RigPAPR: Rig-Based Animation of Static Neural Point Clouds from a Fixed-Viewpoint Video

Shichong Peng[1], **Yanshu Zhang**[1], **Ke Li**[1,2,3]
[1]APEX Lab, School of Computing Science, Simon Fraser University, Canada
[2]Alberta Machine Intelligence Institute (Amii), Canada
[3]Canadian Institute for Advanced Research (CIFAR), Canada
{shichong_peng,yanshu_zhang,keli}@sfu.ca

Abstract

Static neural point reconstructions capture a subject at high fidelity from posed images. Given such a reconstruction, we aim to animate it to follow a monocular fixed-viewpoint driving video of the subject, whether captured or produced by image-to-video (I2V) generation, and to recover a rigged, re-posable 3D asset. Existing methods deform Gaussian splats through direct linear blend skinning (LBS) or mesh proxies, both of which are prone to joint-boundary artifacts under articulation, even with per-primitive corrections. We trace the artifact to the representation: each splat carries an individual shape calibrated in the canonical pose to tile with its neighbours. Under LBS each splat can be moved, rotated, or rescaled but never bent, so the canonical tiling comes apart at joint boundaries. Proximity attention point rendering (PAPR) instead carries no per-primitive shape; each pixel is recomposed at render time from the deformed primitives' positions, so the surface re-forms naturally with the articulation. We present RigPAPR, which auto-rigs a static PAPR cloud and drives it under direct LBS from a single fixed-viewpoint video, without a category template or a pose-dependent correction network. RigPAPR is competitive with the strongest baseline at the supervised view and outperforms splatting-based and mesh-based baselines at novel views, by 3 dB PSNR on synthetic subjects and about 2 dB on real-world subjects, with cleaner joint-boundary renderings. Project page: https://niopeng.github.io/RigPAPR/

1 Introduction

Triangle meshes are the default asset for 3D animation, yet recovering a high-fidelity mesh from real-world capture remains difficult, requiring both accurate surface reconstruction and view-consistent texture and material recovery. Neural rendering sidesteps this by reconstructing scenes at photoreal fidelity from posed images alone, beginning with NeRF's volumetric formulation [31]. Explicit primitives have since become the more popular formulation, led by 3D Gaussian splatting [16] alongside other point-based alternatives [62, 20, 38]. Because each learned primitive is editable after reconstruction, these representations become viable animation assets, not only novel-view renderers.

The most useful animation asset on top of such a static reconstruction is a skeletal rig: a sparse, human-interpretable, re-posable control interface that decades of skeletal-animation practice and automatic rigging are built around [30, 22, 1]. Driving such a rig requires motion; image-to-video (I2V) generation [5, 2, 3] is a natural source, animating any subject from a single canonical frame. Existing rig-based work for neural point representations [43, 45, 25, 36] requires multi-camera or orbiting-camera motion capture, while I2V most directly produces a single fixed-viewpoint video. We focus on this fixed-viewpoint setting: given such a monocular driving video and a static multi-view

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

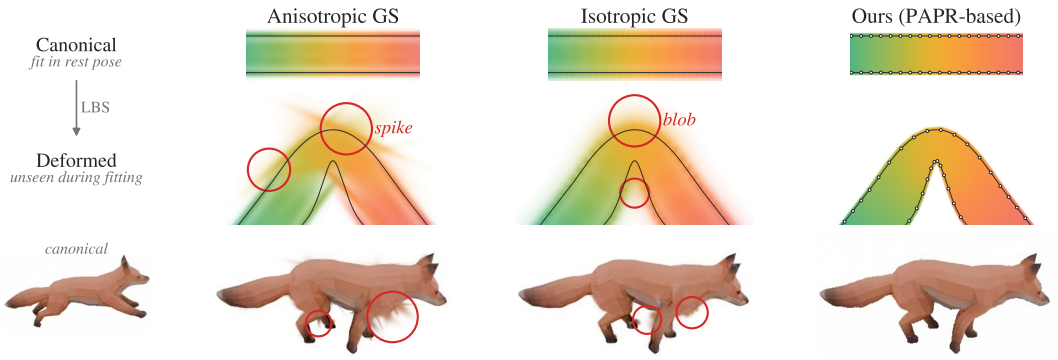

Figure 1: **Top:** a cross-section of a limb, fitted in the canonical pose and then bent at a joint with linear blend skinning (LBS). **Bottom:** anisotropic splats, isotropic splats and ours on the fox, driven by its ground-truth motion. Each Gaussian splat carries its own *individual* shape, fitted in the canonical pose without knowing how the surface will later deform. LBS bends the surface non-rigidly: the outer side stretches, and the inner side compresses and thins. A splat, however, can only be moved, rotated, or rescaled, so its shape stays an ellipsoid and cannot bend with the surface. Long, thin anisotropic splats no longer lie along the bent surface and stick out as spikes. Making the splats isotropic removes the spikes, but the large round splats that fill the interior keep their size as the limb thins, and push through the surface as blobs. PAPR instead builds shape from a *set* of shape-free points: each pixel is recomposed at render time from its deformed neighbours, so the surface follows the bend.

reconstruction of the subject, we aim to recover a rigged 3D asset whose motion can be re-rendered from novel viewpoints and whose skeleton can be re-posed to unseen configurations.

Most existing approaches to rigging neural point-based representations adopt Gaussian-splat primitives, deformed either by direct linear blend skinning (LBS) on the splats [55, 45] or by binding them to a deforming mesh or parametric template [7, 35]. In our setting both routes produce visible artifacts near the joints, where the deformation pushes splats far from their canonical configuration. The cause is the primitive rather than the deformation: each splat carries its own *individual* shape, an ellipsoid fixed by its covariance, which can be moved, rotated, or rescaled, but never bent. Neighbouring splats are fitted to tile into a seamless surface in a single static pose, with no observation of the motion to come, so when LBS later bends a joint the tiling comes apart. Anisotropic splats stand out from the bent surface as spikes; constraining them to be isotropic only rounds these into blobs (Fig. 1).

Directly addressing the cause calls for a different representation: an *interpolation-based* representation, in which no primitive carries a fixed shape parameter. Instead, the shape emerges from the spatial configuration of a *set* of primitives, not from any single one. Under deformation, the positions of the nearby primitives change, and so the rendered pixels change accordingly, which allows the rendering to adapt to the deformed spatial configuration automatically. Fig. 1 (bottom right) shows the consequence: the same deformation that broke the Gaussian tiling can be rendered as a seamless surface with interpolation. Proximity attention point rendering (PAPR) [62] realizes this representation: each primitive carries only a position and a view-independent feature, and each pixel is rendered as an attention-weighted aggregation of its top-K nearest primitives, with weights computed at render time from their current positions. Under direct LBS, the attention weights update automatically with the deformed positions.

Building on this insight, we present *RigPAPR*, a method that animates a static PAPR cloud from a single fixed-viewpoint driving video, captured or I2V-generated, and produces a re-posable rigged 3D asset without category templates. Given a pre-trained PAPR cloud and the driving video, the method first auto-rigs the point cloud through a transient mesh proxy, which is discarded once binding is set. The point cloud is then deformed directly under LBS, with joint rotations and skinning weights refined through two-phase optimization driven by the video. We evaluate on synthetic and real-world subjects against strong representative splatting-based and mesh-based baselines. At the supervised view RigPAPR is competitive with the strongest baseline. At held-out novel views it outperforms all baselines, by 3 dB in PSNR on synthetic subjects and about 2 dB on real-world subjects, and renders joint boundaries more cleanly.

2 Related Work

3D representations for animatable reconstruction. The choice of 3D representation determines how the reconstructed object's geometry is stored before animation. Implicit representations [31, 46, 24] describe geometry as a continuous function of space, whereas explicit representations such as meshes and point clouds store geometry as discrete primitives with positions and, in some cases, local shape parameters. For animation, explicit representations are convenient because their primitives can be transformed directly to change the geometry. Among explicit neural representations, point-based neural renderers reconstruct real-world captures from posed images at high fidelity, without the surface-reconstruction step that mesh-based pipelines require. 3D Gaussian splatting [16] (3DGS) is the dominant such representation. An axis that matters under deformation is how shape is encoded: 3DGS encodes shape *per primitive* via an anisotropic 3D covariance (a local ellipsoidal kernel), whereas PAPR [62] carries no per-primitive shape parameter: shape emerges at render time from proximity attention over a neighbourhood of primitives.

Rig-based animation and its precedent. Captured subjects can be made animatable along a spectrum of control interfaces. Per-frame deformation fields [32, 34, 54, 48] reconstruct the observed motion as a continuous 4D field, but they replay rather than re-pose: the asset exposes no re-pose interface beyond the captured latent trajectory. Cage and control-point methods [12, 11] provide a re-posable asset, but steer it through a dense set of surface handles that remains cumbersome to manipulate directly. Rig-based methods expose a skeleton instead: a sparse, human-interpretable interface that the existing motion-capture data, retargeting tools, and authoring workflows of production pipelines are already built around. We target this setting.

Rig-based animation has been studied extensively for meshes. Automatic rig prediction [1, 52, 41, 59, 28, 53] makes rigging tractable without manual authoring, producing a skeleton with skinning weights bound to a mesh. Recent generative work extends this to motion: ActionMesh [39] generates an animated mesh from a video, and RigMo [57] recovers a rig and its motion from such a mesh sequence. Their combination produces a rigged mesh but recovers no view-consistent appearance of the captured subject from its posed images. Skeleton-driven deformation on non-mesh primitives has precedent: meshless and point-sampled animation driven by skeletons dates to the mid-2000s [10, 40, 37], with modern differentiable descendants [58]. These methods drive point sets or lattices through physics-based deformation (meshless elasticity, shape matching) rather than linear blend skinning, and none target photorealistic rendering of captured scenes.

Fixed-viewpoint rig-based animation. Rig-based work for neural representations has explored multi-view supervision during the driving sequence, whether from multiple synchronised cameras or from a deliberately moving camera that encircles the subject across timesteps [43, 4, 45], coverage that disambiguates depth from articulation. We focus on the setting where the driving sequence is a single fixed-viewpoint video, the form I2V models [5, 2, 3] produce most directly. In this regime, depth and articulation explain the same 2D residual at every frame, and the problem is under-constrained. Methods that operate here constrain the optimization with auxiliary information in three ways: (a) a strong generative prior based on an image or video generation model [49, 8]; (b) a pre-existing category template such as SMPL [19, 35, 25]; (c) monocular pre-trained priors that provide approximate per-frame guidance, such as off-the-shelf depth estimators [27] and 2D point trackers [13]. Our work and AnimaMimic [50] both follow route (c). We differ on the *representation axis*: AnimaMimic's pipeline runs on a mesh that must be supplied upstream, whereas our point-based neural representation is reconstructed directly from posed real images.

Rig-based animation of point-based neural representations. Recent work has begun to rig point-based neural representations. Most efforts operate on Gaussian-splat primitives; one line operates on density-field neural points under volumetric rendering.

Direct LBS on Gaussian splats. LBS is applied directly to the splats with learned per-primitive skinning weights [45, 18]. A parallel line [55, 4] retains the direct-LBS move but adds a pose-dependent detail-deformation head that displaces splat positions at each timestep to compensate for rendering artifacts under articulation. Because the head is trained on the observed pose distribution, this pose-dependent compensation may overfit to those poses and generalize weakly to novel ones.

Mesh-proxy binding. Splats are bound [9, 44, 7, 56, 26] to a triangle mesh that provides a deforming surface reference: as the mesh deforms, splats follow it in position, and their covariances reorient with the local surface frame. A specialized sub-variant uses parametric templates such as SMPL [29] or FLAME [23] as the mesh proxy [19, 25, 35, 36], avoiding topology estimation but restricting the method to categories with a pre-existing template. Several of these works also adopt dual-quaternion skinning [14, 15] or local rigidity regularizers such as ARAP [42] to soften the non-rigid blending of bone transforms at joint boundaries. These fixes adjust the deformed position, but the splat covariance remains calibrated to the canonical-pose surface, so the artifacts are softened but persist.

Cage or sparse control graph. A third family operates on the same Gaussian-splat primitive but replaces the skeleton with a dense set of surface handles, so re-posing becomes geometric editing of the cage rather than skeletal manipulation [49]. These methods sit adjacent to rig-based work and share the primitive-level limitation analyzed below. Recent variants approach the skeleton from the other side: a kinematic-tree control graph [61], or piecewise-rigid Gaussian clusters [17]. Both still expose a dense-handle interface rather than a sparse rig.

All the Gaussian-splat coping patterns above share a structural problem rooted in the primitive rather than the method. Each splat carries an individual shape, an ellipsoid fixed by its covariance, calibrated in the canonical pose to tile with its neighbours. A bone transform moves, rotates and rescales that ellipsoid, and recomputing its covariance from the local deformation only maps it to another ellipsoid. No covariance update bends the splat, so it cannot follow a surface that bends between two bones, and the canonical tiling comes apart at joint boundaries. The mesh-proxy, cage, and correction MLP mechanisms each target a symptom of this mismatch while leaving shape inside the primitive, where the artifact originates. We therefore build on PAPR, an interpolation-based representation whose primitives carry no such parameter: each pixel's rendered output is recomposed at render time from the current positions of its top-K nearest primitives, so the rendered surface follows the deformed configuration.

Density-field neural points. Uzolas et al. [43] rig a Point-NeRF [51]-style density-field point cloud directly with LBS, with no pose-dependent correction MLP. Their points carry no explicit per-primitive shape, yet rendering aggregates per-primitive contributions independently rather than interpolating across primitives at the pixel. They further differ on supervision and on point distribution. They require multi-view supervision (a varying viewpoint across the sequence) to resolve the depth-articulation ambiguity, whereas we operate from a single fixed-viewpoint video. Their points also volumetrically sample the object interior for density-field rendering, which precludes feeding them to surface-based auto-riggers [41, 52, 59] that PAPR's surface-concentrated points support directly.

3 Method

3.1 Overview

From a static PAPR [62] cloud and a fixed-viewpoint driving video, our method produces a rigged point-based asset that reproduces the driving motion, renders faithfully from unseen viewpoints, and is re-posable to unseen configurations. We write the PAPR scene as N points $\{(\mathbf{p}_i, \mathbf{u}_i, \tau_i)\}_{i=1}^{N}$ with positions $\mathbf{p}_i \in \mathbb{R}^3$, view-independent features $\mathbf{u}_i \in \mathbb{R}^d$, and influence scores $\tau_i \in \mathbb{R}$; the skeleton has B bones and each point carries a per-bone skinning weight $\mathbf{w}_i \in \mathbb{R}^B$. A pose at frame t comprises one axis-angle rotation per bone, $\{\mathbf{R}_b^t\}_{b=1}^{B} \subset SO(3)$, plus a global root translation $\mathbf{t}^t \in \mathbb{R}^3$; forward kinematics composes these with the fixed bind-pose bone offsets to yield per-bone $SE(3)$ transforms $\mathbf{T}_b(\mathbf{R}^t)$. The driving video supplies T frames $\{\mathbf{I}^t\}_{t=1}^{T}$ from a single camera $\mathcal{C}$.

Figure 2 shows the overview of our method. §3.2 recalls the PAPR properties we exploit; §3.3 attaches an articulated rig directly to the PAPR cloud, with no mesh proxy at deformation or render time; §3.4 drives the rig from the video through a two-phase optimization that resolves the depth-articulation ambiguity inherent to a single fixed view.

3.2 Preliminaries: Proximity Attention Point Rendering

PAPR represents a scene as N points whose positions and view-independent features are stored as learnable parameters and trained end-to-end against the input images. To render a pixel, PAPR casts a camera ray, selects the top-K points nearest the ray, and aggregates them with a proximity-attention

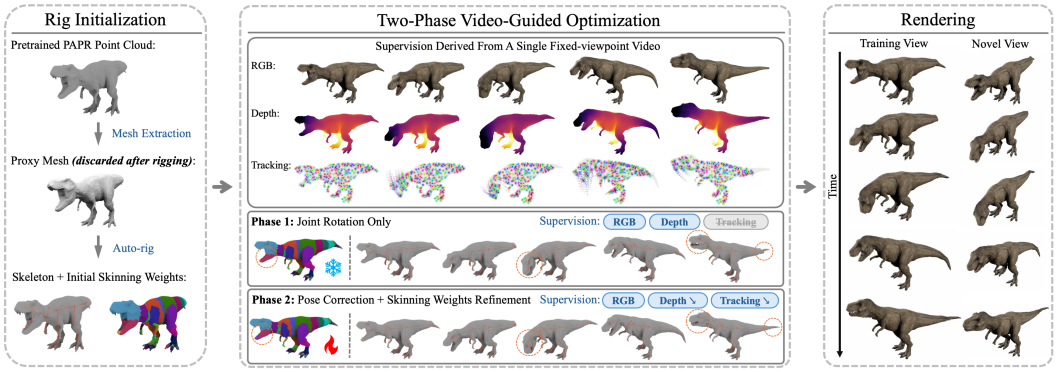

Figure 2: **RigPAPR overview.** A mesh proxy extracted from the PAPR cloud is auto-rigged and the resulting skeleton and initial skinning weights are transferred back onto the points; the proxy is then discarded (§3.3). A two-phase optimization drives the rigged cloud from the fixed-viewpoint video, recovering per-frame joint rotations and refining the skinning weights (§3.4). At inference, LBS deforms the canonical points and the frozen PAPR renderer renders the asset from different viewpoints.

head: the head produces softmax weights over the top-K points and aggregates their per-point contributions into the rendered pixel RGB $\hat{\mathbf{I}}$. The attention weights and per-point contributions are computed from each point's relation to the ray (its position, its depth along the ray, and its perpendicular offset) together with the ray direction; we refer to [62] for the full attention design.

Two properties of PAPR matter for what follows. First, the same indexed point set persists across reconstruction and deformation, and the renderer relies on no surface connectivity (such as mesh edges) that LBS could break. Second, because the attention weights renormalize over the current top-K neighbourhood at render time, every pixel still receives a non-vanishing gradient when points move far from the binding configuration; we exploit this property when fine-tuning the representation per frame. Backbone configuration and the variant we use are detailed in the appendix.

3.3 Direct Linear Blend Skinning on Neural Points

We attach an articulated rig to the PAPR scene by treating each point as a free linear blend skinning (LBS) handle: the same point set used for static reconstruction is the point set we deform per frame, with no mesh proxy or resampling at deformation or render time and no retraining of the renderer.

Skeleton and weight initialization. We mesh the trained PAPR cloud with OffsetOPT [21] and feed the proxy to Puppeteer [41], which returns a B-bone skeleton and per-vertex skinning weights; we transfer the weights to the PAPR points to obtain $\mathbf{w}_i \in \mathbb{R}^B$. The proxy is used only here, to seed the skeleton and skinning weights, and is discarded afterwards; it participates in neither deformation nor rendering. §3.4 then refines $\mathbf{w}$ and recovers per-frame rotations against the driving video.

LBS on PAPR points. Let $\{\mathbf{R}_b^t\}_{b=1}^B$ be the bone rotations at frame t and $\mathbf{T}_b(\mathbf{R}^t) \in SE(3)$ the corresponding skinning transform, defined as the bone-to-world transform at frame t composed with the world-to-bone transform at the binding pose, so that $\mathbf{T}_b(\mathbf{R}^0) = \mathbf{I}$ for every bone and the canonical configuration is the fixed point of Equation (1). With positions written in homogeneous coordinates, the deformed position of point i at frame t is

$$\mathbf{p}_i^t \;=\; \mathbf{t}^t \;+\; \sum_{b=1}^{B} w_{i,b}\, \mathbf{T}_b(\mathbf{R}^t)\, \mathbf{p}_i^0, \tag{1}$$

where $\mathbf{p}_i^0$ is the canonical position and $\mathbf{t}^t$ the global root translation. Only positions are deformed: $\mathbf{u}_i$ and τ_i travel rigidly with the point, and the proximity-attention renderer re-aggregates the deformed neighbourhood at render time. Each PAPR point is therefore a persistent, appearance-stable handle: articulation moves the handle while leaving its appearance content untouched.

A useful side benefit follows directly: with no edges to preserve, the representation does not freeze connectivity at the binding frame. Deformations that require connectivity changes, such as an arm separating from a torso it touches at binding, remain expressible, whereas mesh-bound schemes propagate the binding-frame topology forward indefinitely through their splat-to-triangle pinning.

3.4 Video-Guided Two-Phase Optimization

The single-view setting is under-constrained: depth and articulation explain the same 2D residual at every frame, so a rendering loss alone cannot distinguish the correct configuration from one that renders identically at $\mathcal{C}$ but differs from every other viewpoint. Monocular depth and 2D tracks resolve the ambiguity but inject their own bias and noise; we therefore use them as decaying scaffolding: present early to break the ambiguity, then faded out so the converged optimum is not pinned by their failure modes. Loss weights and schedules are tabulated in the appendix.

Phase 1: Depth-supervised joint-rotation initialization. Phase 1 freezes the skinning weights and optimizes the per-frame joint rotations $\{\mathbf{R}_b^t\}_{t=1}^T$ together with per-frame global root translations $\{\mathbf{t}^t\}_{t=1}^T \subset \mathbb{R}^3$, fit sequentially from the binding frame with each frame warm-started from the previous. The objective combines a rendering loss $\mathcal{L}_{\text{rgb}}$ on the rendered patch with a relative-depth term $\mathcal{L}_{\Delta\text{depth}}$ and a descendant-weighted rotation regularizer that keeps joint rotations small in proportion to how many child joints they propagate to. Monocular predictors such as Depth Anything 3 [27] do not produce depths in PAPR units. We use the metric-depth variant, whose frame-to-frame *relative* signal at the same canonical point tracks true depth changes reasonably well up to a per-frame scalar. We compute that scalar in closed form on each step's sample (held constant in the backward pass) and penalize the calibrated residual in MSE; full construction in the appendix. We do not use 2D tracks at this phase: relative depth alone breaks the front-versus-back ambiguity, and tracking would overconstrain the rotations before they have settled.

Phase 2: Pose corrections and skinning-weight refinement. On top of the Phase 1 trajectory, Phase 2 unfreezes two additional parameter sets: per-joint axis-angle *corrections* produced by a small temporal MLP, and the skinning-weight logits that yield $\mathbf{w}$. The Phase 1 rotations $\{\mathbf{R}_b^t\}$ and the Phase 1 translations $\{\mathbf{t}^t\}$ remain frozen; each correction is added to the Phase 1 rotation of its joint and propagates to points only through LBS with the refined weights. Unlike the per-primitive correction MLPs of Gaussian-splat-based methods (§2), these corrections act at the joint level, on bone rotations rather than primitive parameters.

The Phase 2 objective extends Phase 1 with a CoTracker 3 [13] tracking loss $\mathcal{L}_{\text{track}}$, an as-rigid-as-possible distance-preservation term over each point's k-nearest canonical neighbours that anchors local geometry through deformation, and standard smoothness and magnitude regularizers on the corrections and weights. CoTracker queries are seeded from PAPR's per-pixel attention dominance at the canonical view rather than raw projection, so each track corresponds to a point the renderer actually relies on (details in appendix). Both $\lambda_{\Delta\text{depth}}$ and λ_{track} decay to zero over Phase 2, leaving a pure rendering-driven refinement on top of the joint trajectories that depth and tracking established.

Rendering at novel views. At inference, novel views require a single forward pass: LBS-deform the canonical points with the optimized $(\mathbf{w}, \{\mathbf{R}^t\}, \{\mathbf{t}^t\})$, then render with the frozen PAPR network. The same canonical asset re-poses to arbitrary novel rotations and renders from arbitrary novel views without per-frame or per-view fitting. The one-time fit takes about an hour per phase on a single RTX 3090 for a 16-frame sequence.

4 Experiments

4.1 Experimental Setup

Baselines. We compare against baselines that represent the deformation models our analysis contrasts with. Mani-GS [7] deforms Gaussian splats through a mesh: each splat is bound to a triangle of a mesh proxy that is deformed by the rig. RigGS [55] instead applies linear blend skinning directly to the splats, with a pose-conditioned network that corrects the skinned result. Puppeteer [41] serves as a mesh-based animation baseline. We adapt both splatting-based baselines to our setting,

Table 1: Quantitative comparison on synthetic sequences (5-scene mean) and on ZJU-MoCap (2-scene mean). We report PSNR, SSIM and LPIPS at the training view and at four held-out novel views. Puppeteer [41] is evaluated on synthetic data only. Baselines are Mani-GS [7] and RigGS [55]. **Best** and second best per column; values that print identically are underlined on both sides.

	Synthetic						ZJU-MoCap					
	Train view			Novel view			Train view			Novel view		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Puppeteer	22.48	0.941	0.044	19.27	0.923	0.075	—	—	—	—	—	—
Mani-GS	28.53	0.974	**0.026**	20.51	0.927	0.072	26.38	**0.974**	**0.022**	21.33	0.937	0.065
RigGS	**29.52**	**0.975**	0.029	20.37	0.926	0.069	27.06	0.969	0.037	21.21	0.937	0.057
Ours	29.04	0.971	0.027	**23.51**	**0.944**	**0.045**	**27.67**	0.972	0.034	**23.20**	**0.952**	**0.044**

giving them the same canonical reconstruction, auto-rig, and two-stage optimization as our method; implementation details are in appendix D.3.

Datasets. For synthetic evaluation we pick five objects from Objaverse [6] spanning three morphologies: bipeds (T-Rex, Simpsons), quadrupeds (fox, wolf), and an octopod (spider). For each scene we render a fixed-viewpoint 512×512 video as the training view. For the splatting-based baselines and our method we additionally pre-train on the first frame using multi-view reconstruction, rendered from a ring of cameras around the subject; Puppeteer takes the ground-truth mesh and texture directly. For real-world evaluation we use a short clip of each of two subjects (313 and 390) from ZJU-MoCap [33], a synchronized multi-camera capture of human motion: we pre-train on the first frame of the multi-view capture, use one camera as the fixed-viewpoint driving video, and hold out the remaining cameras as ground truth for novel views. Across both settings, sequences are 16–40 frames; per-scene lengths are tabulated in the appendix. Finally, as a demonstration of the image-to-video use case that motivates the fixed-viewpoint setting, we capture two static scenes of our own (robot, statue) and animate them from a Kling 3.0 [5] video generated at one canonical view; we show qualitative results only (Figure 6).

Metrics. We report PSNR, SSIM [47], and LPIPS [60] (VGG backbone) at the training view and at four held-out cameras. On synthetic data the held-out cameras are placed uniformly on a ring around the subject; on ZJU-MoCap they are the capture camera nearest the driving camera and those at roughly 90°, 180° and 270° from it, with the subject's foreground composited on a white background. All metrics are averaged over the sequence with the canonical frame excluded.

4.2 Quantitative Results

Table 1 reports the quantitative comparison. At the training view, our method is competitive with both splatting-based baselines on synthetic scenes and on ZJU-MoCap, and outperforms Puppeteer on every metric. At held-out novel views, our method outperforms all baselines on every metric on both datasets, by 3 dB PSNR on synthetic scenes and about 2 dB on ZJU-MoCap. Although every method loses quality away from the supervised view, ours best preserves novel-view quality.

4.3 Qualitative Results

Video renderings for all results below are provided on the project website; we encourage the reader to consult them, as articulation artifacts are most evident in motion.

Synthetic qualitative. Figure 3 shows the qualitative comparison on synthetic T-Rex; the remaining four subjects (fox, Simpsons, wolf, spider) are deferred to the appendix (Figures 8–11). Mani-GS and RigGS show visible artifacts under articulation (highlighted in red), most pronounced around the head and belly in the novel view. Puppeteer fails to fit the correct geometry, exemplified by the head not pointing in the right orientation. Our method avoids both failure modes.

Real-world qualitative. Figure 4 shows one subject from ZJU-MoCap at the training view and at a held-out view; a second subject is in the appendix (Figure 12). At the training view all methods reproduce the captured frames. At the held-out view each baseline degrades in its own way: Mani-GS

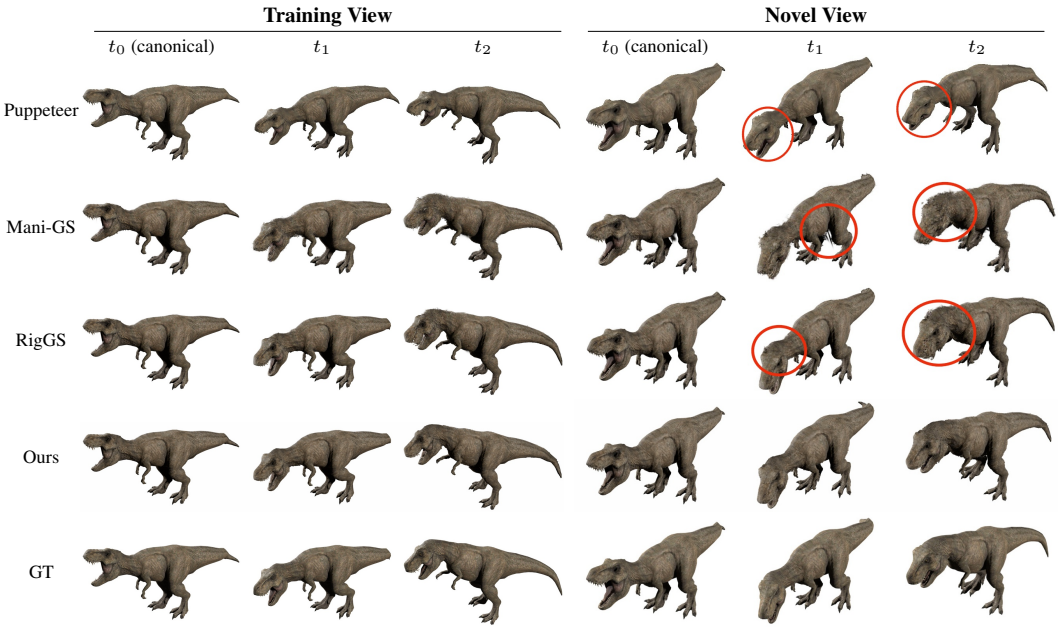

Figure 3: **Qualitative comparison on synthetic T-Rex.** Rows are methods, columns are frames; left block is the supervised *Training View*, right block is a held-out *Novel View*. Column t_0 is the canonical (binding) frame; t_1, t_2 are progressively articulated frames. Red highlights mark Puppeteer's joint-rotation errors and the articulation artifacts of Mani-GS and RigGS; best viewed zoomed in.

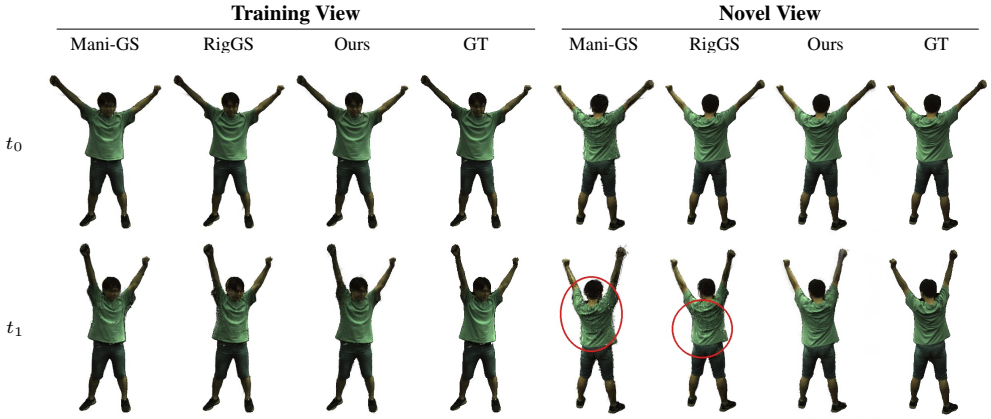

Figure 4: **Qualitative comparison on ZJU-MoCap.** Rows are frames; columns are split into the supervised *Training View* (left) and a held-out *Novel View* (right), each showing Mani-GS [7], RigGS [55], ours and the ground truth. t_0 is the canonical (binding) frame; t_1 is an articulated frame. In the novel view, our renderings are cleaner, without the spikes or gaps of the baselines (red); best viewed zoomed in.

shows needle-like spikes over the back and head, and RigGS opens gaps at the waist, while ours remains clean.

Novel pose driving. Figure 5 drives each fitted model with poses from a later part of the ZJU-MoCap capture that were not part of the fitting trajectory, retargeted to each method's own rig. Only pose-independent outputs carry over to unseen poses: the refined skinning weights for every method, and for Mani-GS additionally the refined per-splat binding rotation and offset; components tied to the training frames (our Phase 2 correction MLP and RigGS's detail head) are disabled at test time. Both splatting-based baselines show artifacts at the joints, while ours does not.

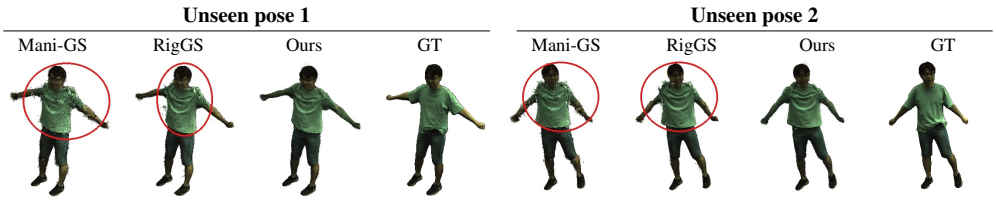

Figure 5: **Novel pose driving on ZJU-MoCap.** Each method is driven at test time with poses from a later part of the capture that was not part of the fitting trajectory; the ground truth is the captured frame. Columns show Mani-GS, RigGS, ours and the ground truth. Our renderings are cleaner, without the spikes or gaps of the baselines (red). Best viewed zoomed in.

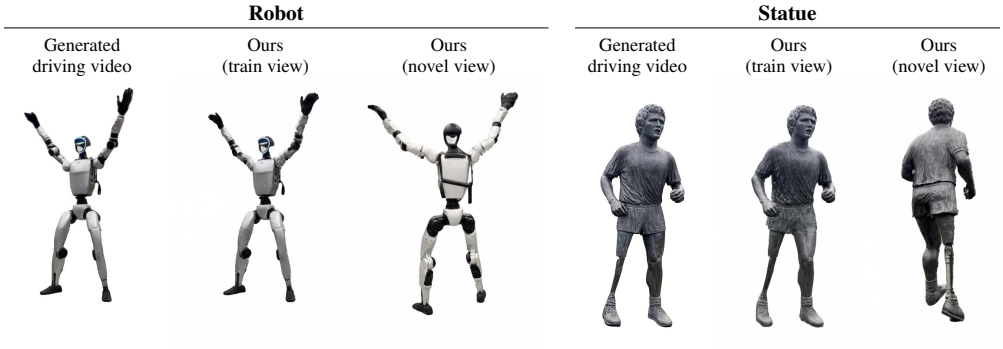

Figure 6: **Captured scenes.** Each static capture is animated from a generated driving video. For each scene, columns show one frame of the driving video and our result at the training view and at a novel view.

Captured scenes. Figure 6 shows our results on the two captured scenes, each animated from its generated driving video. At the training view our model follows the generated motion, and from a novel view it renders a clean surface as the subject moves. More frames and novel pose driving of the robot are in the appendix (Figures 13–15).

4.4 Ablations

Ground-truth deformation. To separate the rendering representation from rig error, we deform every method with the ground-truth motion of the synthetic subject and fine-tune none of them per frame. We compare three splat variants with ours: Mani-GS [7], which binds its splats to the ground-truth mesh; its isotropic variant; and RigGS [55], which skins its splats directly with LBS, with its pose-dependent correction head omitted since the deformation is given. Every splat variant uses more primitives than ours, from about two to fourteen times as many. Even so, on Simpsons (Figure 7), all three produce artifacts at the legs when the knees bend, while ours does not. At the binding frame all methods render comparably, and fox and T-Rex show the same pattern (appendix B.2). With the deformation exact, the artifact remains whether the splats follow a mesh or are skinned directly, and whether they are anisotropic or isotropic; it disappears only when the primitive changes.

Two-phase optimization. Table 2 ablates the two phases of our method on T-Rex, including the relative-depth term of Phase 1. Without $\mathcal{L}_{\Delta\text{depth}}$, Phase 1 fits the training view more closely, by 1.5 dB PSNR, yet the chamfer distance nearly triples and novel-view PSNR drops by 2.7 dB. From a single fixed view, a joint rotated wrongly in depth can still match the pixels well, so a closer training-view fit does not mean a more accurate 3D pose. Phase 2 then raises training-view quality and lowers the chamfer distance further by correcting Phase 1's joint-rotation errors, such as the T-Rex head and tail in Figure 2, while novel-view quality stays about the same. Across methods, Phase 2 likewise raises training-view PSNR while leaving novel-view quality essentially unchanged (appendix B.1).

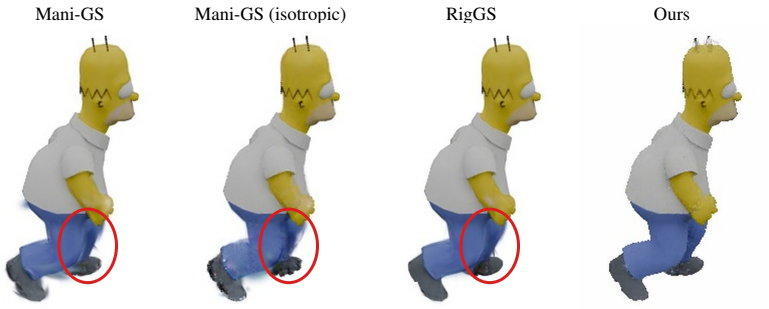

Figure 7: **Ground-truth deformation on Simpsons.** Every method is deformed with the ground truth, so no rig error enters the comparison. The splat-based methods produce artifacts at the legs when the knees bend (red); ours does not. Best viewed zoomed in.

Table 2: **Two-phase ablation of our method on T-Rex.** Training- and novel-view metrics use the same cameras as Table 1; chamfer distance is measured against the ground-truth mesh and averaged over the driving-video frames. Removing $\mathcal{L}_{\Delta\text{depth}}$ improves the training view at the cost of novel-view quality and 3D accuracy; Phase 2 improves both the training view and the chamfer distance.

Configuration	Training view			Novel view			Chamfer ($\times 10^{-3}$)↓
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	
Phase 1 w/o $\mathcal{L}_{\Delta\text{depth}}$	29.21	0.953	0.034	21.38	0.934	0.045	15.64
Phase 1	27.72	0.945	0.035	24.05	0.942	0.039	5.68
Full	31.08	0.972	0.025	23.95	0.945	0.034	4.60

Loss-weight sensitivity. We scale each loss-weight group by $\times\frac{1}{3}$ and $\times 3$ on T-Rex and fox, with all other weights at their defaults (appendix C). Novel-view PSNR varies by at most 0.54 dB for every group except the relative-depth weight (1.55 dB), which matters most, consistent with Table 2.

5 Final Discussion and Conclusion

Limitations. RigPAPR models skeletal articulation under LBS; muscle bulging, soft tissue, cloth, hair, and facial motion are not modelled. Within this scope, RigPAPR inherits the failure modes of its 2D scaffolding. Phase 1 relies on monocular depth and Phase 2 further on 2D tracking; where these priors fail, the errors propagate into the recovered articulation and cannot be corrected by the rendering loss, since the single fixed view does not constrain the failure direction. The auto-rigging stage is similarly load-bearing: a poor skeleton or miscomputed skinning seeds an optimum that downstream refinement only partially escapes, particularly on morphologies the auto-rigger was not trained for. Finally, regions occluded in the binding frame receive no appearance supervision during canonical pretraining, so articulation that later exposes them tends to render as holes or stray colors rather than a coherent surface.

Conclusion. We traced joint-boundary artifacts in Gaussian-splat-based rigging methods to a representation-level incompatibility with LBS, and showed that covariance-free, interpolation-based point primitives address the underlying cause rather than patching the symptom. RigPAPR auto-rigs a static PAPR cloud and animates it from a fixed-viewpoint driving video, captured or I2V-generated. It renders cleanly under articulation at both the supervised and novel views. The rigged cloud is a reusable 3D asset, re-posable to new motions beyond the driving video.

Acknowledgments and Disclosure of Funding

We thank Dun Yang for allowing us to capture the robot and for helping us set it up. This research was enabled in part by support provided by NSERC, the BC DRI Group, the Digital Research Alliance of Canada and the Canada CIFAR AI Chairs Program. We declare no competing interests.

References

[1] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics*, 26(3):72:1–72:8, 2007.

[2] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[3] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators, 2024.

[4] Jun-Jee Chao and Volkan Isler. SV-GS: Sparse view 4D reconstruction with skeleton-driven Gaussian splatting. *arXiv preprint arXiv:2601.00285*, 2026.

[5] Jialu Chen, Yuanzheng Ci, Xiangyu Du, Zipeng Feng, Kun Gai, Sainan Guo, Feng Han, Jing He, Kang He, Xiao Hu, Xiaohuan Hu, Boyuan Jiang, Fangyuan Kong, Hang Li, Jie Li, Qingyu Li, Shen Li, Xiaohan Li, Yan Qiang Li, Jiajun Liang, Borui Liao, Yiqiao Liao, Weihong Lin, Quande Liu, Xiaokun Liu, Yilun Liu, Yuliang Liu, Shun Lu, Hangyu Mao, Yu-Fei Mao, Haodong Ouyang, Wenyu Qin, Wan-Ying Shi, Xiao-Xue Shi, Li Su, Hao Sun, Peiqin Sun, Peng Wan, Chao Wang, Chenyu Wang, Meng Wang, Qiulin Wang, Runqian Wang, Xintao Wang, Xuebo Wang, Zekun Wang, Mingzhe Wei, Tian yong Wen, Guohao Wu, Xiaoshi Wu, Zhenhua Wu, Daihong Xie, Yi-Xue Xiong, Yulong Xu, Si bo Yang, Zikang Yang, Wei Ye, Ziyang Yuan, Shenglong Zhang, Shuai Zhang, Yuanxing Zhang, Yufang Zhang, Wen-Shan Zhao, Rui Zhou, Yan Zhou, Guosheng Zhu, and Yongji Zhu. Kling-Omni technical report. *arXiv preprint arXiv:2512.16776*, 2025.

[6] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[7] Xiangjun Gao, Xiaoyu Li, Yiyu Zhuang, Qi Zhang, Wenbo Hu, Chaopeng Zhang, Yao Yao, Ying Shan, and Long Quan. Mani-GS: Gaussian splatting manipulation with triangular mesh. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21392–21402, 2025.

[8] Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Liu, Wenping Wang, and Yuan Liu. Diffusion as shader: 3D-aware video diffusion for versatile video generation control. In *ACM SIGGRAPH 2025 Conference Papers*, pages 62:1–62:12. ACM, 2025.

[9] Antoine Guédon and Vincent Lepetit. SuGaR: Surface-aligned gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5354–5363, 2024.

[10] Xiaohu Guo and Hong Qin. Real-time meshless deformation. *Computer Animation and Virtual Worlds*, 16(3-4):189–200, 2005.

[11] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. SC-GS: Sparse-controlled Gaussian splatting for editable dynamic scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4220–4230, 2024.

[12] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, Thomas Leimkühler, and George Drettakis. NeRFshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1), May 2023.

[13] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker3: Simpler and better point tracking by pseudo-labelling real videos. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[14] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, pages 39–46, 2007.

[15] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics*, 27(4):1–23, 2008.

[16] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[17] Jinhyeok Kim, Jaehun Bang, Seunghyun Seo, and Kyungdon Joo. Rigidity-aware 3D gaussian deformation from a single image. In *SIGGRAPH Asia 2025 Conference Papers*, pages 179:1–179:11. ACM, 2025.

[18] Taeyeon Kim, Youngju Na, Jumin Lee, Sebin Lee, Minhyuk Sung, and Sung-Eui Yoon. MorphGS: Morphology-adaptive articulated 3D motion transfer from videos. *arXiv preprint arXiv:2601.02716*, 2026.

[19] Muhammed Kocabas, Jen-Hao Rick Chang, James Gabriel, Oncel Tuzel, and Anurag Ranjan. HUGS: Human Gaussian splats. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[20] Christoph Lassner and Michael Zollhöfer. Pulsar: Efficient sphere-based neural rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1440–1449, 2021.

[21] Huan Lei. OffsetOPT: Explicit surface reconstruction without normals. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[22] John P Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 165–172, 2000.

[23] Tianye Li, Timo Bolkart, Michael J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics*, 36:1–17, 2017.

[24] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H. Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[25] Zhe Li, Zerong Zheng, Lizhen Wang, and Yebin Liu. Animatable Gaussians: Learning pose-dependent Gaussian maps for high-fidelity human avatar modeling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[26] Zhiqi Li, Yiming Chen, and Peidong Liu. DreamMesh4D: Video-to-4D generation with sparse-controlled gaussian-mesh hybrid representation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[27] Haotong Lin, Sili Chen, Junhao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

[28] 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, 2025.

[29] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[30] Nadia Magnenat-Thalmann, Richard Laperrière, and Daniel Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics Interface'88*, pages 26–33, 1989.

[31] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.

[32] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5865–5874, 2021.

[33] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9050–9059, 2021.

[34] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10318–10327, 2021.

[35] Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. GaussianAvatars: Photorealistic head avatars with rigged 3D Gaussians. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[36] Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3DGS-Avatar: Animatable avatars via deformable 3D gaussian splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5020–5030, 2024.

[37] Alec R. Rivers and Doug L. James. FastLSM: Fast lattice shape matching for robust real-time deformation. In *ACM SIGGRAPH 2007 Papers*, pages 82:1–82:6, 2007.

[38] Darius Rückert, Linus Franke, and Marc Stamminger. ADOP: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics*, 41(4):99:1–99:14, 2022.

[39] Remy Sabathier, David Novotný, Niloy J. Mitra, and Tom Monnier. ActionMesh: Animated 3D mesh generation with temporal 3D diffusion. *arXiv preprint arXiv:2601.16148*, 2026.

[40] Chao Song, Hong-xin Zhang, Jin Huang, and Hu-jun Bao. Meshless simulation for skeleton driven elastic deformation. *Journal of Zhejiang University-Science A*, 7(9):1596–1602, 2006.

[41] 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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[42] Olga Sorkine-Hornung and Marc Alexa. As-rigid-as-possible surface modeling. In *Eurographics Symposium on Geometry Processing*, volume 4, pages 109–116, 2007.

[43] Lukas Uzolas, Elmar Eisemann, and Petr Kellnhofer. Template-free articulated neural point clouds for reposable view synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 31621–31637, 2023.

[44] Joanna Waczyńska, Piotr Borycki, Sławomir Tadeja, Jacek Tabor, and Przemysław Spurek. GaMeS: Mesh-based adapting and modification of Gaussian splatting. *arXiv preprint arXiv:2402.01459*, 2024.

[45] Diwen Wan, Yuxiang Wang, Ruijie Lu, and Gang Zeng. Template-free articulated Gaussian splatting for real-time reposable dynamic view synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 62000–62023, 2024.

[46] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[47] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[48] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian Splatting for real-time dynamic scene rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, 2024.

[49] Zijie Wu, Chaohui Yu, Yanqin Jiang, Chenjie Cao, Fan Wang, and Xiang Bai. SC4D: Sparse-controlled video-to-4D generation and motion transfer. In *European Conference on Computer Vision (ECCV)*, pages 361–379. Springer, 2024.

[50] Tianyi Xie, Yunuo Chen, Yaowei Guo, Yin Yang, Bolei Zhou, Demetri Terzopoulos, Ying Jiang, and Chenfanfu Jiang. AnimaMimic: Imitating 3D animation from video priors. *arXiv preprint arXiv:2512.14133*, 2025.

[51] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5438–5448, 2022.

[52] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. RigNet: Neural rigging for articulated characters. *ACM Transactions on Graphics*, 39(4):58:1–58:14, 2020.

[53] 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*, 2022.

[54] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20331–20341, 2024.

[55] Yuxin Yao, Zhi Deng, and Junhui Hou. RigGS: Rigging of 3D gaussians for modeling articulated objects in videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5592–5601, 2025.

[56] Yuxin Yao and Junhui Hou. A^3-GS: Animate any articulated objects with 3D Gaussian splatting, 2026.

[57] Hao Zhang, Jiahao Luo, Bohui Wan, Yizhou Zhao, Zongrui Li, Michael Vasilkovsky, Chaoyang Wang, Jian Wang, Narendra Ahuja, and Bing Zhou. RigMo: Unifying rig and motion learning for generative animation. *arXiv preprint arXiv:2601.06378*, 2026.

[58] Hao Zhang, Haolan Xu, Chun Feng, Varun Jampani, and Narendra Ahuja. PhysRig: Differentiable physics-based skinning and rigging framework for realistic articulated object modeling. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[59] 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, 2025.

[60] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018.

[61] Xinyu Zhang, Haonan Chang, Yuhan Liu, and Abdeslam Boularias. Motion blender gaussian splatting for dynamic reconstruction. In *Proceedings of the 9th Conference on Robot Learning (CoRL)*, volume 305 of *Proceedings of Machine Learning Research*, pages 5107–5128. PMLR, 2025.

[62] Yanshu Zhang, Shichong Peng, Alireza Moazeni, and Ke Li. PAPR: Proximity attention point rendering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

This appendix is organized into four parts. Section A extends the qualitative comparison of the main paper to the four remaining synthetic subjects (fox, Simpsons, wolf, spider) and to the second ZJU-MoCap subject, and shows our results on two scenes we captured ourselves (robot, statue), animated from a generated video and driven to novel poses (Section A.1). Section B groups two additional ablations: the two-phase optimization ablation across methods, with a closer look at our method and its depth supervision on T-Rex (Section B.1), and a ground-truth deformation ablation that isolates the rendering representation from geometry and skinning quality (Section B.2). Section C reports the sensitivity of the two-phase optimization to its loss weights. Section D collects additional implementation details: dataset sequence lengths (Section D.1), the PAPR backbone and auto-rig pipeline (Section D.2), the baseline implementations (Section D.3) and a Mani-GS configuration comparison that motivates disabling its per-triangle adaptive scaling (Section D.4), the two-phase optimization losses and schedules (Section D.5), and the attention-guided track-seeding procedure (Section D.6).

A Additional Qualitative Results

Figures 8–11 extend the synthetic qualitative comparison of Section 4.3 (Figure 3) to the four remaining synthetic subjects. Rows are methods (Puppeteer [41], Mani-GS [7], RigGS [55], ours, ground truth) and columns are matched frames; each figure shows the supervised *Training View* and a held-out *Novel View* with t_0 the canonical (binding) frame. Spider is shown with two frames per view rather than three so each image stays large enough to read the limb-junction detail. The same failure modes observed in the main figure recur across morphologies. Mani-GS produces spiky artifacts, especially around joint boundaries: the neck and back for fox, the arm and leg for Simpsons, the chest and tail for wolf, and the legs for spider. RigGS shows similar artifacts at the joints: the feet for Simpsons, the tail and front leg for wolf, and the front leg next to the jaws for spider. Puppeteer fails to deform the geometry to match the observed motion sequence accurately. While its result is sometimes passable, it is not consistently accurate, with limbs and body parts often not reaching their target positions and the geometry sometimes structurally incorrect. Our method recovers the joint rotations more accurately throughout the motion sequence and renders cleanly without artifacts. For parts of the subject that remain occluded under the fixed training view, however, the geometry does not always track as accurately: the 2D priors we use to disambiguate single-view supervision provide no signal in occluded regions either (e.g., the left arm of Simpsons).

Figure 12 shows the comparison on the second ZJU-MoCap subject. At the held-out view, both Mani-GS and RigGS show artifacts on the arms, while ours remains clean.

A.1 Captured Scenes

This section adds frames to the captured scenes of Figure 6. We photograph a static subject (a robot and a statue) from multiple views, pre-train a PAPR cloud on the photos, generate a driving video at one canonical view with Kling 3.0 [5], and animate the cloud from that video. Because the driving video is generated, no ground truth exists at novel views, so we show qualitative results only. Figures 13 and 14 show our results at the training view beside the driving video, and from a held-out viewpoint.

We further drive the captured robot with two pose sequences unseen in the driving video (Figure 15). The model follows both sequences, including arm poses far from any in the driving video.

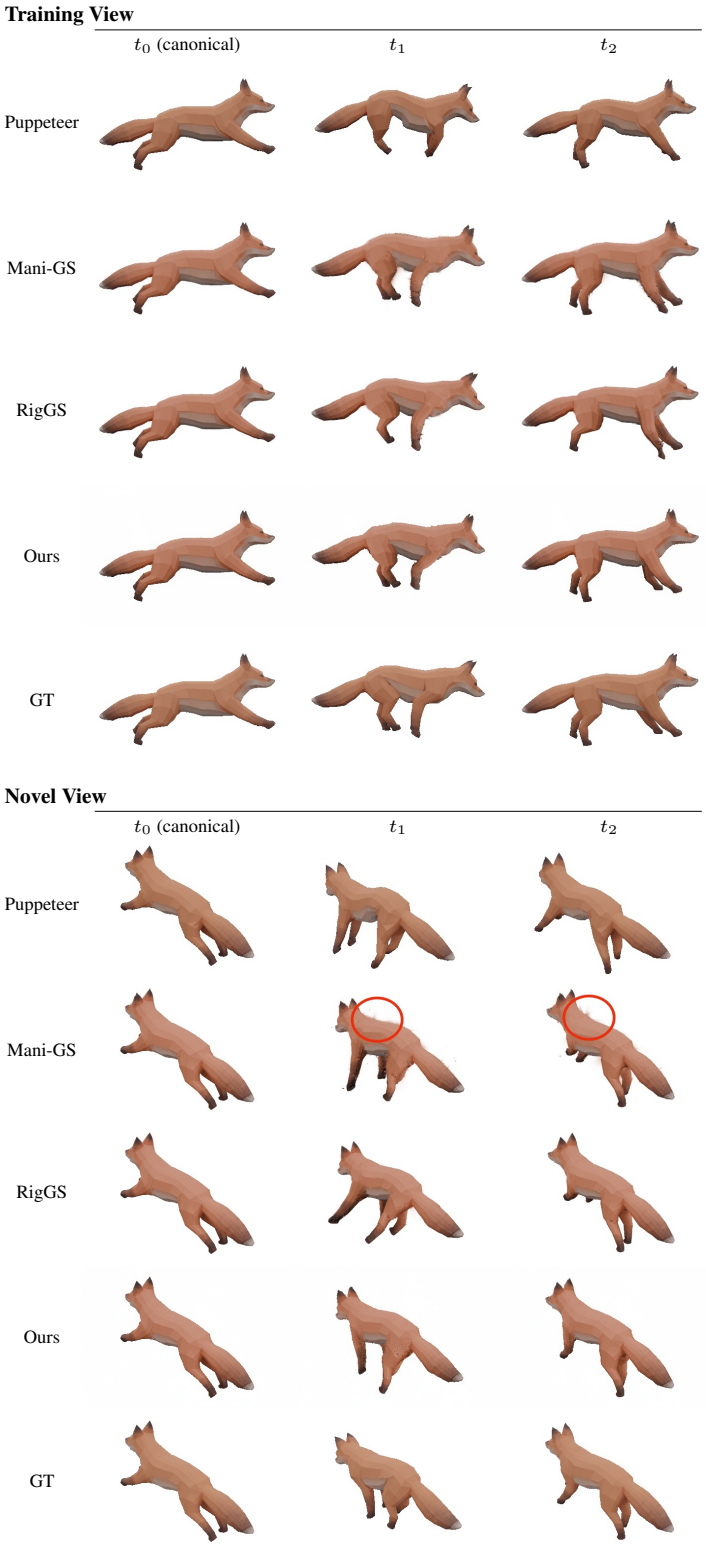

Figure 8: **Qualitative comparison on synthetic fox.** Training View is on top and Novel View is on the bottom. Rows are methods; columns are matched frames, with t_0 the canonical (binding) frame and t_1, t_2 progressively articulated.

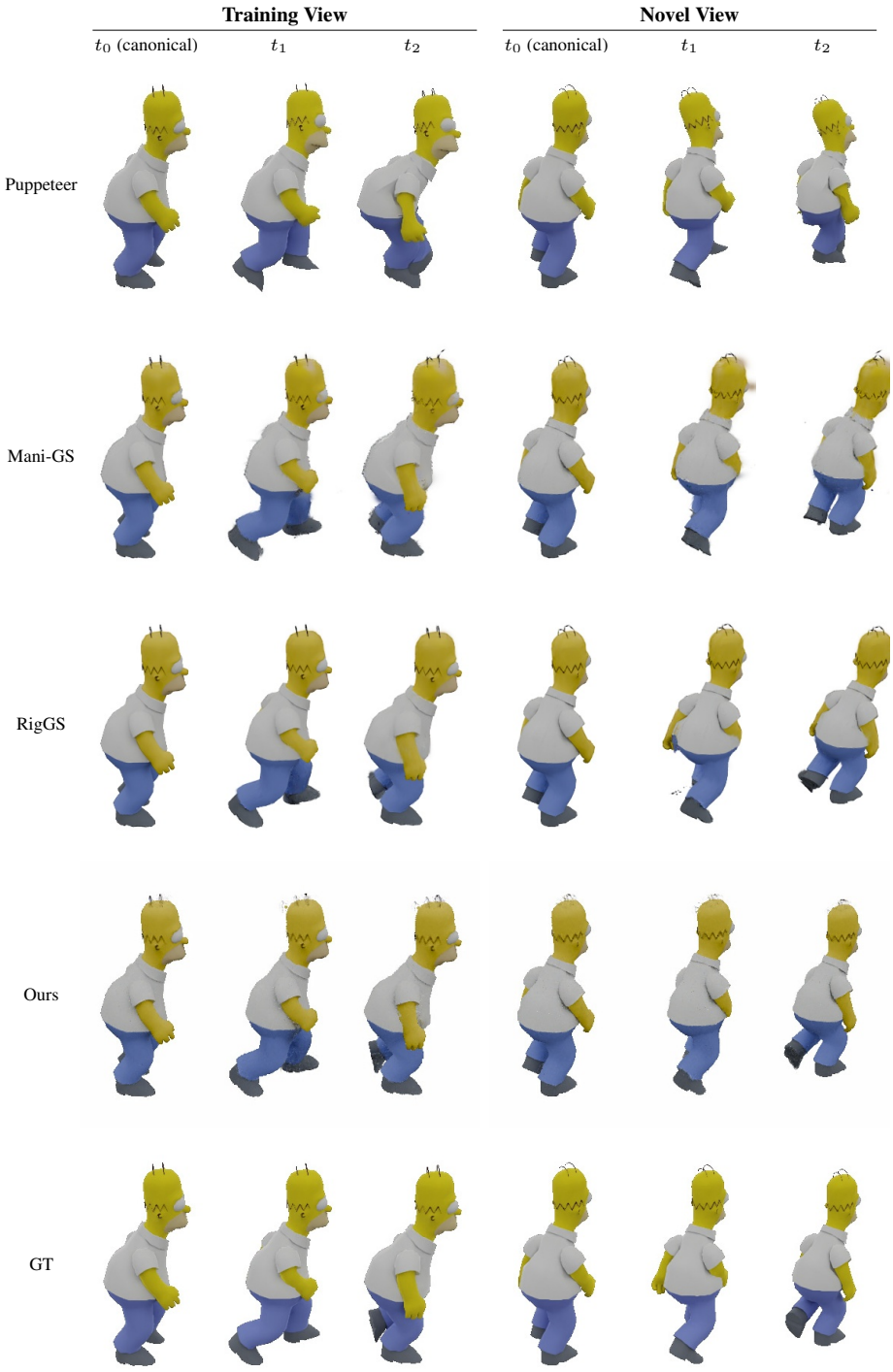

Figure 9: **Qualitative comparison on synthetic Simpsons.** Rows are methods; columns are matched frames split into a supervised *Training View* (left) and a held-out *Novel View* (right). t_0 is the canonical (binding) frame; t_1, t_2 are progressively articulated frames.

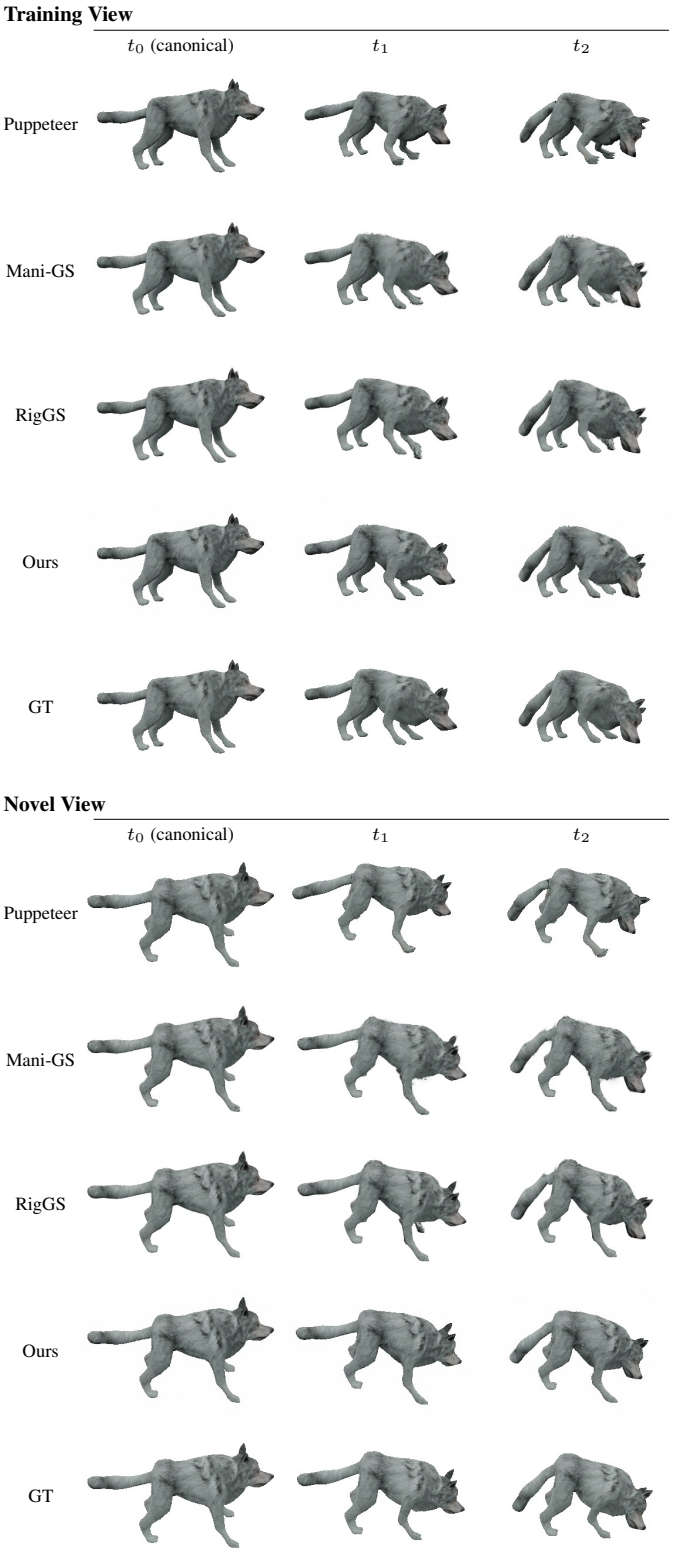

Figure 10: **Qualitative comparison on synthetic wolf.** Training View is on top and Novel View is on the bottom. Rows are methods; columns are matched frames, with t_0 the canonical (binding) frame and t_1, t_2 progressively articulated.

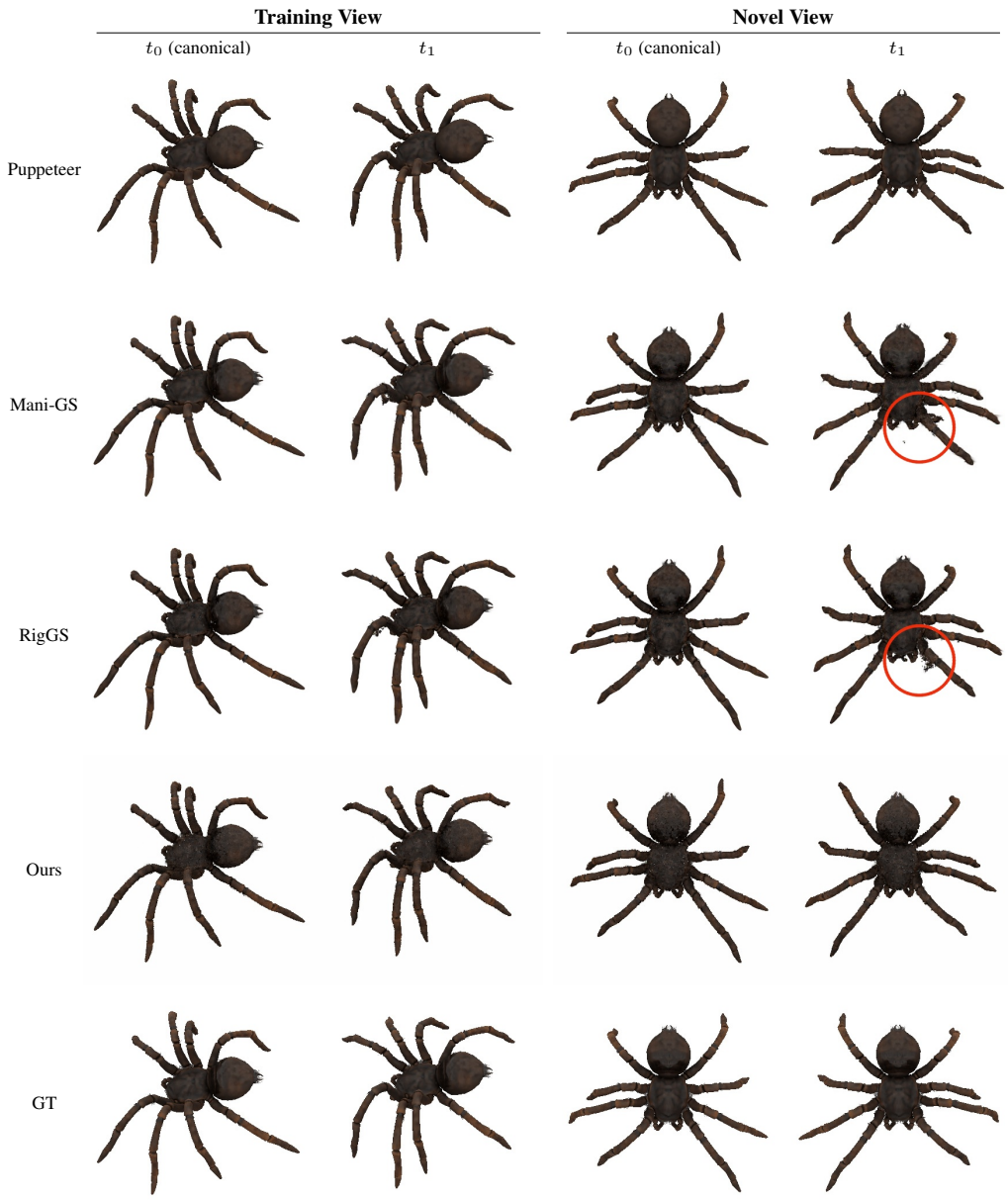

Figure 11: **Qualitative comparison on synthetic spider.** Rows are methods; columns are matched frames split into a supervised *Training View* (left) and a held-out *Novel View* (right). t_0 is the canonical (binding) frame and t_1 is the articulated frame shown; the additional t_2 is omitted to allow larger images.

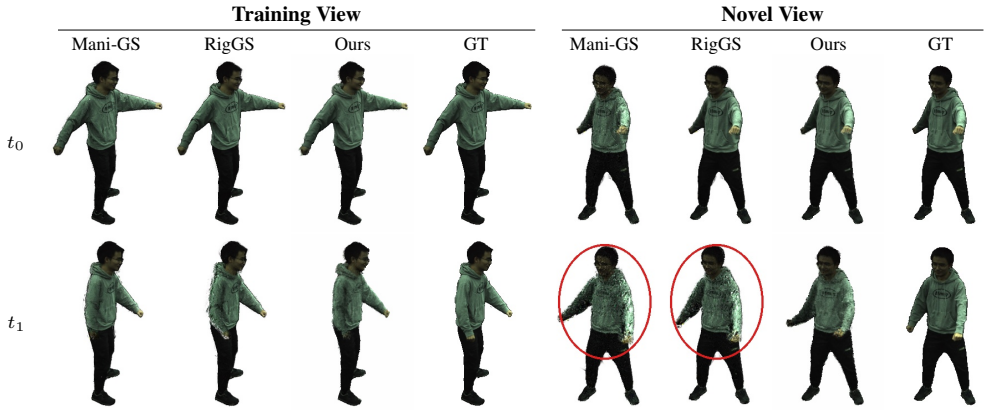

Figure 12: **Qualitative comparison on the second ZJU-MoCap subject.** Layout as in Figure 4: rows are frames; columns are split into the supervised *Training View* (left) and a held-out *Novel View* (right), each showing Mani-GS [7], RigGS [55], ours and the ground truth. t_0 is the canonical (binding) frame; t_1 is an articulated frame. In the novel view, our renderings are cleaner, without the spikes or gaps of the baselines (red); best viewed zoomed in.

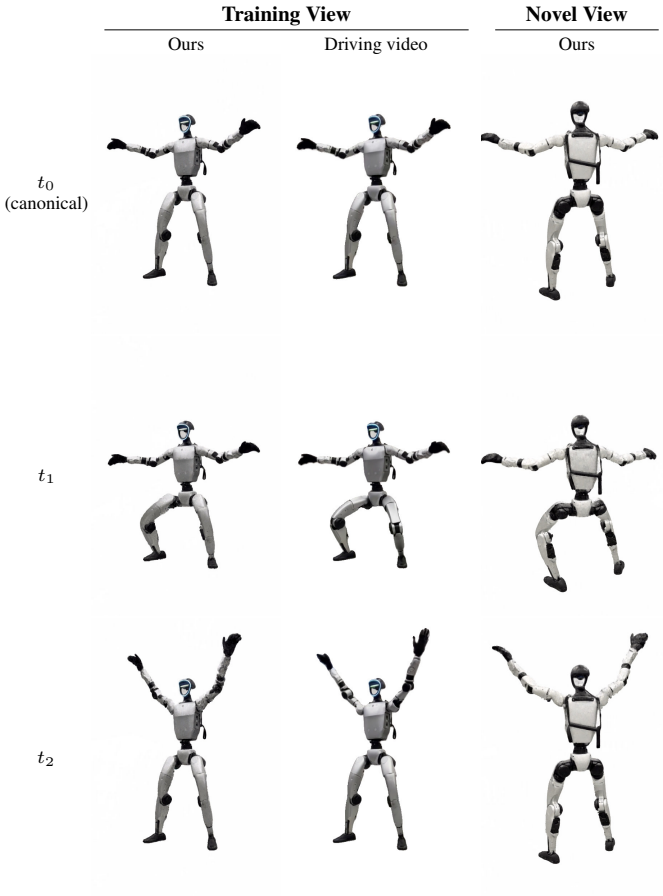

Figure 13: **Our results on the captured robot.** Rows are matched frames (t_0 is the canonical/binding frame; t_1, t_2 are progressively articulated); columns are split into the supervised *Training View* (ours and the generated driving video) and a held-out *Novel View* (ours).

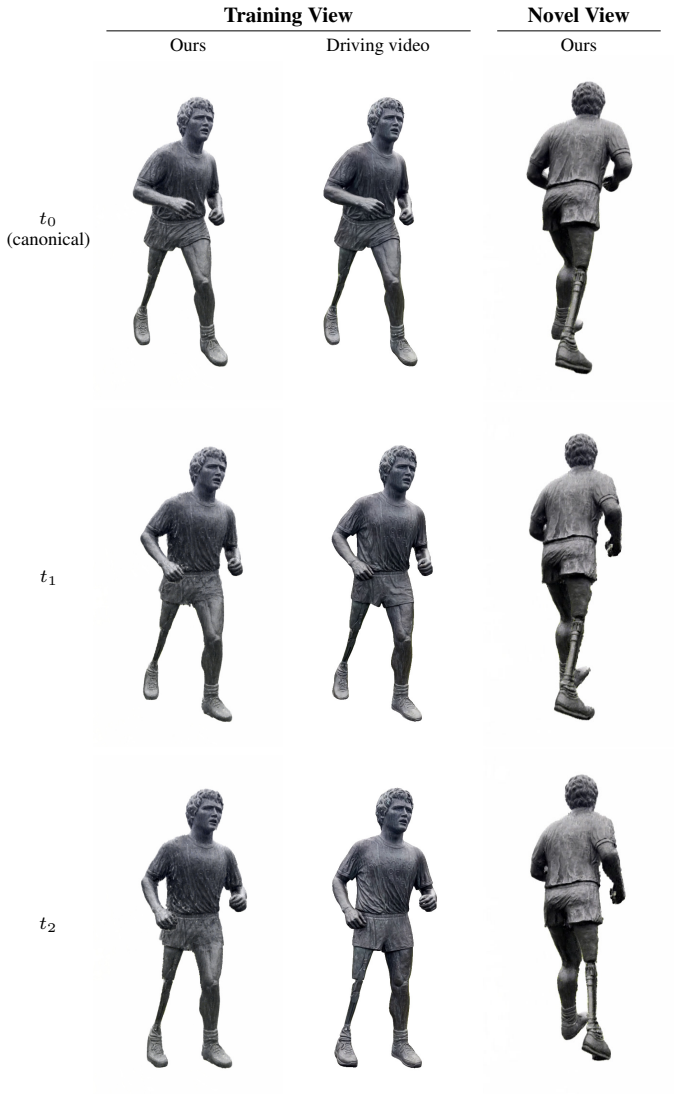

Figure 14: **Our results on the captured statue.** Layout as in Figure 13.

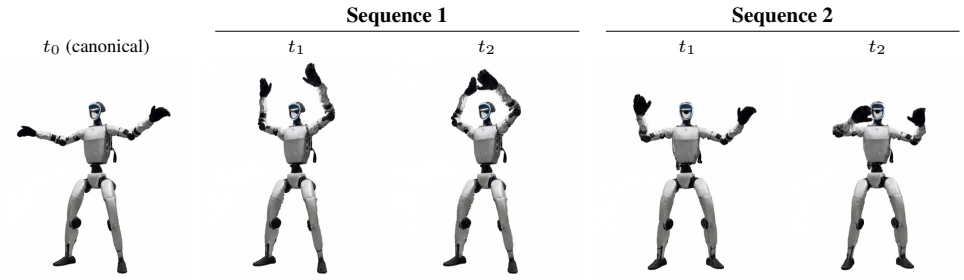

Figure 15: **Novel pose driving on the captured robot.** Our model is driven at test time by two pose sequences not seen during fitting. Columns show the canonical (binding) frame t_0 followed by two frames from each driving sequence. No ground truth is available for the driven poses.

B Additional Ablations

This section reports two additional ablations. Section B.1 compares Phase 1 alone with the full two-phase optimization for every method, and examines what the two phases and the depth supervision each contribute to our method on T-Rex. Section B.2 isolates the rendering representation from any geometry or skinning quality by deforming every method with the ground truth.

B.1 Two-Phase Optimization Ablation

Table 3 compares Phase 1 alone with the full two-phase optimization on all synthetic scenes. Phase 2 raises training-view PSNR for every method ($+1.8$ dB on Mani-GS, $+2.7$ dB on RigGS, $+1.0$ dB on ours), while novel-view PSNR stays within 0.4 dB of Phase 1. For our method this gain comes with visible geometric corrections: Phase 2 fixes per-frame joint rotations that Phase 1 gets wrong, which we examine on T-Rex below.

Table 3: Two-phase ablation on the synthetic scenes. *Phase 1* stops after each method's first training phase; *Full* adds Phase 2 (Section 3.4) on top. **Best** in bold, second-best underlined.

	Training view			Novel view		
Configuration	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Mani-GS (Phase 1)	26.71	0.962	0.037	20.14	0.925	0.073
Mani-GS (Full)	28.53	0.974	**0.026**	20.51	0.927	0.072
RigGS (Phase 1)	26.80	0.963	0.036	20.19	0.924	0.071
RigGS (Full)	**29.52**	**0.975**	0.029	20.37	0.926	0.069
Ours (Phase 1)	28.02	0.964	0.029	**23.70**	**0.946**	**0.044**
Ours (Full)	29.04	0.971	0.027	23.51	0.944	0.045

Our method on T-Rex. We examine our method's two phases more closely on T-Rex, together with the relative-depth term of Phase 1. Figure 16 shows three configurations side by side: Phase 1 with the relative-depth term $\mathcal{L}_{\Delta\text{depth}}$ disabled, the depth-supervised Phase 1, and Full (Phase 1 followed by Phase 2). Without $\mathcal{L}_{\Delta\text{depth}}$, the per-frame joint rotations are supervised only by the rendering loss at a single fixed view, giving the optimizer the freedom to twist rotations off their physically correct paths to improve the 2D pixel match. The no-depth variant therefore fits the training-view silhouette more aggressively, but misaligns the underlying 3D articulation across the sequence; the artifact is most visible at the head, whose rotation locks into a clearly wrong configuration that subsequent training cannot correct. The depth-supervised Phase 1, by contrast, keeps each rotation on a 3D-plausible trajectory, and with the auto-rigged skinning weights still unmodified at this stage (Phase 2 is what refines them), this fidelity to 3D motion comes at the cost of a slightly looser silhouette fit.

Building on the depth-supervised Phase 1, Phase 2 refines both the skinning weights and the per-joint rotation corrections, addressing both ends of the trade at once: with corrected skinning and refined rotations, the deformation matches the ground-truth silhouette without forcing rotations off their 3D-plausible paths. The visible corrections on T-Rex are highlighted in the figure: the head rotation at t_1 and the mouth articulation and tail position at t_2 are all brought closer to the ground truth. The metrics tell the same story: training-view PSNR climbs to its best across all configurations (31.08 dB), and the chamfer distance against the ground-truth mesh improves further from 5.68×10^{-3} at Phase 1 to 4.60×10^{-3} at Full.

B.2 Ground-Truth Deformation

Section 1 attributes the joint-boundary artifacts of Gaussian-splat rigging to the primitive rather than to the deformation. This ablation tests that claim by fixing the deformation to the ground truth and varying only the representation. Every method is deformed with the ground truth of the synthetic subject, so no rig error enters the comparison, and no method is fine-tuned per frame. Mani-GS [7] binds its splats to the ground-truth mesh, so they follow the ground-truth mesh sequence. Mani-GS (isotropic) binds its splats to the same mesh and constrains every splat to be isotropic. RigGS [55] skins its splats directly with linear blend skinning, with no mesh binding, and rotates each splat's covariance by its blended bone rotation. Because the deformation is given by the ground-truth geometry, RigGS's pose-dependent correction head is not needed and is omitted. On fox, RigGS and

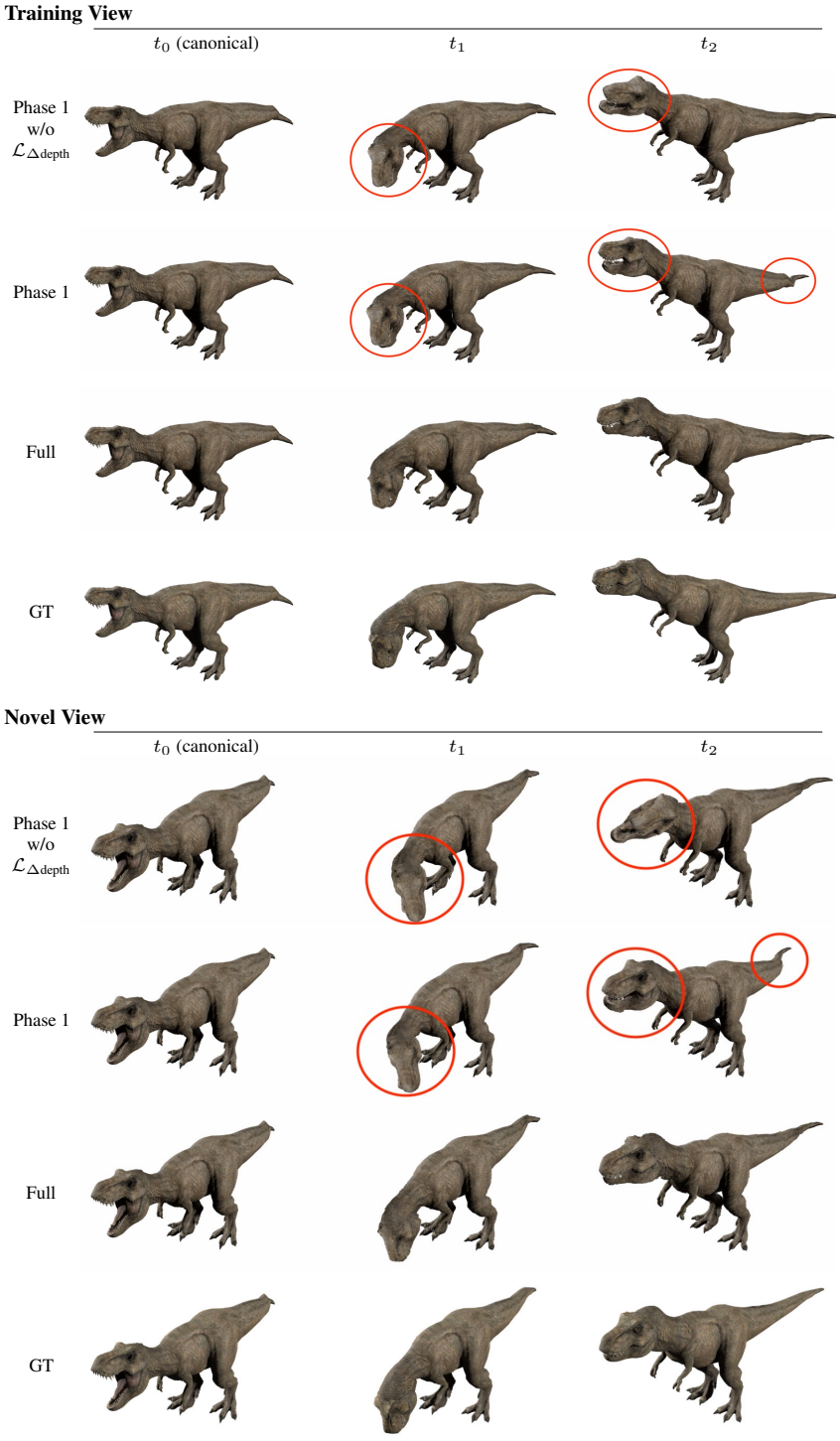

Figure 16: **Two-phase optimization ablation on T-Rex.** Training View is on top and Novel View is on the bottom. Rows are configurations (Phase 1 w/o $\mathcal{L}_{\Delta\text{depth}}$, Phase 1, Full, GT); columns are matched frames with t_0 the canonical (binding) frame and t_1, t_2 progressively articulated.

ours are skinned (Equation (1)) with the ground-truth rig; on T-Rex and Simpsons, which have no ground-truth skeleton, their bones are the faces of the ground-truth mesh sequence. Every splat-based method reorients each splat's covariance with the local deformation and uses more primitives than ours, from about two to fourteen times as many.

At the binding frame all methods render comparably (Figure 17), so any difference at the deformed frame comes from the deformation. At the deformed frame Mani-GS shows artifacts at the joints, around the fox's front leg–body junction, the Simpsons' front shin and the T-Rex's eye and jaw, although the deformation is exact by construction. Isotropic splats do not remove them: the spikes round into blobs, as in Figure 1. RigGS, which drops the mesh, shows them at the same joints. Ours, driven by the same ground-truth deformation, shows no such artifact. With the deformation exact, the artifact survives a change of binding and of covariance parameterisation and disappears only when the primitive changes.

C Sensitivity Analysis

We test how sensitive the two-phase optimization is to its loss weights by scaling one weight group at a time by $\times\frac{1}{3}$ and $\times 3$ around the defaults of Table 6, with all other settings fixed. On T-Rex we vary all six groups; on fox we vary the four groups introduced in Phase 2. Table 4 reports novel-view PSNR. The relative-depth weight is the most sensitive group: reducing it lowers PSNR, while increasing it raises PSNR. The tracking weight has a moderate effect, and the remaining groups change PSNR by at most 0.19 dB.

Table 4: Loss-weight sensitivity. Novel-view PSNR (dB) when one weight group is scaled by $\times\frac{1}{3}$ or $\times 3$, with all other weights at their defaults (Table 6). The two regularizers of each paired group are scaled together. *Spread* is the range over the three settings. Fox varies only the Phase 2 groups.

	T-Rex (default 23.95)			Fox (default 24.13)		
Weight group	$\times\frac{1}{3}$	$\times 3$	Spread	$\times\frac{1}{3}$	$\times 3$	Spread
$\mathcal{L}_{\Delta\text{depth}}$ (relative depth)	23.10	24.65	1.55	–	–	–
$\mathcal{L}_{\text{rot}}$ (rotation reg.)	24.07	24.14	0.19	–	–	–
$\mathcal{L}_{\text{ARAP}}$	24.05	23.92	0.13	24.22	24.27	0.14
$\mathcal{L}_{\text{track}}$ (2D tracking)	24.23	23.69	0.54	23.80	24.07	0.33
Skin-weight reg. (smoothness + sparsity)	23.98	24.10	0.15	24.27	24.14	0.14
Correction reg. (magnitude + temporal smoothness)	24.05	24.08	0.13	24.27	24.24	0.14

D Additional Implementation Details

D.1 Dataset Sequence Lengths

Table 5 lists the per-scene driving-video lengths referenced in Section 4.1.

D.2 PAPR Backbone, Auto-Rig, and Conventions

PAPR backbone. We use a U-Net-free variant of PAPR [62]: the rendering path is purely the proximity-attention aggregation over the top-K nearest points, with no image-space decoder. This keeps the rendering pipeline transparent and ensures that no learned image-space network can compensate for deformation artifacts at render time. Backbone hyperparameters (number of points, attention layer widths, top-K, training schedule) follow the official PAPR release [62]; the renderer is frozen for all downstream optimization in this paper.

Auto-rig pipeline. The auto-rig pipeline runs in two stages on the trained PAPR cloud, and the resulting skeleton and weights are used only as initialization for the two-phase optimization of Section 3.4. (i) We extract a triangle-mesh proxy from the PAPR point cloud using the pre-trained, off-the-shelf meshing model OffsetOPT [21], run with its public release defaults; the proxy serves only the next step. (ii) We feed this proxy to Puppeteer [41] with default settings, obtaining a skeleton of B bones and per-vertex skinning weights on the proxy mesh. We then transfer the per-vertex skinning weights to each PAPR point by k-nearest-neighbour inverse-distance weighting (IDW) in

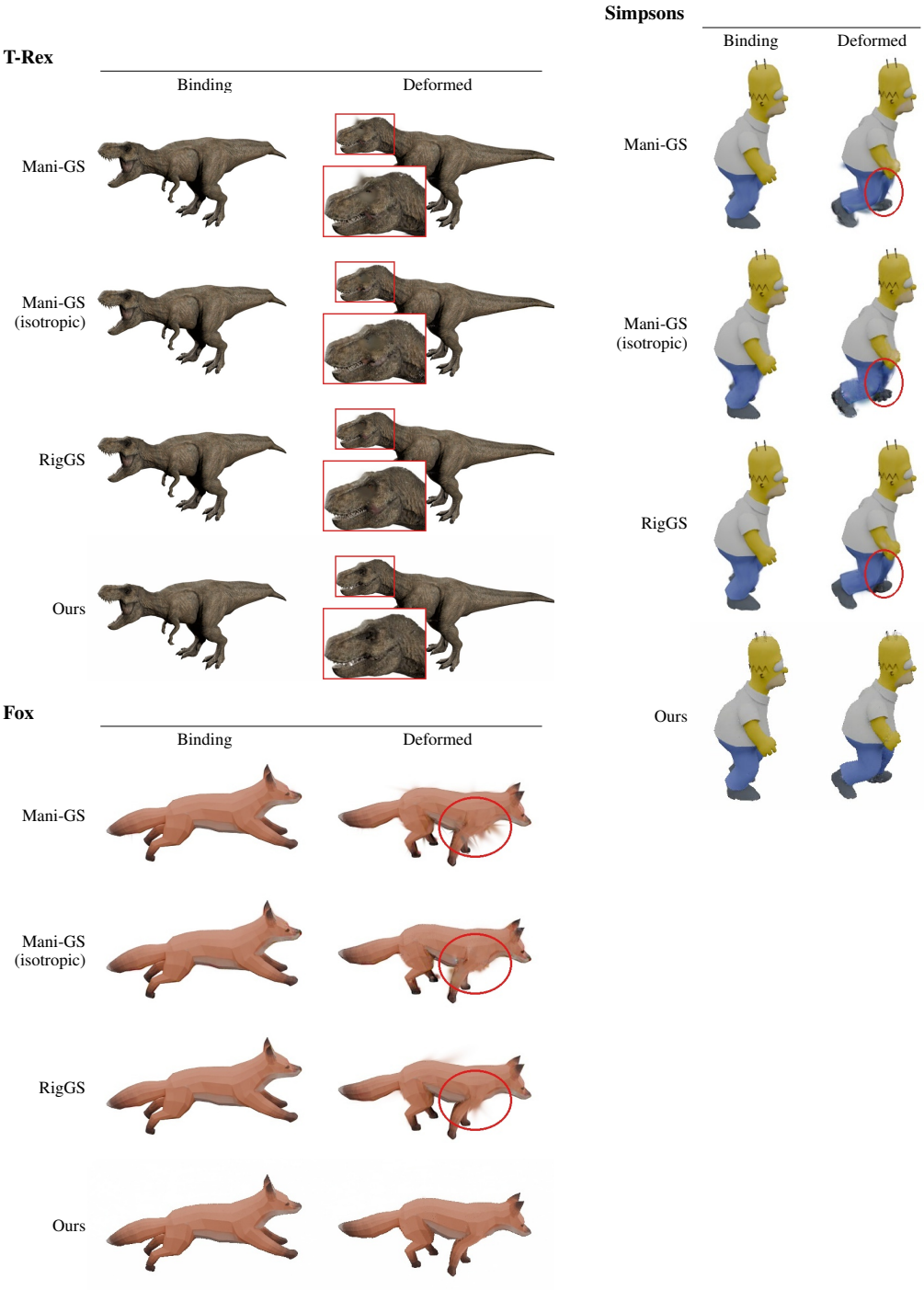

Figure 17: **Splat variants under ground-truth deformation.** Every method is deformed with the ground truth, so no rig error enters the comparison. Rows are methods; columns are the canonical *Binding* frame and one *Deformed* frame. Every splat-based method shows joint-boundary artifacts at the deformed frame; ours does not. Best viewed zoomed in.

Table 5: Per-scene driving-video length, in frames. Synthetic scenes are rendered at 512×512 and ZJU-MoCap frames are used at 512×512; the captured scenes and their generated driving videos are at 960×540.

Scene	Frames
Synthetic	
T-Rex	40
Simpsons	34
Fox	24
Wolf	18
Spider	16
ZJU-MoCap	
313	25
390	24
Captured scenes	
Robot	36
Statue	18

the canonical pose, with $k = 6$ by default, yielding $\mathbf{w}_i \in \mathbb{R}^B$. The proxy mesh is discarded after this transfer and participates in neither deformation nor rendering.

Driving camera. The driving camera's intrinsics and extrinsics are known and held fixed throughout optimization. For synthetic scenes it is the rendering camera, and for ZJU-MoCap it is the provided calibration of the driving camera. For the captured scenes it is the calibrated camera of the canonical view used to generate the driving video, and we assume the generated video preserves this fixed viewpoint.

Rotation parametrization. All bone rotations $\mathbf{R}_b^t$ in the optimization, both the Phase 1 frame-wise rotations and the Phase 2 per-joint corrections, are parametrized as axis-angle vectors $\boldsymbol{\omega}_b^t \in \mathbb{R}^3$ (direction giving the rotation axis, magnitude the rotation angle) and converted to $SO(3)$ on the fly via the Rodrigues exponential map $\mathbf{R}_b^t = \exp([\boldsymbol{\omega}_b^t]_\times)$, where $[\,\cdot\,]_\times$ denotes the skew-symmetric matrix of a 3-vector. No per-bone translation is optimized: only the root carries the global translation $\mathbf{t}^t \in \mathbb{R}^3$, and non-root bone offsets are fixed at bind time. Forward kinematics composes these rest-pose offsets with the bone rotations along the skeleton tree to yield the per-bone $SE(3)$ transforms $\mathbf{T}_b(\mathbf{R}^t)$ used in Equation (1).

Skinning-weight parametrization. The skinning weights $\mathbf{w}_i \in \mathbb{R}^B$ used in Equation (1) are not optimized directly; instead we store unconstrained per-point logits $\mathbf{l}_i \in \mathbb{R}^B$ and recover the active weights at every forward pass by a per-point softmax over bones, $\mathbf{w}_i = \mathrm{softmax}(\mathbf{l}_i)$. This guarantees $w_{i,b} \geq 0$ and $\sum_{b=1}^{B} w_{i,b} = 1$ at every step without explicit projection. The logits are initialized from the auto-rig output as $\mathbf{l}_i = \log \tilde{\mathbf{w}}_i$, where $\tilde{\mathbf{w}}_i$ is the IDW-transferred Puppeteer weight clamped to a small floor for numerical stability so the logarithm is finite. Phase 1 keeps $\mathbf{l}_i$ frozen at this initialization; Phase 2 optimizes $\mathbf{l}_i$ jointly with the correction MLP, with the regularizers tabulated in Table 6 acting on the resulting $\mathbf{w}_i$.

D.3 Baseline Implementation

Mani-GS rig pipeline. The Mani-GS baseline binds splats to a triangle-mesh proxy extracted from its trained first-stage 3DGS, following Mani-GS's own meshing procedure [7]. We auto-rig this proxy with Puppeteer [41] (the same auto-rigger used in our method) to obtain a skeleton and per-vertex skinning weights, and the resulting skinning weights are refined as part of the shared Phase 2 schedule.

Mani-GS modifications. The Mani-GS baseline runs through the same two-stage process as our method, with three additional modifications. First, we disable the per-triangle adaptive scaling throughout training; left on, it drives covariance blow-up that Phase 2 binding-rotation refinement only partially suppresses, leaving visibly noisier renderings (see Section D.4). Second, in Phase 2 we

additionally fine-tune each splat's binding rotation and offset against the driving-view loss, freeing the bindings from being fit only to the canonical frame. Third, we add a foreground-mask loss to its rendering objective for added silhouette supervision. Together these modifications yield the strongest Mani-GS baseline we could obtain for this fixed-camera setting.

RigGS. RigGS [55] originally jointly reconstructs the canonical object and its rig from a monocular video in which the camera moves relative to the subject. Our setting instead begins with an existing canonical reconstruction and estimates its motion from a single fixed-viewpoint video. RigGS's paper notes that the method "faces challenges with videos captured by a fixed camera" and that improved results "can be achieved when the camera is allowed to move." On ZJU-MoCap, it accordingly cycles through six cameras to simulate a monocular video with camera movement. Its complete reconstruction-and-rigging pipeline is therefore not directly applicable under our input assumptions.

We therefore adapt RigGS to our pipeline, following its released implementation. RigGS reconstructs the canonical splats and extracts a skeleton with skinning weights derived from point-to-bone distances, both from the driving video itself. In our setting these are available as inputs, so we provide them rather than have them inferred: the canonical splats and mesh reconstructed from the first frame, and the same canonical skeleton and skinning field the other methods receive, carried to each splat through its binding face. The deformation model is then RigGS's own. Each splat is skinned directly by linear blend skinning, taking its position from the blended bone transforms and its orientation from the blended joint quaternions, with scale left unchanged, and RigGS's pose-conditioned detail-deformation head adds a per-splat world-space translation after skinning. We use the released head at its published width, depth, and positional encoding, and its offset regularizer at the published weight including the canonical-frame term, and the skinning weights remain trainable as in RigGS. The canonical splats are not reinitialized, and the training schedule and evaluation match the other methods.

D.4 Mani-GS Adaptive Scaling Ablation

In Section D.3 we noted that the Mani-GS baseline has its per-triangle adaptive scaling disabled throughout training. Mani-GS rescales each splat's scaling vector at every frame in proportion to its bound triangle's deformed edge lengths, so the splat tracks the triangle's shape change [7]. On thin or near-degenerate triangles, this rescaling drives the splat covariance to extreme values and produces visible artifacts under articulation. Phase 2 binding-rotation refinement reduces some of these artifacts by reorienting affected splats, but cannot address them as effectively as disabling adaptive scaling outright.

Figure 18 compares the two configurations on Simpsons and fox. Both runs use our full two-phase optimization; only the adaptive-scaling toggle differs. With scaling enabled, articulated frames retain visible artifacts even after Phase 2; we therefore disable it to obtain the stronger Mani-GS baseline used throughout Section 4.

D.5 Optimization Details

This section gathers the loss weights, regularizers, and schedules used in the two-phase optimization of Section 3.4. Both phases use the rendering term $\mathcal{L}_{\text{rgb}} = 0.8 \cdot \text{L1} + 0.2 \cdot (1 - \text{SSIM})$ on the rendered patch, augmented with an LPIPS term at weight 0.01. The remaining weights are summarized in Table 6.

On wolf, Phase 2 keeps the composed-rotation regularizer at its Phase 1 weight (0.1 instead of 0.01). On ZJU-MoCap subject 390, the Δ-depth weight is 2 in Phase 1 and 8 in Phase 2.

Optimizer and learning-rate schedule. Both phases use Adam with cosine-annealing learning-rate schedules. In Phase 1, each frame is fit with a single learning rate over its joint rotations and root translation, annealed from 5×10^{-3} to 1×10^{-4}. In Phase 2, the optimizer uses per-parameter-group learning rates: $3{\times}10^{-4}$ for the correction MLP and $1{\times}10^{-2}$ for the skinning-weight logits. By default, both groups are cosine-annealed to $1{\times}10^{-5}$ over $30{,}000$ Phase 2 iterations. On ZJU-MoCap, Phase 2 runs $20{,}000$ iterations with a skinning-weight learning rate of 1×10^{-4}; on wolf, the correction MLP uses a learning rate of 1×10^{-4}.

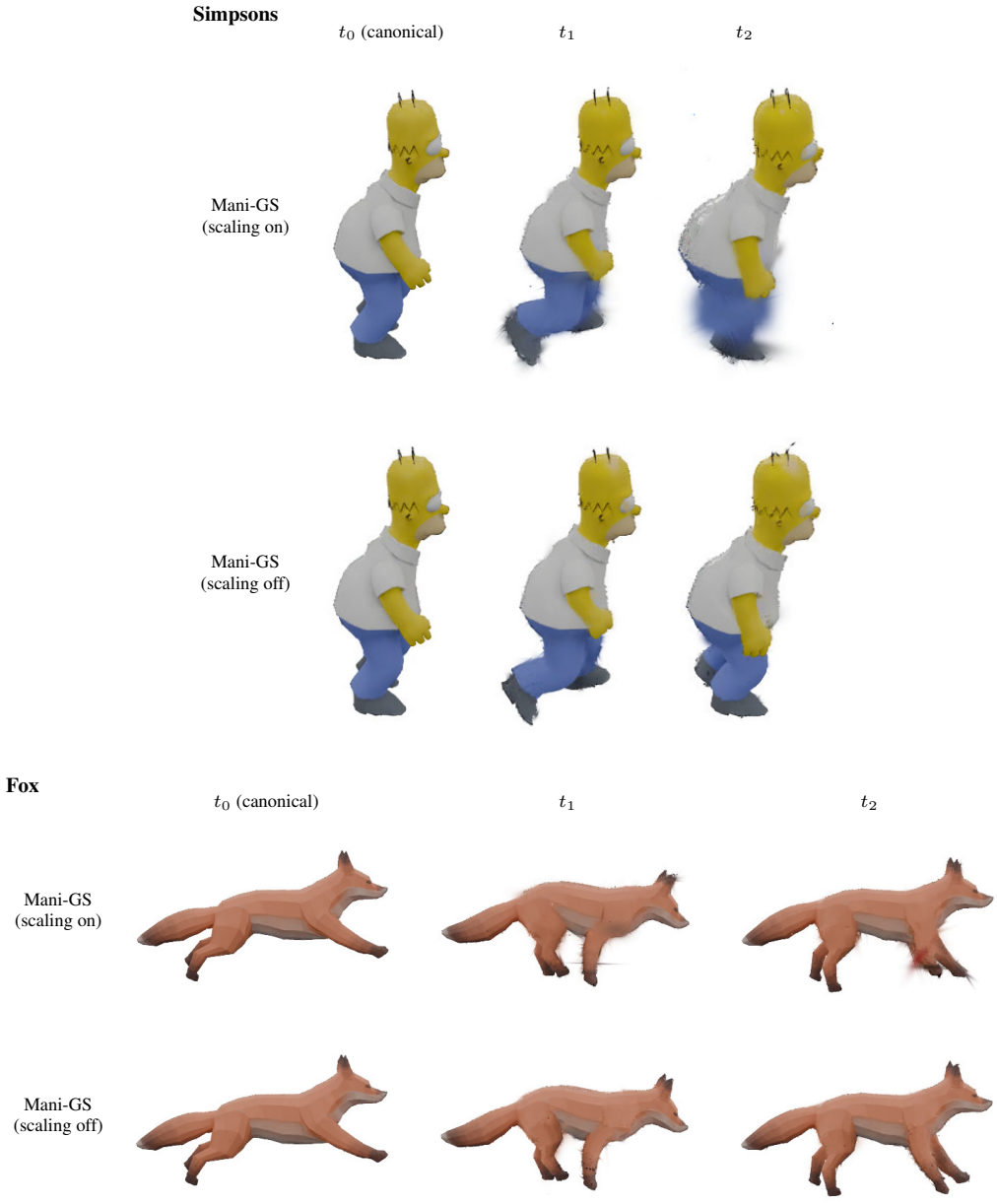

Figure 18: **Mani-GS adaptive-scaling ablation.** Simpsons (top) and fox (bottom), each shown as two rows comparing the default Mani-GS configuration (per-triangle adaptive scaling *on*) with the configuration used in the main paper (scaling *off*). Both rows share our full two-phase optimization; only the scaling toggle differs. Columns are matched training-view frames, with t_0 the canonical (binding) frame and t_1, t_2 progressively articulated. Artifacts produced by the most degenerate splats remain visible with scaling enabled even after Phase 2.

Table 6: Loss weights for the two-phase optimization of Section 3.4. Entries marked "decays $a \to 0$" follow a schedule that decays from the listed value to zero over the course of Phase 2. Dashes indicate the term is not active in that phase. These are the default values; per-scene deviations are listed below the table.

Term	Phase 1	Phase 2
$\mathcal{L}_{\text{rgb}}$ $(0.8 \cdot \text{L1} + 0.2 \cdot (1 - \text{SSIM}))$	1.0	1.0
LPIPS	0.01	0.01
$\mathcal{L}_{\Delta\text{depth}}$ (relative depth)	1.0	decays $1.0 \to 0$
$\mathcal{L}_{\text{rot}}$ (descendant-weighted rotation reg.)	0.1	—
$\mathcal{L}_{\text{track}}$ (2D tracking)	—	decays $0.005 \to 0$
$\mathcal{L}_{\text{ARAP}}$ (k-NN distance preservation, k=8)	—	500
Composed-rotation magnitude reg. (Phase $1 +$ correction)	—	0.01
Correction L2 magnitude	—	5×10^{-4}
Correction temporal smoothness	—	0.1
Skin-weight smoothness (k-NN graph)	—	0.03
Skin-weight sparsity	—	0.01

Δ-depth supervision. At each optimization step the relative-depth loss $\mathcal{L}_{\Delta\text{depth}}$ is computed on a random subsample of 4096 canonical points. The same points are projected through the binding-frame and current-frame cameras to obtain camera-space depths z_i^0, z_i^t and pixel coordinates p_i^0, p_i^t; binding-frame quantities are detached so that gradients flow only through the deformation. Monocular depths $\tilde{m}_i^0, \tilde{m}_i^t$ are bilinearly sampled from the Depth Anything 3 maps at p_i^0 and p_i^t (we use the metric-depth variant of Depth Anything 3, so $\tilde{m}$ is in metric units rather than affine-invariant disparity, and the linear scale alignment below is well-posed), and we form the per-point deltas $\Delta z_i = z_i^t - z_i^0$ and $\Delta m_i = \tilde{m}_i^t - \tilde{m}_i^0$. We retain only points that lie in front of the camera and project inside the image at *both* frames, and whose current-frame projection lands on a foreground-mask pixel; the foreground requirement restricts the depth signal to points that currently contribute to the rendered silhouette. The per-frame scale is computed in closed form on the retained samples, $s_t = \langle \Delta z, \Delta m \rangle / \|\Delta m\|^2$, under `no_grad`, so it acts as a constant during backpropagation; it is recomputed fresh each step on the new subsample and never persists as state. The loss is then $\mathcal{L}_{\Delta\text{depth}} = \frac{1}{n} \sum_i (\Delta z_i - s_t \Delta m_i)^2$, where n is the number of retained samples. Phase 2 uses the same construction with $\lambda_{\Delta\text{depth}}$ decayed to zero according to Table 6.

Correction network. The Phase 2 correction $f_\theta(t)$ is a small temporal MLP that takes a positional encoding of the normalized frame index t/T as input and outputs a per-joint axis-angle offset, added to the Phase 1 rotation of that joint before LBS. The positional encoding uses sine and cosine bases with frequency exponents 0 through 10 (eleven bands), giving an input of dimension $1 + 2 \cdot 11 = 23$. The MLP has two hidden layers of width 128 with ReLU activations and a linear output head producing $3B$ values reshaped to per-joint axis-angle vectors; the output layer is zero-initialized so f_θ produces zero correction at the start of training. The Phase 1 rotations and the PAPR network remain frozen throughout Phase 2.

Correction regularizers. The correction outputs are regularized with an L2 penalty on their magnitude (weight 5×10^{-4}) and a temporal-smoothness penalty on consecutive-frame differences (weight 0.1); together these keep the corrections small and slowly varying.

Skin-weight regularizers (Phase 2). The Phase 2 skin-weight refinement is regularized by two terms acting on the active weights $\mathbf{w}_i = \text{softmax}(\mathbf{l}_i)$. The smoothness term penalizes disagreement among k-nearest canonical neighbours,

$$\mathcal{L}_{\text{smooth}} \;=\; \frac{1}{Nk} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}(i)} \left\| \mathbf{w}_i - \mathbf{w}_j \right\|_1, \tag{2}$$

on the same canonical k-NN graph used by $\mathcal{L}_{\text{ARAP}}$ ($k = 8$). The sparsity term is the per-coordinate negative binary entropy,

$$\mathcal{L}_{\text{sparse}} \;=\; -\frac{1}{NB} \sum_{i=1}^{N} \sum_{b=1}^{B} \Big[w_{i,b} \log w_{i,b} + (1 - w_{i,b}) \log(1 - w_{i,b}) \Big], \tag{3}$$

which pushes each $w_{i,b}$ toward 0 or 1 and so toward fewer effective bones per point.

Descendant-weighted rotation regularizer. Joint rotations are penalized in proportion to how many child joints they propagate to, so that bones high in the kinematic tree (which carry their entire subtree with them) are encouraged to be quieter than leaf bones. Let $n_{\text{desc}}(b)$ denote the number of descendants of bone b in the Puppeteer-returned kinematic tree (zero for leaves), and define

$$\alpha_b \;=\; \frac{1 + n_{\text{desc}}(b)}{1 + n_{\text{desc}}(\cdot)}, \tag{4}$$

normalized so $\overline{\alpha} = 1$ across the B bones. Phase 1 regularizes the per-frame axis-angle magnitudes,

$$\mathcal{L}_{\text{rot}}^{(1)} \;=\; \frac{1}{B} \sum_{b=1}^{B} \alpha_b \left\| \boldsymbol{\omega}_b^t \right\|_2^2. \tag{5}$$

Phase 2 regularizes the *composed* rotation (Phase 1 base $\mathbf{R}_b^t$ rotated by the per-frame correction $\mathbf{R}_b^{(c,t)}$ produced by f_θ) via a Frobenius-norm proxy,

$$\mathcal{L}_{\text{rot}}^{(2)} \;=\; \frac{1}{TB} \sum_{t=1}^{T} \sum_{b=1}^{B} \alpha_b \left\| \mathbf{R}_b^t \mathbf{R}_b^{(c,t)} - \mathbf{I} \right\|_F^2, \tag{6}$$

which is monotonically related to rotation magnitude and avoids a matrix logarithm.

ARAP distance-preservation term. The as-rigid-as-possible term $\mathcal{L}_{\text{ARAP}}$ in Phase 2 acts on a fixed canonical k-nearest-neighbour graph over the PAPR points, with $k = 8$ neighbours per point computed once at the binding pose. For each canonical edge (i, j) we penalize the squared change in pairwise distance under the deformed configuration, $\sum_{(i,j)} \left(\|\mathbf{p}_i^t - \mathbf{p}_j^t\| - \|\mathbf{p}_i^0 - \mathbf{p}_j^0\| \right)^2$, weighted by the loss coefficient in Table 6. The graph is canonical (not recomputed per frame), so the term anchors local geometry to the binding-pose neighbourhood structure throughout the sequence.

Sequential frame fitting. In Phase 1 we fit the per-frame rotations sequentially from the binding frame: each frame $t{+}1$ is initialized from the optimized rotations of frame t, exploiting temporal smoothness in the driving video so that no frame is optimized from scratch. Each frame is optimized for 2,000 iterations.

Hardware and runtime. All optimization runs on a single NVIDIA RTX 3090 GPU. For a 16-frame sequence, Phase 1 takes approximately one hour, and Phase 2 takes another hour for its 30,000 iterations.

D.6 Attention-Guided Track Seeding

The 2D tracking loss $\mathcal{L}_{\text{track}}$ in Phase 2 requires a set of (point, query-pixel) pairs at the canonical frame: each selected canonical point is propagated forward in time by CoTracker from its query pixel, and the resulting trajectory supervises the projection of the LBS-deformed point. The construction below chooses which points to supervise and where to query them.

Off-ray issue with naive projection. The simplest construction, projecting all canonical points through the camera and using those pixel positions as queries, is problematic for an attention-based renderer like PAPR. A point can contribute to a pixel without lying exactly on its ray, and conversely a point's geometric projection is not necessarily where it visually contributes. Projection-only queries therefore include points the renderer effectively ignores at the canonical view, points whose projection lands a pixel or two off the visible surface, and silently-occluded back-facing points. Each of these injects label noise into the tracking loss.

Attention-dominant selection. We instead select queries from the renderer's own attention. We run PAPR over the canonical-frame image (tiled in patches for memory) to obtain, for each foreground pixel, the indices and attention weights of the top-K points selected by proximity attention. For each foreground pixel we take the argmax over its top-K attention weights to identify the dominant point:

the single point that most explains that pixel. Pixels whose top-1 attention falls below a threshold (we use 0.05) are discarded as smeared/low-confidence, and the remaining indices are deduplicated to yield the set of dominant points for the canonical view. Each selected point is then projected through the canonical camera; by construction this projection lies inside the rendered silhouette near a pixel the point dominates, and that 2D position is the CoTracker query for the point. Tracks and the corresponding 3D point indices are persisted together so that the optimizer knows which canonical point each trajectory supervises.

Mani-GS query seeding. For Mani-GS we seed CoTracker queries by projecting the canonical mesh vertices through the same canonical view, restricting to those inside the foreground mask, and uniformly subsampling on the image plane by grid binning to a comparable query count. The CoTracker model, temporal window, canonical view, and query count are thus matched to ours; the seeding differs only in per-pixel point selection (mesh-vertex grid sampling vs. attention dominance).

D.7 Licenses for Existing Assets

We use the following existing code, model, and dataset assets, each cited at first use in the main paper. We list each asset with the license under which it is released (or, where no open-source license applies, the terms under which we use it). All assets are used within their stated terms.

- **PAPR** [62] (code and backbone, `https://github.com/zvict/papr`): MIT License.
- **Mani-GS** [7] (code, used as baseline, `https://github.com/gaoxiangjun/Mani-GS`): released under the Gaussian-Splatting research license inherited from the original 3D Gaussian Splatting code, which restricts use to non-commercial research.
- **RigGS** [55] (code, used as baseline, `https://github.com/yaoyx689/RigGS`): GNU General Public License v3.0 (GPL-3.0).
- **Puppeteer** [41] (auto-rigging model, `https://github.com/Seed3D/Puppeteer`): Apache License 2.0.
- **OffsetOPT** [21] (meshing model, `https://github.com/EnyaHermite/OffsetOPT`): no license file is provided in the repository; we use it under fair-use research terms and credit the authors via citation.
- **Depth Anything 3** [27] (monocular depth predictor, `https://github.com/ByteDance-Seed/depth-anything-3`): Apache License 2.0.
- **CoTracker 3** [13] (2D point tracker, `https://github.com/facebookresearch/co-tracker`): CC BY-NC 4.0 (non-commercial); used here for non-commercial research.
- **Objaverse** [6] (synthetic 3D assets): ODC-By 1.0 for the dataset; per-asset licenses vary by source, and we use only assets whose individual licenses permit research use.
- **Kling 3.0** [5] (image-to-video model used to synthesize real-world driving videos): a commercial closed-source service; we use it under its standard terms of service. No open-source license applies.

E Broader Impact

RigPAPR enables turning a real-world capture into a rigged, re-posable asset directly from the capture itself, without an intermediate re-modeled mesh, which we expect to benefit animators, accessibility tools, and creators without specialized infrastructure. The same accessibility could be repurposed to re-animate captures of real people without consent. This concern is shared with related neural avatar work; practitioners deploying such pipelines on human subjects should pair them with consent-aware capture, content provenance tags, and downstream detection.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See the abstract and Section 1; the claims are supported by Sections 3 and 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the "Limitations" paragraph in Section 5 (Final Discussion and Conclusion).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper presents an empirical method and contains no theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 3 and Section D in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code and sample data are available at `https://github.com/niopeng/RigPAPR`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 4.1 and Section D in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We report aggregate point estimates across scenes (Tables 1 and 3). Each method is fit deterministically per scene from a fixed initialization; multi-seed runs were not conducted due to limited computational resources.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See the "Hardware and runtime" paragraph in Section D.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics, and the research conforms with it in all respects.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Section E ("Broader Impact") in the appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper releases neither a pretrained generative model nor a scraped dataset, so no high-misuse-risk asset requires safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Section D.7 in the appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper involves no crowdsourcing and no research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper involves no research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Large language models are not used as a methodological component of this work.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.