binding accuracy is also limited by the sparse coverage of detailed garment and hairstyle rigs in our training data. UniRig (Zhang et al., 2025a) uses a dedicated VRoid subset containing spring-bone rigs for hair, clothing, and tails, illustrating the value of structure-specific supervision. Increasing the joint count expands kinematic flexibility without supplying either this supervision or the missing dynamics. Targeted annotations and local binding refinement could improve fine-scale skeletal control. The recovered body rig could then provide kinematic inputs for separate cloth or strand representations coupled with specialized simulation, extending the framework to richer secondary motion.

Rigid mechanical assemblies. Mechanical articulation requires component-consistent rigid transforms and constrained motion axes. AniRig predicts soft vertex influences without explicit rigid-part constraints, so neighboring components can acquire coupled motion or deform at their boundaries. Preserving mesh connectivity also prevents shared boundary vertices from following independent component motions where separation is required. Part-level methods provide complementary structural information through open-vocabulary, hierarchical, prompted, or generative decomposition (Ma et al., 2025; Yang et al., 2024; Liu et al., 2025b; Ma et al., 2026; Li et al., 2026; Zhang et al., 2025b), while Shape2Motion (Wang et al., 2019) additionally estimates moving parts and motion attributes. Combining such part structure and motion cues with the recovered rig could enable component-wise rigid binding, separation of shared boundary vertices, and explicit hinge-axis constraints, extending AniRig toward mechanically constrained animation.