ARTSPLAT: FEED-FORWARD ARTICULATED 3D GAUSSIAN SPLATTING FROM SPARSE MULTI-STATE VIEWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Articulated object reconstruction requires simultaneous inference of geometry and underlying articulation structure. Existing methods based on NeRF and 3D Gaussian Splatting (3DGS) typically rely on dense views or strong priors (*e.g.*, depth maps, joint types, predefined number of joints), requiring costly per-object optimization. In this paper, we propose ArtSplat, the first feed-forward 3DGS framework that reconstructs both geometry and joint parameters from sparse multi-view images across multiple articulation states in a single forward pass. Our novel per-pixel joint map representation enables integration of joint parameter estimation into the feed-forward pipeline. We further propose a Cross-State Attention (CSA) mechanism with state tokens, which effectively captures discrete motion across input states. Comprehensive experiments demonstrate competitive geometry reconstruction, accurate joint estimation and cross-dataset generalizability of ArtSplat on both single- and multi-joint objects. Compared to the per-object optimization baselines, ArtSplat achieves 389 times faster inference time.

1 INTRODUCTION

An *articulated object* consists of rigid parts connected by joints, allowing the parts to move relative to one another. These joints come in two types: revolute joints, which rotate about an axis (*e.g.*, a laptop lid or a cabinet door), and prismatic joints, which translate along an axis (*e.g.*, a drawer). Representing such an object, therefore, requires not only the geometry of each part but also the joint parameters that determine how those parts move. The *articulated object reconstruction* is a task to recover both of these from observed samples. Unlike static reconstruction, the resulting representation should be able to move along the recovered joints into any valid pose, called an *articulation state*, of its parts. This makes it directly applicable to downstream tasks such as manipulation policy learning, sim-to-real asset generation, and digital twin construction.

Early work on articulated object reconstruction adopts implicit representations such as neural radiance fields (Mildenhall et al., 2021), recovering both geometry and joint parameters from multi-state observations (Jiang et al., 2022; Liu et al., 2023; Tseng et al., 2022). Recently, 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) is widely adopted to leverage its explicit primitives and faster rendering to improve reconstruction quality (Guo et al., 2025; Kim et al., 2025; Lin et al., 2025; Liu et al., 2025; Shen et al., 2025; Wu et al., 2025; Yu et al., 2025). Both approaches, however, rely on test-time optimization, still requiring fitting each object from scratch over dense (typically 50+ per articulation state) multi-view captures. These methods do not scale well to a large number of novel objects, since both the capturing and optimization steps should be repeated for each new object, requiring tens of minutes or even hours per object.

A natural alternative is to predict the reconstruction through a single feed-forward pass, which has been successfully applied to static scenes, where the scene does not change across input views, by transformer-based geometry models (Wang et al., 2024; Leroy et al., 2024; Wang et al., 2025a; Yang et al., 2025; Zhang et al., 2025b) and feed-forward 3DGS variants (Charatan et al., 2024; Chen et al., 2024; Xu et al., 2025; Ye et al., 2025; Smart et al., 2024; Jiang et al., 2025), and to dynamic scenes by their dynamic counterparts (Zhang et al., 2025a; Wang et al., 2025b; Chen et al., 2025), where an ordered sequence of views are given with only small motion. However, articulated objects do not fit either case. The input views are *not static* as they contain parts with non-trivial movements across their articulation states, but they are still *unordered*. Given a few RGB views containing multiple

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

articulation states, the task aims to recover geometry and infer how the parts move at the same time from limited observations, and this task is trivially solved by neither direction.

In order to tackle these challenges, we propose ArtSplat, a feed-forward model that predicts 3D Gaussians with joint parameters from sparse uncalibrated views captured at two or more articulation states. A feed-forward model is ideal for this task, since it avoids fitting each object at test time and enables fast reconstruction from sparse input views. However, existing feed-forward models are not directly applicable, since they treat the inputs as either static views or an ordered sequence with small motion, whereas views of articulated objects are unordered and may contain large part movements across articulation states. Incorporating 3DGS, our proposed architecture allows us to handle both the object geometry and articulation states with a single feed-forward model; specifically, the explicit Gaussian primitives allow direct application of predicted joint transformations, while splatting enables efficient rendering of the articulated reconstruction.

The core of our design is the joint prediction module. Joint parameters are not visible in any single view but emerge only from differences between unordered articulation states. To capture such inter-state motion, each state is paired with its own learnable *state* token, and a cross-state attention block allows each token to attend to *patch* tokens of another state, enabling the tokens to capture inter-state motion. We employ a dual-branch decoder structure, one for state-invariant joint properties like joint type, axis, and pivot, while the other for state-variant motion like rotation angle and translation distance. Conditioned on the state tokens, they decode the patch tokens into a per-pixel joint map. The resulting joint map encodes joint type, axis, pivot, and motion magnitude at every pixel. Combined with 3D Gaussian primitives, these per-pixel parameters articulate the Gaussians into any target articulation state for rendering.

In summary, our contributions are as follows:

- We propose ArtSplat, the first feed-forward 3DGS framework for articulated object reconstruction that jointly predicts 3D Gaussians and joint parameters from uncalibrated RGB views, eliminating the costly test-time optimization required for each object by prior methods.
- We design a joint prediction module that explicitly captures discrete inter-state motion, enabling accurate joint estimation from sparse views.
- Extensive experiments show that our model matches or outperforms prior baselines in reconstruction and joint-estimation quality, while running 389 times faster than prior works.

2 Related Work

Articulated object reconstruction. Early implicit methods (Liu et al., 2023; Jiang et al., 2022; Tseng et al., 2022) optimize per-object neural fields from multi-view observations across articulation states. Subsequent 3D Gaussian Splatting (3DGS) approaches (Guo et al., 2025; Liu et al., 2025; Wu et al., 2025; Shen et al., 2025; Yu et al., 2025) have improved rendering fidelity, yet many still rely on strong priors such as depth, joint types (Lin et al., 2025) or predefined number of joints (Liu et al., 2025; Wu et al., 2025). Kim et al. (2025) have attempted to relax these assumptions by inferring joint types and counts directly from RGB images. Across both paradigms, however, most methods remain formulated around a single joint, handling multi-part objects through iterative or sequential processing. Moreover, they typically require dense multi-view samples and rely on per-object optimization, incurring substantial computational overhead that limits practical applicability.

Generalizable feed-forward reconstruction. A parallel line of work focuses on eliminating per-scene optimization via feed-forward inference, directly predicting 3D structure from input images. For static scenes, transformer-based models such as DUSt3R (Wang et al., 2024) and its successors (Leroy et al., 2024; Wang et al., 2025a; Yang et al., 2025; Zhang et al., 2025b) have demonstrated remarkable capabilities in recovering 3D geometry from sparse views. Similarly, 3DGS-based feed-forward methods (Charatan et al., 2024; Chen et al., 2024; Xu et al., 2025; Ye et al., 2025; Smart et al., 2024; Jiang et al., 2025) enable efficient reconstruction in a single pass. However, these methods assume that the object remains static across all input views. For multi-state inputs of articulated objects, they cannot separate viewpoint variations from inter-state part motions, producing ghosting artifacts. While dynamic scene models like MonST3R (Zhang et al., 2025a) and others (Chen et al., 2025; Wang et al., 2025b) extend this to moving geometry, they typically assume dense temporal correspondence or negligible, gradual displacements across frames. Articulated objects, in contrast, often appear in a

few discrete states with large-magnitude motion, motivating a dedicated feed-forward framework tailored to such discrete, large-displacement state transitions.

Feed-forward articulated reconstruction. Most closely related to our work are ART (Li et al., 2026) and LARM (Yuan et al., 2025), which have established articulated object reconstruction in a single forward pass. ART uses a transformer with fixed-part slots to decompose objects via SDF-based volume rendering, conditioned on Plücker ray embeddings that require known camera poses. While providing structured outputs, it is limited by a predefined part count and suffers from a heavy computational bottleneck due to dense ray sampling at rendering. LARM, on the other hand, employs a transformer decoder to synthesize novel views conditioned on camera poses and joint states. Rendering an image thus requires a full forward pass through the transformer from the target pose and state, so its inference cost scales linearly with the number of desired views and states. In contrast, our method directly predicts 3D Gaussians and joint parameters from uncalibrated RGB images in a single forward pass, enabling real-time rendering of articulated objects.

3 Problem Formulation

The **articulated object reconstruction** task takes S articulation states with V sparse views for each as inputs, giving a total of $N = V \cdot S$ pose-free RGB images $\mathcal{I} = \{\mathbf{I}_{i,s} \in \mathbb{R}^{H\times W\times 3}\}_{i=1,\dots,V,\, s=1,\dots,S}$ of an articulated object with one or more movable parts at resolution $H \times W$. An articulation between two states is either *revolute*, a rotation by angle θ around an axis at a pivot point (rotation requires axis location, not just direction), or *prismatic*, a displacement by d along an axis.

The model is expected to encode the N input images and to produce predictions for multiple heads, *e.g.*, camera, depth, joint, and Gaussian, in a single forward pass:

- The *camera* head outputs the per-image camera parameters $\boldsymbol{\pi}_{i,s} \in \mathbb{R}^9$, further decomposed into translation $\mathbf{t} \in \mathbb{R}^3$, unit quaternion $\mathbf{q} \in \mathbb{R}^4$, and camera field-of-view $\mathbf{f} \in \mathbb{R}^2$.
- The *depth* head outputs the per-pixel depth $\mathbf{D}_{i,s} \in \mathbb{R}^{H\times W}$ and the confidence maps $\mathbf{C}_{i,s} \in \mathbb{R}^{H\times W}$. Note that the depth map determines the Gaussian means, and when unprojected with the predicted camera $\boldsymbol{\pi}_{i,s}$, it produces a point map $\mathbf{P}_{i,s} \in \mathbb{R}^{H\times W\times 3}$.
- The *Gaussian* head predicts per-Gaussian attributes, including scale $\mathbf{s} \in \mathbb{R}^3$, rotation quaternion $\mathbf{r} \in \mathbb{R}^4$, opacity $\alpha \in \mathbb{R}$, and view-dependent color $\mathbf{c} \in \mathbb{R}^{3\times(L+1)^2} = \mathbb{R}^{75}$ represented with spherical harmonics of degree $L = 4$, yielding 83 parameters per Gaussian. Together with the depth-derived means, they construct the set of Gaussians, $\mathcal{G} = \{g_n\}_{n=1}^{H\times W\times N}$.
- The *joint* head predicts a per-pixel joint map $\mathbf{J}_{i,s} \in \mathbb{R}^{H\times W\times 11}$, with channels for the joint type logits over *static*, *revolute*, and *prismatic* (3), axis direction $\mathbf{a}$ (3, unit-norm), pivot location $\mathbf{p}$ (3), revolute angle θ (1), and prismatic displacement d (1). The Gaussian transformation is conditioned on the predicted joint type: *revolute* pixels use rotation parameters $\{\mathbf{a}, \mathbf{p}, \theta\}$, *prismatic* pixels use translation parameters $\{\mathbf{a}, d\}$, and *static* pixels remain unchanged.

As illustrated in Fig. 1, a differentiable transformation of this Gaussian set by the joint map values produces a state-conditioned representation, which can then be rendered at any target viewpoint.

4 ArtSplat: The Proposed Method

ArtSplat is a transformer-based feed-forward model that extends VGGT (Wang et al., 2025a) from static scenes to articulated objects by introducing a per-pixel joint map representation (Sec. 4.1). As shown in Fig. 1, it takes sparse multi-view images of two articulation states and predicts the camera, depth, Gaussian, and joint parameters in a single forward pass. Each input image is patchified and encoded into a set of tokens by DINOv2 (Oquab et al., 2023), prepended with learnable camera and state tokens, and contextualized through L layers of alternating frame and global attentions. These outputs are decoded by task-specific heads and a joint prediction module (Sec. 4.2) into per-pixel predictions that together form a state-conditioned Gaussian set $\mathcal{G}$ (Sec. 4.3).

4.1 Joint Map Representation

To reconstruct an articulated object, a model needs to determine (i) which parts can move, (ii) along which axes, and (iii) their rotation angle or displacement. One naive approach is to first segment the

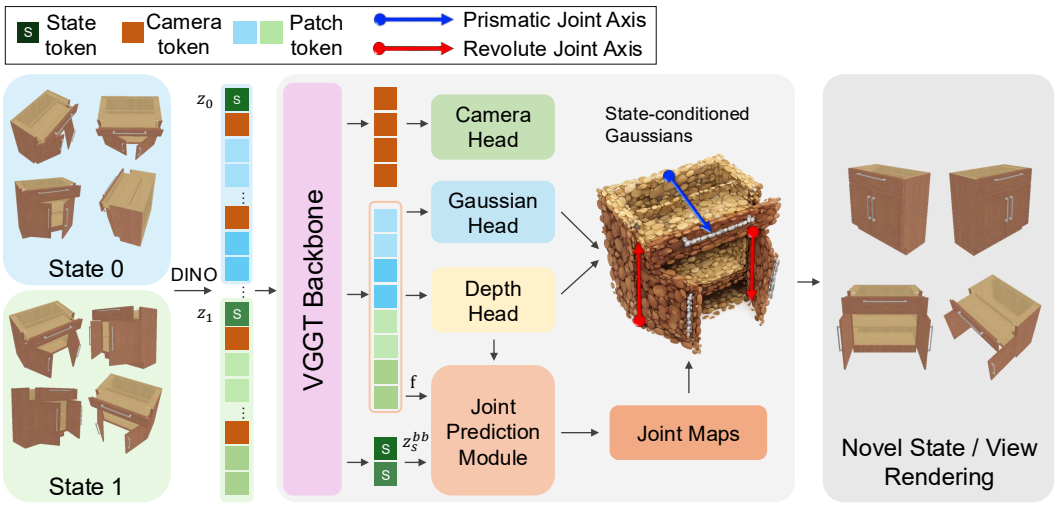


Figure 1: Overview. Given sparse multi-view images across two states, our model predicts geometry and joint parameters in a forward pass. Depth and Gaussian predictions are integrated with the joint maps to produce a state-conditioned Gaussian set, enabling articulated novel-state rendering without per-object optimization.

object into discrete parts and then regress one joint per part. However, the segmentation step relies on non-differentiable operations, breaking end-to-end training and making it difficult to integrate the pipeline into a feed-forward network. Another naive approach is to reserve a fixed number of output slots, where each slot predicts the joint parameters of one part. However, the number of parts varies across objects, and the assignment between the predicted slots and ground-truth parts is often unstable during training, leading to inconsistent learning signals.

We instead represent all joint parameters as a dense, per-pixel joint map $\mathbf{J}$, where each pixel is represented as an 11-dimensional vector defined in Sec. 3. These per-pixel joint parameters directly apply to its Gaussian primitive, constructed on the same pixel. By formulating articulation as a per-pixel regression, the joint head becomes fully differentiable and can be trained end-to-end with the depth and Gaussian heads, integrating articulation into our feed-forward reconstruction pipeline.

The 11 channels of the joint map $\mathbf{J}$ can be categorized into two groups. One is the *invariant* group $\mathbf{J}^{inv} \in \mathbb{R}^9$ containing properties intrinsic to the joint, *e.g.*, the joint type logits, the axis direction, and the pivot location. They remain consistent for any pixel belonging to the same part across two states. The other is the *variant* group $\mathbf{J}^{var} \in \mathbb{R}^2$, containing the rotation angle θ and translation displacement d, which describe how far the part has moved at the current state. They differ across states. See Tab. I in App. B for more details.

4.2 Joint Map Prediction Module

To predict a joint map, the network needs to identify what change arises between the two states; that is, which pixels move along an axis and by how much. We model this with three components: *state tokens* that capture per-state information, a *cross-state attention block* that compares the two states, and *invariant/variant decoders that turn* the results into a per-pixel joint map.

State tokens. We add a learnable state token per state, denoted by $\mathbf{z}_0, \mathbf{z}_1 \in \mathbb{R}^D$, inspired by the camera-token design of VGGT (Wang et al., 2025a). The state token $\mathbf{z}_s$ is prepended to the V views of state s. As the VGGT backbone's alternating frame- and global-attention layers process the full sequence, $\mathbf{z}_s$ attends to all V views of state s. We denote the resulting backbone output as $\mathbf{z}_s^{\text{bb}} \in \mathbb{R}^D$.

Cross-state attention. To strengthen the comparison between the two states, we introduce a single Cross-State Attention (CSA) block in which $\mathbf{z}_s^{\text{bb}}$ attends to the other state's image features. Let $F^{(s)} \in \mathbb{R}^{V \times N_p \times D}$ denote the patch tokens at the backbone's last block, collected across all V images of state s (N_p patches per image). Using $\mathbf{z}_s^{\text{bb}}$ as the query and $F^{(1-s)}$ (the patch tokens of the other state) as both keys and values, CSA produces a refined summary $\tilde{\mathbf{z}}_s$ for each state:

$$\tilde{\mathbf{z}}_s = \text{CSA}\big(Q = \mathbf{z}_s^{\text{bb}},\ K = F^{(1-s)},\ V = F^{(1-s)}\big), \qquad \text{where } s \in \{0, 1\}. \tag{1}$$

Invariant/variant decoders. The joint head uses a DPT-style (Ranftl et al., 2021) decoder with two branches that share the fusion stages: a 9-channel invariant branch for the joint-wise invariant properties (*e.g.*, type, axis, pivot) and the 2-channel variant branch for the state-dependent θ and d. The conditioning vectors are derived from the CSA outputs to match the symmetry of each branch:

$$\mathbf{c}^{inv} = (\tilde{\mathbf{z}}_0 + \tilde{\mathbf{z}}_1)/2, \qquad \mathbf{c}_s^{var} = \tilde{\mathbf{z}}_s, \quad \text{where } s \in \{0, 1\}.$$

Here, the invariant vector $\mathbf{c}^{inv}$ is obtained by symmetrically pooling the two CSA outputs, making it suitable for predicting quantities shared across both states. The variant vector $\mathbf{c}_s^{var}$ keeps the state-specific component for predictions that should differ between states.

Let $\mathbf{f} \in \mathbb{R}^{C \times h \times w}$ denote the per-pixel feature map at a given fusion stage of the decoder. We first inject geometry by adding the output of a pointwise MLP, which projects the $\mathbf{P}_{i,s}$ into the feature dimension, then apply FiLM (Perez et al., 2018):

$$\mathbf{f}' = \mathbf{f} + \mathrm{MLP}(\mathbf{P}_{i,s}), \qquad \mathbf{f}'' = \boldsymbol{\gamma} \odot \mathbf{f}' + \boldsymbol{\beta}, \tag{2}$$

where $\boldsymbol{\gamma}, \boldsymbol{\beta}$ are linear projections of the branch's conditioning vector ($\mathbf{c}^{inv}$, $\mathbf{c}_s^{var}$). The modulated feature $\mathbf{f}''$ is then decoded by each branch into 9-channel invariant and 2-channel variant predictions, which are concatenated into the 11-channel joint map $\mathbf{J}_{i,s}$.

4.3 Articulation transform

To move Gaussians of the same part together, a part needs to be assigned to each pixel. At training, we supervise the model to group Gaussians by part for the articulation transform, detailed below, using the true part label, available in the training data. The joint map is learned end-to-end, without any clustering. At inference, we assign part pseudo-labels by clustering Gaussians within each joint type using HDBSCAN (Campello et al., 2013), a density-based algorithm that infers the number of clusters from the data. For revolute joints, we cluster on the Plücker line coordinates $(\mathbf{a}, \mathbf{a} \times \mathbf{p})$, which uniquely identify the rotation axis as a 3D line. For prismatic joints, we cluster on the axis direction $\mathbf{a}$ and displacement d. This yields a per-Gaussian pseudo-label for each part p_n.

Then, for each part p, we obtain a part-level axis $\mathbf{a}_p$, pivot $\mathbf{p}_p$, and reference angle $\bar{\theta}_p$ (revolute) or displacement $\bar{d}_p$ (prismatic) by averaging the per-Gaussian predictions over all Gaussians belonging to the part p. Given target articulation angles θ_p^* or displacements d_p^*, we compute the part-level relative motion $\Delta\theta_p = \theta_p^* - \bar{\theta}_p$ or $\Delta d_p = d_p^* - \bar{d}_p$, and apply it rigidly to every Gaussian for the part:

$$(\boldsymbol{\mu}_n', \mathbf{q}_n') = \begin{cases} \Big(R(\mathbf{a}_{p_n}, \Delta\theta_{p_n})(\boldsymbol{\mu}_n - \mathbf{p}_{p_n}) + \mathbf{p}_{p_n}, \;\; q(\mathbf{a}_{p_n}, \Delta\theta_{p_n}) \otimes \mathbf{q}_n\Big) & \text{revolute,} \\ \big(\boldsymbol{\mu}_n + \Delta d_{p_n}\, \mathbf{a}_{p_n}, \;\; \mathbf{q}_n\big) & \text{prismatic,} \end{cases} \tag{3}$$

where rotation matrix $R(\mathbf{a}, \Delta\theta)$ is defined by Rodrigues' formula (Hartley & Zisserman, 2003), and the corresponding unit quaternion $q(\mathbf{a}, \Delta\theta)$ represents the same rotation of angle $\Delta\theta$ about axis $\mathbf{a}$:

$$R(\mathbf{a}, \Delta\theta) = I + \sin(\Delta\theta)\,[\mathbf{a}]_\times + (1 - \cos(\Delta\theta))\,[\mathbf{a}]_\times^2, \tag{4}$$

$$q(\mathbf{a}, \Delta\theta) = \big(\cos(\Delta\theta/2),\; \sin(\Delta\theta/2)\,\mathbf{a}\big), \tag{5}$$

where $[\mathbf{a}]_\times$ is the skew-symmetric cross-product matrix of $\mathbf{a}$, and $\otimes$ denotes quaternion multiplication. Applying this transform to every Gaussian yields a state-conditioned Gaussian set $\mathcal{G}^{(s^*)}$ that places each part at its target articulation and can be rendered from any viewpoint.

4.4 Training

Inaccurate joint maps can transform Gaussians incorrectly and produce misleading photometric gradients when all losses are optimized jointly. We therefore use two-stage training: Stage 1 directly supervises geometry and joint parameters, and Stage 2 adds rendering supervision to refine appearance and the joint map through the differentiable articulation transform in Eq. (3).

In *Stage 1*, we minimize the following loss:

$$\mathcal{L}_{stage1} = \lambda_{\mathrm{p}}\mathcal{L}_{pose} + \lambda_{\mathrm{d}}\mathcal{L}_{depth} + \lambda_{\mathrm{j}}\mathcal{L}_{joint} + \lambda_{\mathrm{c}}\mathcal{L}_{consist} + \lambda_{\mathrm{s}}\mathcal{L}_{smooth}. \tag{6}$$

$\mathcal{L}_{pose}$ and $\mathcal{L}_{depth}$ supervise the camera and depth heads following VGGT (Wang et al., 2025a). The remaining three terms supervise the joint map. $\mathcal{L}_{joint}$ aggregates per-channel supervision on pixels by

$$\mathcal{L}_{joint} = \mathrm{CE}(\boldsymbol{\tau}, \boldsymbol{\tau}^*) + \|\mathbf{a} - \mathbf{a}^*\|_1 + d_\perp(\mathbf{p}, \mathbf{p}^*; \mathbf{a}^*) + \mathcal{H}(\theta, \theta^*) + \mathcal{H}(d, d^*), \tag{7}$$

where $\boldsymbol{\tau}, \mathbf{a}, \mathbf{p}, \theta, d$ are the predicted joint type, axis, pivot, angle, and displacement, starred quantities denote ground truth, and CE is the cross-entropy loss. Note that $d_\perp$ is the perpendicular distance from the predicted pivot $\mathbf{p}$ to the GT axis line passing through $\mathbf{p}^*$ in direction $\mathbf{a}^*$:

$$d_\perp(\mathbf{p}, \mathbf{p}^*; \mathbf{a}^*) = \left\| (\mathbf{p} - \mathbf{p}^*) - \big((\mathbf{p} - \mathbf{p}^*) \cdot \mathbf{a}^*\big)\mathbf{a}^* \right\|_2, \tag{8}$$

which is applied to penalize only deviations from the ground-truth axis line while ignoring offsets along the axis direction. $\mathcal{H}$ denotes the Huber loss (Huber, 1964), applied to θ and d only for revolute and prismatic pixels, respectively. Construction of the true joint map is detailed in App. B. $\mathcal{L}_{consist}$ enforces the invariant prediction to be same across the two states, averaged per part. Denoting the set of moving parts by $\mathcal{P}$ and the set of pixels belonging to part p in state s by $\Omega_p^{(s)}$, we define the consistency loss as

$$\mathcal{L}_{consist} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \left\| \mu_p^{(0)} - \mu_p^{(1)} \right\|_2, \quad \text{where} \quad \mu_p^{(s)} = \frac{1}{|\Omega_p^{(s)}|} \sum_{i \in \Omega_p^{(s)}} \mathbf{J}_i^{inv}. \tag{9}$$

$\mathcal{L}_{smooth}$ is a standard part-aware total variation (Rudin et al., 1992) that penalizes ℓ_1 differences only between adjacent pixels of the same part, preserving part boundaries.

In *Stage 2*, we add a rendering loss $\mathcal{L}_{rgb}$, weighted by λ_{rgb}, to $\mathcal{L}_{stage1}$. The rendering loss combines a per-pixel MSE term and a LPIPS perceptual term between rendered and ground-truth images.

5 Experiments

5.1 Experimental settings

Dataset. We train on PartNet-Mobility (Chang et al., 2015; Mo et al., 2019; Xiang et al., 2020), a large-scale dataset of URDF-annotated articulated objects across diverse categories. More details on this dataset are provided in App. A. For evaluation, we hold out 18 single-joint and 50 multi-joint objects from the training set. We additionally evaluate cross-dataset generalization on 62 objects from the AKB-48 subset of GAPartNet (Liu et al., 2022; Geng et al., 2023), covering two categories: Box (40 objects) and Drawer (22 objects).

Evaluation protocol. All methods receive the same 8 input views (2 states $\times$ 4 views) and are evaluated on 12 held-out target views per state. We report results separately on the single-joint and multi-joint splits. Details on view sampling are provided in App. C.

Baselines. We compare with per-object optimization methods built on implicit representations (Liu et al., 2023; Weng et al., 2024) and 3DGS (Liu et al., 2025; Kim et al., 2025; Wu et al., 2025), covering a range of input priors from RGB-only to RGB-D with the number of joints supervision.

Evaluation metrics. Following ScrewSplat (Kim et al., 2025), we adopt three categories of evaluation metrics. For *geometry*, we report Chamfer Distance separately for the static part (CD-s), the movable parts (CD-m, averaged over all movable parts), and the entire object (CD-w). For *motion*, we measure the angular error (Ang$_m$, $^\circ$) between the estimated and ground-truth axes, and the axis position error (Pos$_m$) for revolute joints, computed as the minimum distance between corresponding joint axes. For *appearance*, we report Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) on rendered images from the held-out target views.

Architecture. We take the 1B-parameter VGGT aggregator initialized from AnySplat (Jiang et al., 2025) as our backbone. We adapt it with rank-16 LoRA (Hu et al., 2022) adapters on every attention projection, and add a learnable state token of dimension $D = 2048$. All other VGGT parameters are frozen. CSA is a single block with 16 attention heads and a feature dimension of 2048. We take DPT (Ranftl et al., 2021) for the decoder, where the head aggregates intermediate features from layers $\{4, 11, 17, 23\}$ of the backbone and progressively fuses them into 256-channel feature maps, from which it produces an 11-channel output split into the invariant (9) and variant (2) branches. The Gaussian head shares the same DPT layout and outputs the AnySplat Gaussian parameters together with a confidence channel.

Optimization. We train with AdamW ($\beta_1 = 0.9, \beta_2 = 0.95$, weight decay 0.05) using a learning rate of 1×10^{-4}, with a 1,000-step warm-up followed by cosine decay to 1×10^{-5}. LoRA adapters inside the backbone use $0.1\times$ the base learning rate. Loss weights are λ_{p}=3, λ_{d}=3, λ_{j}=5, λ_{c}=0.5,

Table 1: Evaluation on PartNet-Mobility. Best results are in **bold**, and the second best are underlined.

	Method	Prior	Geometry (↓)			Motion (↓)		Appearance (↑)	
			CD-w	CD-s	CD-m	Ang_m	Pos_m	PSNR	SSIM
Single-joint	PARIS (Liu et al., 2023)	RGB + number of joints	0.022	0.029	0.294	55.96	**0.166**	20.51	0.909
	DTA (Weng et al., 2024)	RGB-D + number of joints	**0.004**	**0.008**	0.145	33.79	2.305	20.23	0.822
	ArtGS (Liu et al., 2025)	RGB-D + number of joints	0.013	0.010	0.102	64.67	1.165	28.01	0.924
	REArtGS (Wu et al., 2025)	RGB + number of joints	0.045	0.129	0.443	47.87	1.993	23.44	0.865
	ScrewSplat (Kim et al., 2025)	RGB	0.036	0.039	0.301	26.27	2.401	22.93	0.918
	ArtSplat (Ours)	RGB	0.031	0.023	**0.028**	**6.65**	0.436	**28.70**	**0.941**
Multi-joint	PARIS (Liu et al., 2023)	RGB + number of joints	**0.017**	0.029	0.335	53.08	**0.221**	20.03	0.913
	DTA (Weng et al., 2024)	RGB-D + number of joints	0.025	0.053	0.527	44.48	22.790	19.56	0.813
	ArtGS (Liu et al., 2025)	RGB-D + number of joints	0.096	0.102	0.782	49.31	1.825	25.21	0.893
	ScrewSplat (Kim et al., 2025)	RGB	0.035	0.044	0.405	46.78	3.368	23.49	0.912
	ArtSplat (Ours)	RGB	0.034	**0.024**	**0.030**	**5.79**	0.864	**26.30**	**0.921**

λ_{s}=0.1, and λ_{rgb}=1, where $\mathcal{L}_{rgb} = \text{MSE} + 0.1 \cdot \text{LPIPS}$. Inputs are 448×448 images, giving 32×32 patches with the VGGT patch size of 14. Each batch contains $V \cdot S = 8$ images from a single object. Stage 1 and Stage 2 run for 80k and 40k iterations, respectively, using gradient clipping at 0.5 and bf16 mixed precision on 4 NVIDIA RTX A6000 GPUs. The full training schedule takes $\sim$4 days.

Inference. At inference, we run a single forward pass on the 8 input images and form the canonical set $\mathcal{G}$ by merging the per-pixel Gaussians with AnySplat's differentiable voxelization at voxel size 0.003. We then recover pseudo-labels for the parts by HDBSCAN on Plücker line coordinates (Sec. 4.3) and apply the articulation transform of Eq. (3) to obtain the state-conditioned set $\mathcal{G}^{(s^*)}$, which is rendered with the standard 3DGS rasterizer (Kerbl et al., 2023).

5.2 Main Results

Tab. 1 compares our method with baselines on both single-joint and multi-joint subsets of PartNet-Mobility. Only ArtSplat and ScrewSplat operate from RGB alone, while the rest additionally require depth maps, ground-truth joint counts, or both.

Geometry. Among RGB-only methods, ArtSplat achieves the best Chamfer Distance on all three metrics and both splits, consistently outperforming ScrewSplat. In particular, ArtSplat attains the lowest CD-m overall, even against baselines leveraging depth or joint-count priors. Since CD-m measures how well movable parts are reconstructed, it directly reflects whether the model correctly identifies which regions move and how they transform. Notably, ArtSplat's geometry remains robust with increased number of joints, which we attribute to cross-state attention explicitly comparing the two articulation states.

Motion. Accurate axis estimation is critical, as even small angular errors compound into large displacements for parts far from the joint origin. Existing methods struggle to recover accurate joint axis orientation, with most reporting angular errors above 30° regardless of the input prior. In contrast, ArtSplat predicts joint axes with substantially lower angular error on both single- and multi-joint splits, demonstrating robust axis estimation that does not degrade for objects with more joints.

Appearance. ArtSplat achieves competitive appearance quality on both splits, which is noteworthy given the fundamental efficiency gap: per-object optimization methods fit each scene for hundreds of iterations over sampled views, whereas ArtSplat produces all Gaussians in a single feed-forward pass from only 8 input images.

Qualitative comparison. As shown in Fig. 2, baselines often mispredict joint axes, causing movable parts to be rendered at incorrect positions or orientations and producing visible artifacts. In contrast, ArtSplat predicts joint axes accurately, yielding clean renderings of movable parts. Fig. 3 shows that ArtSplat also extracts competitive meshes while localizing joint axes more accurately than the baselines. As further shown in Fig. 4, ArtSplat can render the object at arbitrary articulation states $s \in [0, 1]$, smoothly interpolating between fully closed ($s = 0$) and fully open ($s = 1$) configurations.

Cross-dataset Generalization. We further compare with LARM (Yuan et al., 2025), another feed-forward articulated reconstruction method. For this, we compare the performance of ArtSplat trained on PartNet-Mobility against the publicly available LARM checkpoint on the AKB-48 subset of

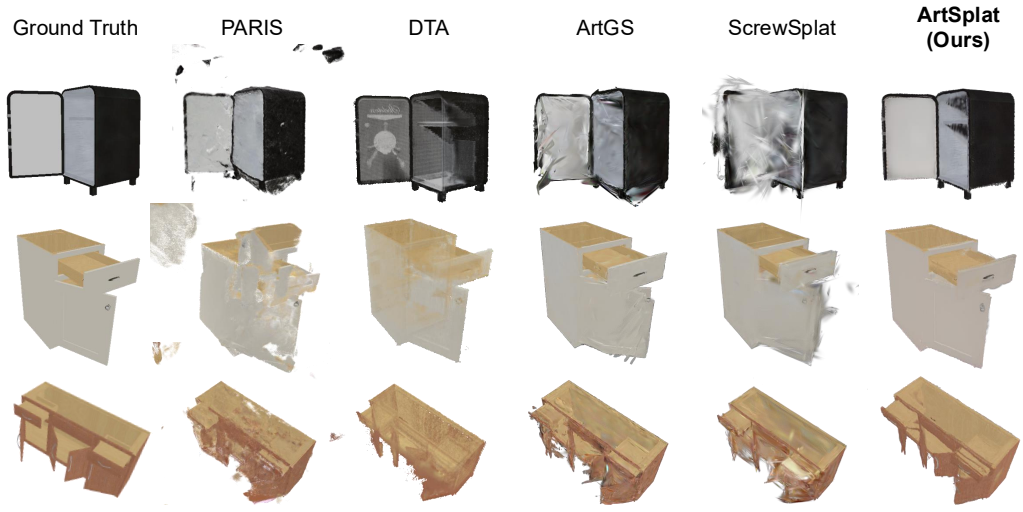


Figure 2: Qualitative comparison of novel-view renderings via Gaussian rasterization. Baselines exhibit ghosting and misaligned edges around the joints due to inaccurate axis estimation, whereas ArtSplat produces clean renderings of both static and movable parts.

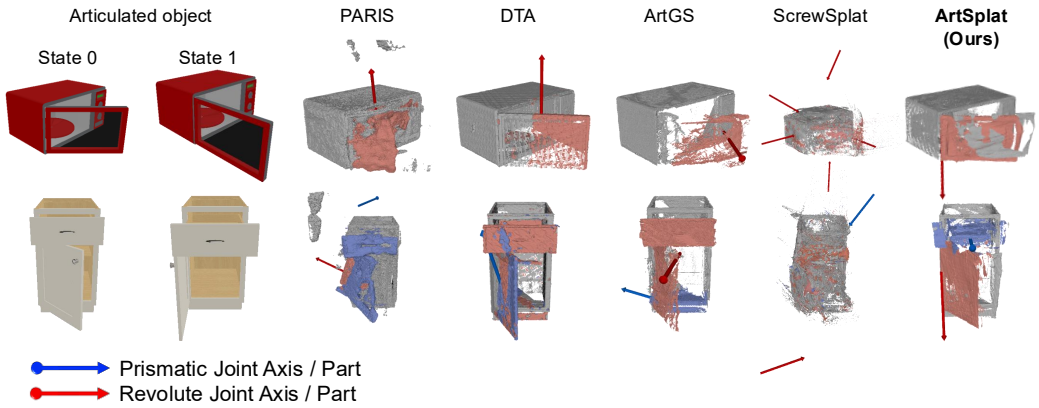


Figure 3: Qualitative comparison of extracted meshes and predicted joint axes.

GAPartNet.[1] As shown in Tab. 2, ArtSplat estimates joint-axis more accurately than both LARM and the optimization-based baselines, while matching the rendering quality of ArtGS and surpassing LARM in both metrics. These results support generalizability of ArtSplat beyond its training dataset.

Inference speed. Tab. 3 compares the inference speed of competing methods. As a feed-forward model, ArtSplat reconstructs each object in under 2 seconds, achieving over 389× speedup over existing baselines and approximately 700× over ScrewSplat, which also uses only RGB inputs.

5.3 Ablation Studies

Cross-state attention. We ablate the cross-state attention (CSA) by disabling it while retaining the state token, so the state token cannot attend to features from the other state but still provides FiLM conditioning to the joint head. As in Tab. 4, this substantially degrades axis and angle estimation accuracy while leaving per-state geometry largely intact. This asymmetric degradation indicates that axis and angle estimation relies mostly on the state token aggregating features across states.

State token conditioning. We further remove the state token in addition to CSA, so image tokens are passed directly to the DPT joint head without any state-specific signal. Tab. 4 reports that this variant performs worst across all metrics, with a particularly large drop relative to the CSA-only ablation. The gap indicates that the state token and CSA contribute beyond what either component alone does.

[1]We use this external subset because LARM's PartNet-Mobility training-object split could not be verified, making overlap with our PartNet-Mobility test set difficult to rule out.

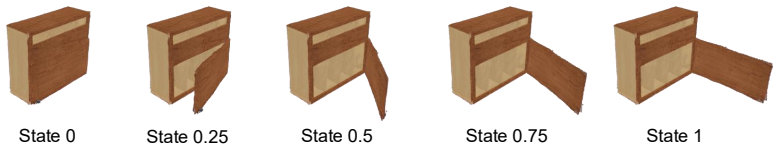


Figure 4: Illustration of novel state rendering, between fully closed ($s = 0$) to fully open ($s = 1$).

Table 2: Evaluation on AKB-48. We compare the PartNet-Mobility-trained ArtSplat model with per-object optimization methods and the feed-forward baseline LARM. Ang_m measures joint-axis direction error.

Method	Inference	$\mathrm{Ang}_m \downarrow$	PSNR↑	SSIM↑
ArtGS	Optimization	72.21	<u>28.61</u>	0.9179
ScrewSplat	Optimization	<u>41.60</u>	21.60	0.8920
LARM	Feed-forward	65.30	22.70	<u>0.9240</u>
ArtSplat (Ours)	Feed-forward	**22.00**	**28.70**	**0.9384**

Table 3: Per-object inference speed on PartNet-Mobility. We compare average wall-clock time to reconstruct a single articulated object, measured on a single NVIDIA RTX A6000 GPU with 8 input views.

Method	PARIS	DTA	ArtGS	REArtGS	ScrewSplat	**ArtSplat (Ours)**
Inference time ↓	<u>12m 57s</u>	56m 07s	15m 50s	1h 37m	23m 05s	**<2s**

Table 4: Architecture and training-schedule ablations. Each row removes or replaces one component and is retrained with the same schedule as the full model.

	Geometry (↓)			Motion (↓)		Appearance (↑)	
Method	CD-w	CD-s	CD-m	Ang_m	Pos_m	PSNR	SSIM
ArtSplat (full)	**0.034**	**0.024**	**0.030**	**5.79**	**0.864**	**26.30**	**0.921**
w/o Cross-state Attention (CSA)	0.067	0.029	0.082	18.74	1.852	24.18	0.901
w/o CSA and state token conditioning	0.112	0.041	0.156	31.46	2.917	22.51	0.879
w/o Inv/Var split (single branch)	<u>0.042</u>	<u>0.026</u>	<u>0.041</u>	<u>7.12</u>	1.643	<u>25.07</u>	<u>0.913</u>
Single-stage joint training	0.058	0.030	0.069	12.85	<u>1.428</u>	23.74	0.895

Invariant/variant DPT. We ablate the invariant/variant split by replacing the dual-branch DPT head with a single shared head that jointly regresses all eleven channels, without FiLM conditioning. As seen in Tab. 4, the single-head variant achieves comparable axis estimation but consistently underperforms on angle and displacement. This result shows that structurally distinguishing them yields more accurate variant predictions without sacrificing invariant-attribute quality.

Two-stage training. We compare our two-stage training with single-stage training that jointly optimizes all losses, including the rendering loss. Both use the same total number of training iterations. The last row of Tab. 4 shows substantially worse performance. We attribute this to misleading photometric gradients through transforms based on noisy joint predictions, which destabilize the joint and point heads and cause oscillating joint losses. Our two-stage training first stabilizes geometry and joint predictions, then introduces photometric supervision to refine the reconstruction. Stage 1 also reduces per-iteration cost by omitting differentiable rasterization and photometric supervision.

6 Conclusion

We propose ArtSplat, a feed-forward framework that reconstructs articulated geometry and joint parameters from sparse, uncalibrated RGB views. Its per-pixel joint map and joint prediction module capture inter-state differences, enabling single-pass reconstruction without per-object optimization. Experiments on PartNet-Mobility demonstrate effective geometry and motion reconstruction at substantially faster speeds than optimization-based methods.

Limitations and future work. Our model currently handles only single-degree-of-freedom joints connected directly to a static base. Extending it to kinematic chains, where multiple joints are connected in series as in a robotic arm, and validating on real-world captures would be promising directions for future work.

AI USE STATEMENT

In this work, we used generative AI tools for assistance with translation, research methodology and experimental design, and implementation of the proposed method. We have not used generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, cleaning or reformatting datasets, conducting qualitative or thematic data analysis, or interpreting experimental results, and tasks involving providing critical ingredients for mathematical proofs or writing proofs are not applicable to this work. Additionally, we used generative AI tools for improving manuscript readability, searching for information, brainstorming, preparing tables and documents, and writing or editing auxiliary code.

We have reviewed all AI-assisted work. We checked revised and translated text against the original to ensure that the intended technical meaning was preserved. The authors reviewed suggestions on research methodology and experiments and determined whether to adopt them. We executed or tested AI-assisted code, checked retrieved information against original sources, and verified tables and documents against the underlying data and experimental results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-Based Clustering Based on Hierarchical Density Estimates. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2013.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv:1512.03012*, 2015.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian Splats from Image Pairs for Scalable Generalizable 3D Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. Easi3R: Estimating Disentangled Motion from DUSt3R Without Training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: Efficient 3D Gaussian Splatting from Sparse Multi-View Images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

Haoran Geng, Helin Xu, Chengyang Zhao, Chao Xu, Li Yi, Siyuan Huang, and He Wang. GAPartNet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Junfu Guo, Yu Xin, Gaoyi Liu, Kai Xu, Ligang Liu, and Ruizhen Hu. ArticulatedGS: Self-supervised Digital Twin Modeling of Articulated Objects using 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. LoRA: Low-Rank Adaptation of Large Language Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *ACM SIGGRAPH 2024 conference papers*, pp. 1–11, 2024.

Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. AnySplat: Feed-forward 3D Gaussian Splatting from Unconstrained Views. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 2025.

Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building Digital Twins of Articulated Objects from Interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Seungyeon Kim, Junsu Ha, Young Hun Kim, Yonghyeon Lee, and Frank C Park. ScrewSplat: An End-to-End Method for Articulated Object Recognition. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2025.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding Image Matching in 3D with MASt3R. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

Zizhang Li, Cheng Zhang, Zhengqin Li, Henry Howard-Jenkins, Zhaoyang Lv, Chen Geng, Jiajun Wu, Richard Newcombe, Jakob Engel, and Zhao Dong. ART: Articulated Reconstruction Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Shengjie Lin, Jiading Fang, Muhammad Zubair Irshad, Vitor Campagnolo Guizilini, Rares Andrei Ambrus, Greg Shakhnarovich, and Matthew R Walter. SplArt: Articulation Estimation and Part-Level Reconstruction with 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. PARIS: Part-level Reconstruction and Motion Analysis for Articulated Objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Liu Liu, Wenqiang Xu, Haoyuan Fu, Sucheng Qian, Qiaojun Yu, Yang Han, and Cewu Lu. AKB-48: A real-world articulated object knowledge base. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Yu Liu, Baoxiong Jia, Ruijie Lu, Junfeng Ni, Song-Chun Zhu, and Siyuan Huang. ArtGS: Building Interactable Replicas of Complex Articulated Objects via Gaussian Splatting. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Communications of the ACM*, 65 (1):99–106, 2021.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning Robust Visual Features without Supervision. *arXiv:2304.07193*, 2023.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision Transformers for Dense Prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

Licheng Shen, Saining Zhang, Honghan Li, Peilin Yang, Zihao Huang, Zongzheng Zhang, and Hao Zhao. GaussianArt: Unified Modeling of Geometry and Motion for Articulated Objects. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2025.

Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3R: Zero-shot Gaussian Splatting from Uncalibrated Image Pairs. *arXiv:2408.13912*, 2024.

Wei-Cheng Tseng, Hung-Ju Liao, Lin Yen-Chen, and Min Sun. CLA-NeRF: Category-Level Articulated Neural Radiance Field. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual Geometry Grounded Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3D Perception Model with Persistent State. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025b.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D Vision Made Easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Yijia Weng, Bowen Wen, Jonathan Tremblay, Valts Blukis, Dieter Fox, Leonidas Guibas, and Stan Birchfield. Neural Implicit Representation for Building Digital Twins of Unknown Articulated Objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Di Wu, Liu Liu, Zhou Linli, Anran Huang, Liangtu Song, Qiaojun Yu, Qi Wu, and Cewu Lu. REArtGS: Reconstructing and Generating Articulated Objects via 3D Gaussian Splatting with Geometric and Motion Constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. DepthSplat: Connecting Gaussian Splatting and Depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3R: Towards 3D Reconstruction of 1000+ Images in One Forward Pass. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No Pose, No Problem: Surprisingly Simple 3D Gaussian Splats from Sparse Unposed Images. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Tianjiao Yu, Vedant Shah, Muntasir Wahed, Ying Shen, Kiet A Nguyen, and Ismini Lourentzou. Part^{2}GS: Part-aware Modeling of Articulated Objects using 3D Gaussian Splatting. *arXiv:2506.17212*, 2025.

Sylvia Yuan, Ruoxi Shi, Xinyue Wei, Xiaoshuai Zhang, Hao Su, and Minghua Liu. LARM: A Large Articulated Object Reconstruction Model. In *Proceedings of the SIGGRAPH Asia Conference Papers (SIGGRAPH Asia)*, 2025.

Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. MonST3R: A Simple Approach for Estimating Geometry in the Presence of Motion. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025a.

Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. FLARE: Feed-forward Geometry, Appearance and Camera Estimation from Uncalibrated Sparse Views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025b.

APPENDIX

A TRAINING DATA DETAILS

From PartNet-Mobility, we retain objects with valid revolute or prismatic joints, resulting in 972 objects across 15 articulated object categories. We divide these objects into 832 training, 72 validation, and 68 test objects. The test split covers all 15 categories and contains 18 single-joint and 50 multi-joint objects.

A.1 MULTI-VIEW RENDERING

For each training object, we pre-render RGB images, depth maps, and per-part segmentation masks from 48 camera viewpoints across 16 articulation states, yielding $48 \times 16 = 768$ frames per object.

Camera placement. Cameras are distributed on the upper hemisphere via stratified jittering, looking at the object center with no roll. The hemisphere is covered by two layers: a *main layer* of 5 elevation bins $\times$ 8 azimuth bins ($= 40$ views) spanning elevation $[0°, 72°]$ and full $360°$ azimuth, and a *top layer* of 2 elevation bins $\times$ 4 azimuth bins ($= 8$ views) spanning elevation $[72°, 85°]$.

Articulation state sampling. We sample 16 states per object using stratified jittering over the normalized articulation range $[0, 1]$. The range is divided into 16 equal bins, and one articulation value is randomly sampled within each bin. For multi-joint objects, each joint receives an independent random permutation of the 16 bins, so joint configurations are decorrelated across states.

A.2 TRAINING VIEW SAMPLING

At each training iteration, we sample one object and draw 2 states uniformly from the 16 pre-rendered states, then select 4 views per state from the 48 available viewpoints, forming a batch of $V \cdot S = 8$ images. All supervision signals—ground-truth depth, part segmentation, camera extrinsics, and the joint map.

B GROUND-TRUTH JOINT MAP DETAILS

The ground-truth joint map $\mathbf{J}^*_{i,s} \in \mathbb{R}^{H \times W \times 11}$ is constructed per frame from per-pixel part segmentation labels, joint annotations (type, axis, pivot), and joint positions (angle or displacement) from the PartNet-Mobility dataset. The channel layout is summarized in Tab. I. Following VGGT (Wang et al., 2025a), all geometric quantities are expressed in a *canonical frame* defined by the first camera $[R_0 \mid \mathbf{t}_0]$, and further divided by a radius normalization factor $\bar{r}$ (the mean distance of foreground 3D points from the canonical origin) so that the scale-normalized space is shared across the point map, camera translations, and joint map.

For each pixel with part label ℓ (-1 = background, 0 = static base, ≥ 1 = moving part), the 11 channels are populated as follows:

- **Channels 0–2** (joint type): One-hot encoding of the joint type: static $[1, 0, 0]$, revolute $[0, 1, 0]$, or prismatic $[0, 0, 1]$.
- **Channels 3–5** (axis direction): The joint axis direction associated with the corresponding articulated part, transformed into the canonical frame and normalized to a unit vector.
- **Channels 6–8** (pivot position): The joint pivot position associated with the articulated part, transformed into the canonical frame and normalized by $\bar{r}$.
- **Channel 9** (rotation angle θ): For revolute joints, the articulation value is converted into a rotation angle using the joint motion range.
- **Channel 10** (displacement d): For prismatic joints, the articulation value is converted into a linear displacement using the joint motion range and normalized by $\bar{r}$.

All pixels belonging to the same moving part share identical values for channels 0–8 (invariant properties), while channels 9–10 (variant properties) vary according to the articulation state of the current frame.

Table I: Channel layout of the per-pixel joint map **J** (11 channels). Invariant channels describe the joint identity and must agree across states; variant channels describe state-dependent motion.

Channel	Quantity	Type	Invariant
0–2	joint type logits	static / revolute / prismatic	✓
3–5	axis direction $\mathbf{a}$	unit 3-vector	✓
6–8	pivot $\mathbf{p}$	3-D world point	✓
9	rotation θ	radian (revolute)	per-state
10	displacement d	meter (prismatic)	per-state

C Evaluation setup

C.1 Evaluation view sampling

All cameras are placed on the upper hemisphere looking at the object center. Views are generated via *stratified jittering*: we partition the elevation range into N_e equal intervals and the azimuth range into N_a equal intervals, forming an $N_e \times N_a$ grid of bins. Within each bin, one camera is placed at a uniformly random elevation and azimuth, yielding $N_e \cdot N_a$ views in total.

Input views. Each method receives 8 images (2 states $\times$ 4 views). The 4 views per state use $N_e{=}1, N_a{=}4$ over elevation $[5°, 60°]$ and azimuth $[-180°, 180°)$.

Target views. Each state is evaluated on 12 held-out views, using $N_e{=}3, N_a{=}4$ over elevation $[0°, 90°]$ and azimuth $[-180°, 180°)$. The wider elevation range ensures that target views include viewpoints above and below the input distribution, testing generalization rather than interpolation alone.

D Additional quantitative Analyses

D.1 Sensitivity to HDBSCAN Configurations

To assess robustness to the HDBSCAN configuration used for inference-time part assignment, we compare four paired settings of `min_cluster_size` and `min_samples`, including the default $(50, 25)$, using the same checkpoint and network predictions. As shown in Table II, performance is stable across a practical range of configurations, with only the most conservative setting $(300, 100)$ showing a noticeable degradation. We additionally evaluate a reference that replaces the inferred part assignments with ground-truth part labels while keeping all other predictions fixed. This reference yields only marginal gains over the default configuration, indicating that our clustering-based part assignment closely approaches the ground-truth grouping.

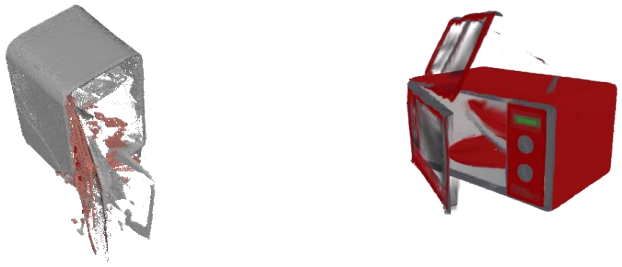
Table II: Sensitivity to paired HDBSCAN configurations. The default setting is highlighted. The GT-label reference uses ground-truth part assignments and is not an inference-time configuration. PSNR is reported in dB.

`min_cluster_size`	`min_samples`	$\text{Ang}_m \downarrow$	PSNR↑
10	5	6.25	26.24
50 (default)	**25**	**5.79**	**26.30**
150	50	6.42	26.15
300	100	8.85	25.40
GT part labels	—	5.21	26.85

E Additional qualitative results

Fig. II provides additional qualitative comparisons on three single-joint and three multi-joint objects from the test set. Notably, the bottom example contains 8 revolute and 6 prismatic joints, and ArtSplat still achieves favorable rendering quality compared to the baselines, highlighting its scalability to complex articulation structures.

Failure cases. Fig. I illustrates two representative failure cases of ArtSplat. First, 3D Gaussians are inherently unstructured primitives without explicit surface connectivity, which makes high-fidelity mesh reconstruction challenging. Adopting surface-aligned representations such as 2D Gaussian Splatting Huang et al. (2024) is a promising direction to mitigate this limitation. Second, when the articulation change between the two observed states is small, the predicted joint axis can deviate significantly from the ground truth, leading to incorrect articulation transforms and consequently degraded novel-view rendering. Leveraging external priors, *e.g.*, vision-language models that explicitly reason about inter-state differences, could help resolve such ambiguities.



(a) Failure Case: Mesh Reconstruction (b) Failure Case: Joint Axis Prediction

Figure I: Representative failure cases. *Left:* Mesh reconstruction artifacts. *Right:* Rendering errors caused by inaccurate joint axis prediction when the articulation difference between two states is small.

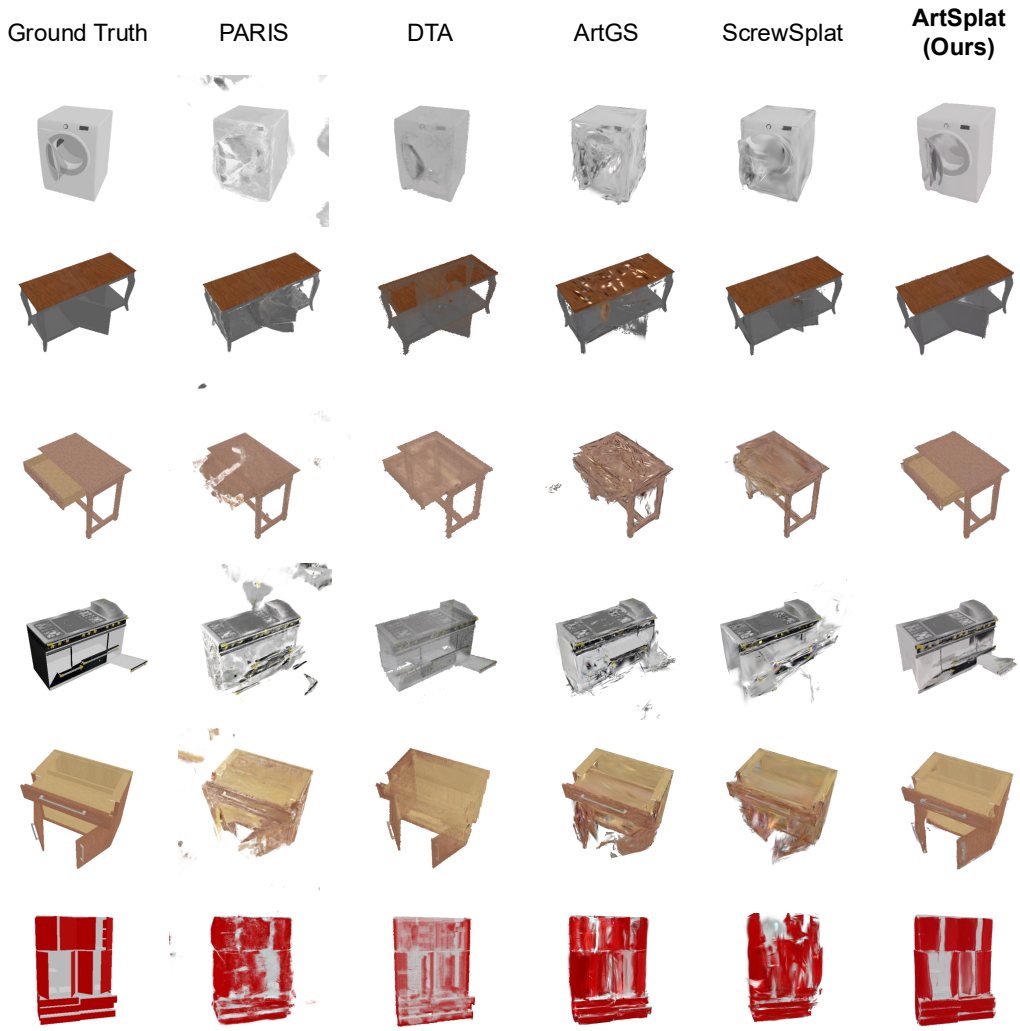


Figure II: Additional qualitative comparisons on novel-view rendering across diverse articulated objects from the test set.