ScrewSplat: An End-to-End Method for Articulated Object Recognition

Seungyeon Kim[1] **Junsu Ha**[1] **Young Hun Kim**[1] **Yonghyeon Lee**[2] **Frank Chongwoo Park**[1]
[1]Seoul National University, [2]Massachusetts Institute of Technology
{ksy, hajunsu, yhun}@robotics.snu.ac.kr, yhl@mit.edu, fcp@snu.ac.kr

Abstract: Articulated object recognition – the task of identifying both the geometry and kinematic joints of objects with movable parts – is essential for enabling robots to interact with everyday objects such as doors and laptops. However, existing approaches often rely on strong assumptions, such as a known number of articulated parts; require additional inputs, such as depth images; or involve complex intermediate steps that can introduce potential errors – limiting their practicality in real-world settings. In this paper, we introduce *ScrewSplat*, a simple end-to-end method that operates solely on RGB observations. Our approach begins by randomly initializing screw axes, which are then iteratively optimized to recover the object's underlying kinematic structure. By integrating with Gaussian Splatting, we simultaneously reconstruct the 3D geometry and segment the object into rigid, movable parts. We demonstrate that our method achieves state-of-the-art recognition accuracy across a diverse set of articulated objects, and further enables zero-shot, text-guided manipulation using the recovered kinematic model. See the project website at: https://screwsplat.github.io.

Keywords: Articulated objects, Gaussian splatting, Screw theory

1 Introduction

Articulated objects with movable parts – such as doors, laptops, and drawers – are common in everyday environments, and manipulating them requires understanding both their 3D geometry and underlying kinematic structure (e.g., joint types and axes). While prior work has addressed this using large-scale datasets of 3D objects with annotated joint axes in supervised settings [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11], such methods struggle to generalize to unseen categories – a natural limitation of supervised learning. In this work, we tackle a more challenging yet practical scenario: inferring kinematic structure directly from multi-view RGB images under varying object configurations, without relying on category-specific supervision (see the left of Figure 1).

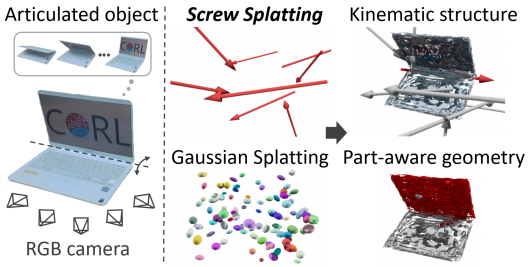

Figure 1: Articulated object recognition by splatting screw axes and Gaussians.

Spurred in part by the success of neural rendering-based 3D reconstruction methods that require no supervised training [12, 13, 14, 15], recent works have adapted these frameworks for articulated object recognition [16, 17, 18, 19, 20], achieving promising results using raw RGB observations. However, a key drawback of these methods lies in their reliance on strong assumptions, such as a known number of articulated components or predefined joint types. Moreover, they often involve multi-stage pipelines with intermediate procedures like point correspondence matching or part clustering, which not only increase overall complexity but also introduce potential points of failure –

errors in these stages can significantly impair final performance. Furthermore, some of these approaches rely on auxiliary depth inputs, which are often noisier than RGB images – particularly for transparent or reflective surfaces – thereby limiting their robustness in real-world scenarios.

We propose *ScrewSplat*, a simple end-to-end method for articulated object recognition that avoids intermediate steps, auxiliary data, and any prior knowledge of joint types or counts. We formulate the task as a joint optimization over part-aware geometry, joint axes and types, and joint angles, such that the rendered views match the observations. This problem is particularly challenging due to its hybrid structure – continuous variables (geometry and joint angles) are coupled with discrete, combinatorial ones (part segmentation labels, joint types, and joint counts).

Our key idea is to adopt Gaussian splatting to represent geometry and appearance [14], and extend it with a screw model that provides a continuous parameterization of joint axes [21]. To represent joint types and counts without relying on discrete variables, we introduce confidence scores over screw axes and use a probability simplex to softly assign Gaussians to rigid parts. This unified formulation enables smooth, end-to-end optimization over both geometric and kinematic components. As a result, our method achieves accurate recovery of 3D geometry, screw axes, and part decomposition of articulated objects (see the right of Figure 1).

Building on this framework, we develop a simple yet effective algorithm for articulated object manipulation. A core strength of our model is its rendering fidelity, which we leverage alongside a large vision-language model to infer target configurations – such as the joint angles corresponding to an "open laptop" or "folded chair." These inferred goals are then used to control a robotic manipulator. Experiments show that *ScrewSplat* consistently outperforms prior methods in recovering both geometric and kinematic structures across a wide range of single- and multi-joint objects. We also demonstrate its effectiveness in zero-shot, text-guided manipulation tasks, validating its practical utility in real-world robotic scenarios.

2 Related Works

This section focuses on articulated object recognition methods that *do not* rely on explicit supervision [16, 17, 18, 19, 20]. A detailed review of supervised methods is provided in Appendix A.1.

A representative early work is PARIS [16], which achieves part-level reconstruction and articulation discovery with NeRF, but is restricted to single-joint objects with known joint types. DTA [17], a more recent approach, extend to multi-joint objects by reconstructing meshes from RGB-D data and inferring kinematic structures via correspondence matching; however, they still depend on depth input and prior knowledge of the number of movable parts. Gaussian splatting-based methods, including ArtGS [19] and ArticulatedGS [20], provide an alternative. While ArtGS maintains assumptions similar to DTA, ArticulatedGS relaxes them, but remains limited to recovering only one articulation per optimization step. A more detailed review of these approaches is provided in Appendix A.2.

Collectively, while these methods demonstrate strong performance in a category-agnostic setting, they suffer from a critical limitation – reliance on strong assumptions, such as prior knowledge of the number of articulated joints or even predefined articulation types. This limitation arises from their common strategy of decomposing the problem into the discovery of static and movable object parts, followed by analyzing the motion of the movable parts [17, 18, 19, 20]. Accurate part discovery generally requires prior knowledge of the number of parts to achieve meaningful segmentation, which restricts the applicability of these methods in real-world scenarios.

3 Preliminaries

In this section, we introduce two core components of our approach. First, we briefly review screw theory – a fundamental concept in robotics and rigid-body kinematics that enables efficient modeling of joint motion and spatial transformations. Next, we outline 3D Gaussian splatting, a recent technique for representing and rendering 3D scenes with anisotropic Gaussian primitives.

3.1 Screw Theory

Screw theory provides a natural mathematical formulation for describing the motion of a screw, which involves rotation about an axis combined with translation along the axis [21]. For a given reference frame, a screw axis $\mathcal{S}$ is written as a six-dimensional vector given by:

$$\mathcal{S} = \begin{bmatrix} \omega \\ v \end{bmatrix} \in \mathbb{R}^6, \tag{1}$$

where $\omega \in \mathbb{R}^3$ and $v \in \mathbb{R}^3$, and they satisfy either (i) $||\omega|| = 1$ or (ii) $\omega = 0$ and $||v|| = 1$.

If (i) holds, the screw represents a motion consisting of rotation about the axis ω combined with translation along the same axis. In particular, it satisfies $v = -\omega \times q + h\omega$, where q is an arbitrary point on the screw axis and h is the pitch of the screw. In this paper, we set $h = 0$, as we consider only *revolute joints*, which represent pure rotational motion without pitch. If (ii) holds, the screw represents a motion of pure translation along the axis v, and in this case, it corresponds to a *prismatic joint*.

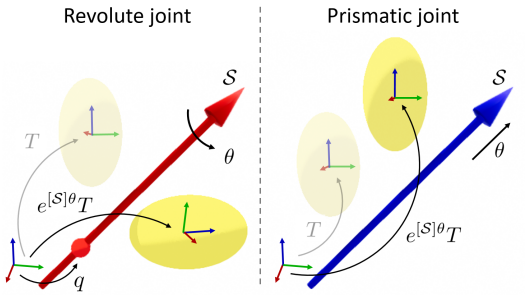

Figure 2: Revolute and prismatic screws axes.

Given a screw axis $\mathcal{S}$ and a joint angle θ, the motion of an arbitrary rigid body coordinate $T \in \mathrm{SE}(3)$ along the screw axis can be expressed using the matrix exponential:

$$T' = e^{[\mathcal{S}]\theta}T, \tag{2}$$

where $T' \in \mathrm{SE}(3)$ denotes the transformed rigid body coordinate, and $[\mathcal{S}]$ is the 4×4 matrix representation of the screw axis $\mathcal{S}$, defined as

$$[\mathcal{S}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times4}, \quad [\omega] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times3}, \tag{3}$$

where $\omega = (\omega_1, \omega_2, \omega_3)$. Figure 2 illustrates the revolute and prismatic screw axes and their corresponding screw motions. The matrix exponential formulation admits closed-form expressions for both revolute and prismatic screw axes; further details are provided in Appendix B.1.

3.2 3D Gaussian Splatting

3D Gaussian splatting was developed for novel-view synthesis from multiple RGB images and can also be used to obtain a 3D representation of scenes [14]. It represents a scene using a set of 3D Gaussians, where the ith Gaussian $\mathcal{G}_i$ is parameterized by the tuple $(T_i, s_i, \sigma_i, c_i)$. Here, $T_i = [R_i, \mu_i] \in \mathrm{SE}(3)$ denotes the pose of the Gaussian, comprising its position $\mu_i \in \mathbb{R}^3$ and orientation $R_i \in \mathrm{SO}(3)$. The parameter $s_i \in \mathbb{R}^3_+$ specifies the scale, with the covariance matrix of the Gaussian given by $\Sigma_i = R_i \mathrm{diag}(s_i)^2 R_i^T$. The term $\sigma_i \in [0, 1]$ denotes the opacity, while c_i represents the surface color of the Gaussian ellipsoid, defined by spherical harmonics coefficients.

To render an RGB image from the colored Gaussians, a typical α-blending approach is used, where α_i is a scaled Gaussian function of the ith Gaussian $\mathcal{G}_i$, defined in 3D space $\mathbb{R}^3$ as

$$\alpha_i(\mathbf{x}) = \sigma_i e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1}(\mathbf{x}-\mu_i)}, \tag{4}$$

where $\mathbf{x} \in \mathbb{R}^3$. The final color C of a pixel is then computed by blending $\mathcal{N}$ ordered Gaussians overlapping the pixel:

$$C = \sum_{i\in\mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{5}$$

The Gaussians are then optimized to minimize the rendering loss function $\mathcal{L}_{\text{render}} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$, where $\mathcal{L}_1$ and $\mathcal{L}_{\text{D-SSIM}}$ denote the L1 loss and the D-SSIM loss between the rendered RGB image and the ground-truth RGB image, respectively. The weight λ is typically set to 0.2.

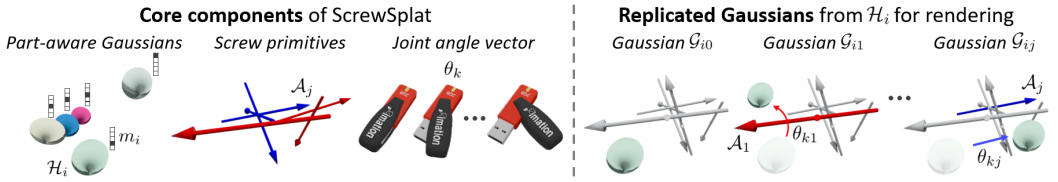

Figure 3: Core components of ScrewSplat (*Left*) and the replicated Gaussians derived from the part-aware Gaussian primitive $\mathcal{H}_i$ (*Right*).

4 ScrewSplat: Integrating Screw Model with 3D Gaussians

This section introduces *ScrewSplat*, a smooth and differentiable formulation of the rendering-based joint optimization problem over part-aware geometry, joint axes, and joint types of articulated objects. We first outline the detailed formulation of ScrewSplat along with the associated optimization procedure. In the following subsection, we present a simple yet effective algorithm for articulated object manipulation that leverages the optimized ScrewSplat as an RGB-based renderer.

4.1 ScrewSplat

In this section, we describe ScrewSplat, including (i) its core components, (ii) the RGB rendering procedure, (iii) the loss function used for optimization, and (iv) additional implementation details. To avoid ambiguity in notation, we first establish our indexing conventions: let i denote the index for Gaussians, j the index for screw axes, and k the index for joint angles of the articulated object. We assume that the observations consist of multi-view RGB images captured under n_a object configurations, such that the index k ranges from from 1 to n_a. Additionally, we assume that all movable parts articulate with respect to a single *static* base part; that is, we do not consider chain structures.

Core Components of ScrewSplat. First, we define n_s *screw primitives*, where the jth screw primitive $\mathcal{A}_j$ is parametrized by a tuple $(\mathcal{S}_j, \gamma_j)$, with $\mathcal{S}_j \in \mathbb{R}^6$ representing a screw axis and $\gamma_j \in [0, 1]$ denoting the confidence. Next, we define n_g *part-aware Gaussian primitives*, where the ith primitive $\mathcal{H}_i$ is parametrized by an augmented tuple $(T_i, s_i, \sigma_i, c_i, m_i)$. Here, $m_i = (m_{i0}, \cdots, m_{in_s}) \in \Delta^{n_s}$ is a probability simplex over $(n_s + 1)$ parts. Specifically, m_{i0} denotes the probability that the Gaussian belongs to the static base part, and m_{ij} for $j \geq 1$ denotes the probability that the Gaussian is associated with the part whose motion is dominated by the jth screw primitive $\mathcal{A}_j$. Lastly, we assign the *joint angle vector* $\theta_k = (\theta_{k1}, \cdots, \theta_{kn_s}) \in \mathbb{R}^{n_s}$ for RGB observations under k'th configuration of the articulated object. The overall core components are illustrated in the left of Figure 3.

RGB Rendering Procedure with ScrewSplat. The key idea behind the RGB rendering procedure is to *replicate* Gaussians from each part-aware Gaussian primitive and assign each replicated Gaussian to either the static base or one of the movable parts. Specifically, we replicate (n_s+1) Gaussians $\mathcal{G}_{ij}$ from the ith part-aware Gaussian primitive $\mathcal{H}_i$, where $j = 0, \cdots, n_s$. Each Gaussian $\mathcal{G}_{ij}$ is assigned to the base part if $j = 0$, and to a movable part associated with screw primitive $\mathcal{A}_j$ if $j \geq 1$. Given a joint angle vector θ_k, the replicated Gaussians are parameterized as:

$$\mathcal{G}_{i0} = (T_i, s_i, \sigma_i m_{i0}, c_i), \quad \mathcal{G}_{ij} = (e^{[\mathcal{S}_j]\theta_{kj}} T_i, s_i, \sigma_i \gamma_j m_{ij}, c_i), \quad \text{for } 1 \leq j \leq n_s. \tag{6}$$

The scale and surface color of each replicated Gaussian are inherited from s_i and c_i of $\mathcal{H}_i$, respectively. The pose is T_i for the base part and $e^{[\mathcal{S}_j]\theta_{kj}} T_i$ for movable parts associated with screw primitive $\mathcal{A}_j$. The opacity is derived by scaling the base opacity σ_i with the part probability m_{ij}, and further modulated by the screw confidence γ_j for movable parts. After replicating a total of $n_g \cdot (n_s + 1)$ Gaussians from the n_g part-aware Gaussians, the final RGB image is rendered using an α-blending approach. An illustration of the replicated Gaussians is shown on the right of Figure 3.

Loss Function. The part-aware Gaussian primitives, screw primitives, and joint angles are jointly optimized to minimize the following loss function:

$$\mathcal{L} = \mathcal{L}_{\text{render}} + \beta \sum_j \sqrt{\gamma_j}, \tag{7}$$

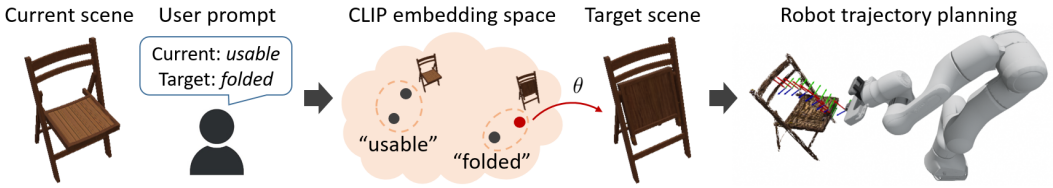

Figure 4: The problem setting for text-guided articulated object manipulation (*Left*), optimization of the target joint angles using CLIP (*Middle*), and corresponding robot trajectory planning (*Right*).

where $\mathcal{L}_{\text{render}}$ is the rendering loss described in Section 3.2, and β is a weighting coefficient set to 0.002. The second term serves as a regularization term – referred to as the *parsimony loss* – which encourages ScrewSplat to represent articulated objects using the smallest possible number of screw primitives. This term not only pushes the model to select a minimal set of screws, but also promotes the identification of the most reliable ones.

Implementation Details. For the part-aware Gaussians, we adopt the same initialization scheme as in Gaussian Splatting, with the exception that the part probabilities $m_i \in \Delta^{n_s}$ are initialized as a uniform distribution. For the screw primitives, the screw axes $\mathcal{S}_j$ are randomly initialized, and all screw confidences γ_j are initialized to 0.9. All joint angle vectors θ_k are initialized as zero vectors.

The optimization procedure generally follows that of the original Gaussian Splatting, with a few additional modifications. The most important modification is that all screw confidences γ_j and part probabilities m_i are periodically reset to 0.9 and uniform distribution, respectively. This periodic reinitialization helps ScrewSplat effectively discover meaningful screw primitives. After optimization, only the screw primitives that satisfy certain criteria remain, ensuring that only the most relevant ones are retained. Further optimization details can be found in Appendix B.2.

4.2 Controlling Joint Angles Using ScrewSplat as a Renderer

The optimized ScrewSplat serves as an RGB image renderer conditioned on the joint angle vector θ; that is, the visual appearance (i.e., RGB image) of the articulated object I from an arbitrary camera pose can be obtained through a continuous – and even differentiable – function π, such that $I = \pi(\theta)$. This function π enables a variety of applications, such as estimating the current pose of the articulated object, by defining an appropriate objective function on the rendered image and optimizing the joint angles accordingly [22].

In this paper, we primarily focus on controlling the joint angles of an articulated object to match a given text prompt using visual foundation models. Specifically, given the current visual appearance of the object I_c and a text description t_c of its current state, along with a target text prompt t_p, our goal is to find a joint angle vector θ such that the rendered appearance $I = \pi(\theta)$ aligns with the target prompt t_p. The problem setting is illustrated on the left of Figure 4.

To achieve this, we utilize the CLIP model, which embeds both RGB images and text into a shared latent space, enabling the computation of similarity between visual and textual inputs [23]. Let e_t and e_I denote the pretrained text and image encoders of CLIP, respectively. In this paper, we adopt a directional CLIP loss [24], defined as:

$$\mathcal{L}_{\text{CLIP-dir}} = 1 - \frac{\triangle I(\theta) \cdot \triangle T}{||\triangle I(\theta)||\,||\triangle T||}, \tag{8}$$

where $\triangle I(\theta) = e_I(\pi(\theta)) - e_I(I_c)$ and $\triangle T = e_T(t_p) - e_T(t_c)$ represent the directional shifts in the CLIP latent space. We optimize joint angle vector θ using Bayesian optimization. This process is illustrated in the middle of Figure 4.

After obtaining the target joint angle vector θ, we plan a simple trajectory for the robot end-effector tip and generate a kinematically feasible robot trajectory accordingly, as shown on the right of Figure 4. Further details of the optimization and planning procedure are provided in Appendix B.3.

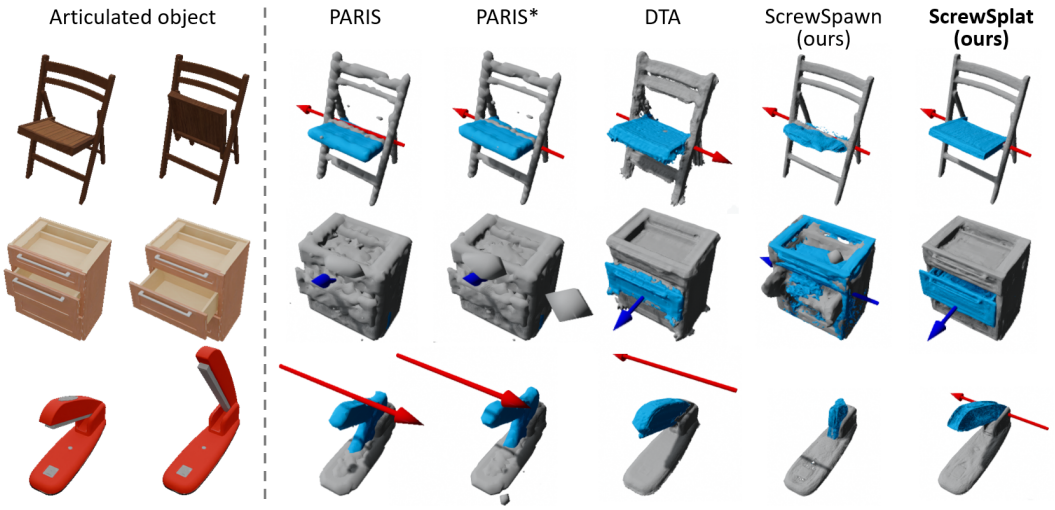

Figure 5: Reconstructed meshes and screw axes for single-joint objects using each method. Static parts are shown in gray, movable parts in cyan, revolute joints in red, and prismatic joints in blue.

Table 1: Recognition performance for single-joint objects, averaged across all instances. Object-wise recognition performance is provided in Appendix D.1.

	Geometry (↓)			Motion (↓)		Appearance (↑)	
METHOD	CD-s	CD-m	CD-w	Ang.	Pos.	PSNR	SSIM
PARIS [16]	54.015	18.032	40.192	17.656	2.020	28.64	0.970
PARIS* [16]	49.706	8.864	33.856	16.287	1.742	28.66	0.970
DTA [17]	0.538	0.528	0.360	0.437	0.308	-	-
ScrewSpawn (ours)	0.617	11.566	0.946	24.869	0.902	29.11	0.982
ScrewSplat (ours)	**0.319**	**0.211**	**0.261**	**0.084**	**0.010**	**38.07**	**0.993**

5 Experiments

In this section, we empirically demonstrate that (i) ScrewSplat outperforms existing state-of-the-art methods in recognizing articulated objects, and (ii) our method can be effectively applied to text-guided articulated object manipulation in both simulated and real-world settings, followed by successful robot-object interaction.

Baseline Methods. We compare ScrewSplat with the following baselines: *PARIS* [16], *PARIS**, *DTA* [17], and *ScrewSpawn*. PARIS and PARIS* (PARIS augmented with depth data) recover a single joint axis assuming a known joint type, while DTA discovers a predefined number of joint axes. Since only DTA supports multiple joint axes, we compare against DTA for multi-joint objects. These methods recover kinematic structures from multi-view images under two different object configurations; PARIS uses RGB input, while PARIS* and DTA use RGB-D. We also introduce ScrewSpawn, an ablation model that follows the ScrewSplat framework but spawns only a single screw (with a known joint type). This model is used to validate the necessity of "splatting" multiple screw primitives in ScrewSplat. Detailed implementations can be found in Appendix C.1.

Dataset. We select ten single-joint objects and three multi-joint objects from distinct categories in the PartNet-Mobility dataset [25]. Using Blender [26], a photorealistic renderer, we obtain multi-view RGB images and depth images (used for training PARIS* and DTA) under varying object configurations. We place 48 camera poses uniformly over a hemisphere centered on each object. For ScrewSplat and ScrewSpawn, we use images captured under five configurations, while for PARIS, PARIS*, and DTA, we use images from two configurations selected from the same five. For single-joint objects, the five configurations correspond to evenly spaced joint angles along the motion range. For multi-joint objects, five randomly sampled joint angle vectors are used. Further details for the evaluation dataset are provided in Appendix C.1.

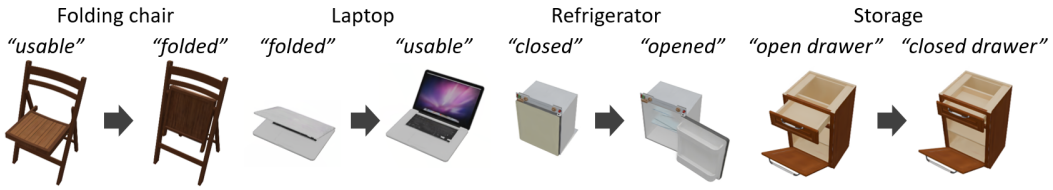

Figure 7: Text-guided articulated object manipulation results in simulation.

Table 2: Recognition performance on multi-joint objects, averaged over all instances, geometries, and screw axes. Detailed object-wise recognition results are provided in Appendix D.1.

METHOD	Geometry (↓)			Motion (↓)		Appearance (↑)	
	CD-s	CD-m$_m$	CD-w	Ang$_m$.	Pos$_m$.	PSNR	SSIM
DTA [17]	**0.568**	5.647	**0.476**	7.233	28.877	-	-
ScrewSplatting(ours)	0.675	**0.096**	0.666	**0.130**	**0.002**	**36.76**	**0.987**

5.1 Articulated Object Recognition Performance

We compare the performance of ScrewSplat with the baselines. To evaluate the recognition quality of articulated objects, we adopt three types of metrics: *geometry*, *motion*, and *appearance*. The geometry metric includes the bi-directional Chamfer-$l2$ distance between point clouds sampled from the reconstructed and ground-truth meshes. We report this metric separately for the static part (CD-s, mm), the movable parts (CD-m, mm), and the entire object (CD-w, mm). The motion metric includes the angular error (Ang., °) between the estimated and ground-truth screw axes, and the axis position error (Pos., 0.1m) for revolute joints, calculated as the minimum distance between corresponding screw axes. The appearance metric includes Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM), computed on rendered images under unseen joint angles. Details of the evaluation metrics and their computation are available in Appendix C.1.

Single-joint Objects. Figure 5 shows the recognition results, including reconstructed meshes and screw axes, on three representative single-joint objects. PARIS and PARIS* performed well on the folding chair but failed to recognize the other objects. DTA generally outperforms PARIS, but still fails in certain cases such as the stapler. ScrewSplat consistently demonstrates the best performance across all cases, both in terms of geometry reconstruction and kinematic structure estimation. In contrast, ScrewSpawn fails to accurately recover both geometry and kinematic structure in most cases, highlighting the effectiveness of ScrewSplat's multi-screw formulation. Table 1 shows the quantitative recognition results. We demonstrate that ScrewSplat achieves the highest overall performance across geometry, motion, and visual appearance. PARIS and PARIS* struggle to recognize almost all of the other objects. ScrewSpawn is able to reconstruct the static parts to some extent but fails to accurately estimate the movable parts and screw axes. DTA performs slightly worse than our method but achieves comparable results overall. Although our method uses data from a wider range of joint configurations, it is particularly notable that it achieves the best performance while relying solely on RGB inputs and without any prior knowledge of the joints. Detailed object-wise results are provided in Appendix D.1.

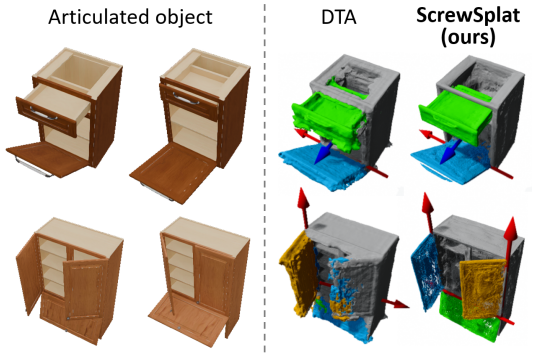

Figure 6: Reconstructed meshes and screw axes for multi-joint objects using each method.

Multi-joint Objects. Figure 6 presents the recognition results on multi-joint objects. DTA successfully predicts both screw axes for objects with two screws (top row of Figure 6), but fails to correctly identify more than one axis in objects with three screws (bottom row). In contrast, ScrewSplat successfully recognizes all screw axes in both cases. Quantitative recognition results for multi-joint objects are provided in Table 2. While DTA performs slightly better on static parts and whole-object

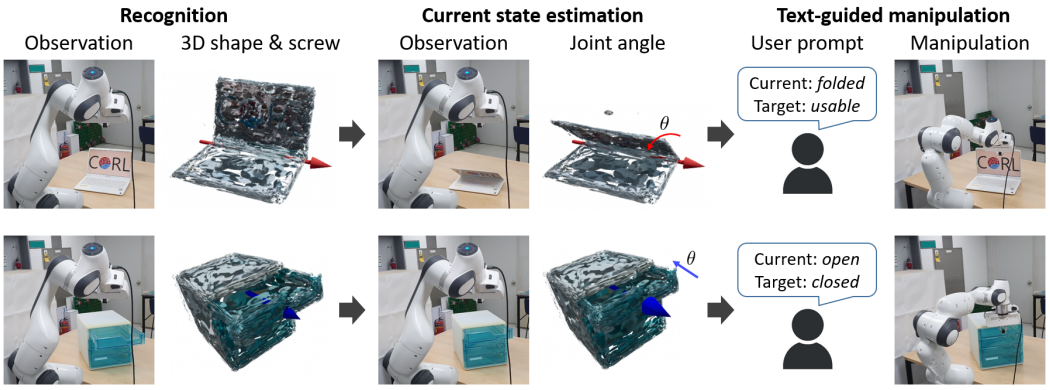

Figure 8: Overall pipeline for text-guided robotic manipulation of the real-world articulated objects.

geometry, ScrewSplat achieves substantially higher accuracy in reconstructing the geometry of movable parts and estimating joint axes. Overall, ScrewSplat consistently outperforms prior methods in recognizing articulated objects, both for single-joint and multi-joint objects.

5.2 Articulated Object Manipulation Results

In this section, we demonstrate the effectiveness of ScrewSplat on the task of text-guided articulated object manipulation in both simulated and real-world settings.

Simulated Articulated Object Manipulation. Figure 7 shows the results of text-guided joint angle optimization using ScrewSplat models optimized on simulated articulated objects. By providing a suitable description of the current state along with a target text prompt, the objects can be manipulated into the desired configuration. We observe that for single-joint objects, simple prompts are sufficient to find the desired configurations, whereas multi-joint objects require more specific instructions (e.g., "closed drawer" instead of just "closed").

Real-world Articulated Object Manipulation. For real-world scenarios where a robot manipulator physically adjust the joint angles of articulated objects, the overall manipulation pipeline consists of three main stages, as illustrated in Figure 8: (i) a *recognition* step, where ScrewSplat is optimized using multi-view RGB observations collected under several joint configurations manually manipulated by a human; (ii) a *current state estimation* step, which estimates the object's current joint angle by optimizing an appropriate loss function; and (iii) a *text-guided manipulation* step, which determines the target joint angle based on a given text prompt and executes the corresponding robot manipulation. As shown on the left of Figure 8, ScrewSplat accurately reconstructs the shape and kinematic structure of real-world objects – including even a translucent drawer. A well-trained ScrewSplat further enables precise estimation of the current joint angle and facilitates successful text-guided object manipulation, as shown in the middle and right of Figure 8, respectively. Further details for real-world text-guided manipulation are provided in Appendix C.2 and D.3, respectively.

6 Conclusion

We propose ScrewSplat, a novel end-to-end framework for articulated object recognition that operates solely on RGB observations. By leveraging screw theory and Gaussian splatting, and introducing confidence scores over screw axes along with a part probability simplex for Gaussians, our formulation enables smooth and unified optimization over both geometric and kinematic components. Unlike prior approaches, ScrewSplat avoids strong assumptions, complex intermediate steps, and reliance on depth data, resulting in a more robust and generalizable solution. We also demonstrate that ScrewSplat shows state-of-the-art performance in recovering the geometry and kinematic structure of both single- and multi-joint articulated objects. Furthermore, we show that ScrewSplat can be directly applied to zero-shot, text-guided manipulation of articulated objects, enabling robots to physically adjust joint angles according to high-level user intent in real-world environments.

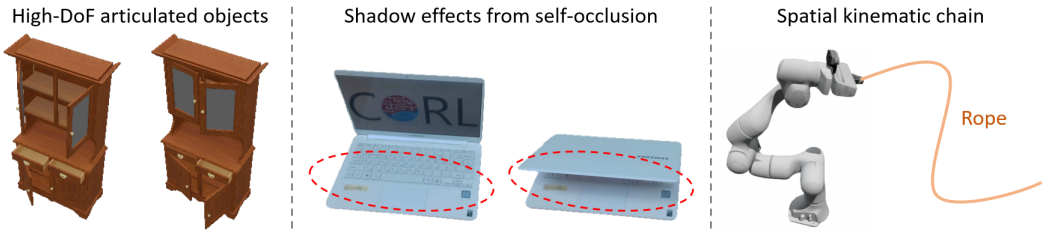

Figure 9: Limitations and Future Directions of ScrewSplat.

Limitations and Future Directions

The recognition performance of ScrewSplat is sensitive to the weight β of the parsimony loss, particularly for multi-joint objects, making it crucial to find an appropriate β. Specifically, when β is set too low, ScrewSplat attempts to recognize the articulated object using many screws, leading to a trivial solution where the same movable part in each configuration is assigned to different screws. Conversely, when β is set too high, the algorithm fails to detect any movable parts and recognizes the object as static. Due to these tendencies, ScrewSplat still struggles to identify all reliable joint axes for *high-DoF articulated objects*, such as object with six joints shown in the left of Figure 9. We aim to address this limitation in the future by developing stable optimization techniques. A key future direction will be to explore methods for properly initializing the screw axes at the beginning, and to develop adaptive techniques for adjusting β based on the current recognition performance during optimization. Additionally, inspired by the optimization techniques in Gaussian Splatting [14], we will explore techniques to either remove or densify screw axes during optimization.

One of the challenges we observed in articulated object recognition is accounting for the *shadow effects caused by movable parts*. As shown in the middle of Figure 9, different configurations of a movable part cast varying shadow effects on the static parts in the RGB image. In other words, the same part (or the same part-aware Gaussian primitive) should exhibit different colors depending on the joint angle. With a sufficiently low β, when these shadow effects are significant, ScrewSplat may discover additional Gaussians or even identify new parts to model the effect. To find the appropriate kinematic structure, it is crucial to model these effects, and this remains an area for future work. Possible future directions include formulating a method that explicitly optimizes for light information to model these shadow effects, as suggested in [27], and modeling the color of Gaussians as a function of joint configurations – using additional deformation functions such as Implicit Linear Blend Skinning (LBS) [22, 28] – and optimizing it directly.

Lastly, a promising extension of ScrewSplat is the ability to recognize articulated objects using a *spatial kinematic chain*. Consider a kinematic chain in which a screw $\mathcal{S}_1$ is attached to a static base part, and subsequent screws $\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_n$ are serially connected. Given the joint angles $\theta_1, \theta_2, ..., \theta_n$ corresponding to each screw, the motion of an arbitrary rigid body coordinate $T \in \mathrm{SE}(3)$ attached to the nth movable part can be described by the following *product of exponentials formula* [21]:

$$T' = e^{[\mathcal{S}_1]\theta_1} e^{[\mathcal{S}_2]\theta_2} \cdots e^{[\mathcal{S}_n]\theta_n} T, \tag{9}$$

where $T' \in \mathrm{SE}(3)$ represents the transformed coordinate. We have focused on objects where all movable parts articulate with respect to a single static base part, but there are many objects, such as a robotic manipulator as shown in the right of Figure 9, where modeling a spatial kinematic chain is inevitable. In future work, we aim to extend ScrewSplat by leveraging the product of exponentials formula to recognize the geometry and kinematic structure of spatial chains, such as robots, using only RGB images – this direction aligns with a recent work that uses full point clouds [29]. Furthermore, we plan to explore the recognition of one-dimensional deformable objects such as ropes, in the spirit of pseudo-rigid-body theory, and use the recognized model to develop effective manipulation strategies [30, 31], which will also be a direction for future work.

Acknowledgments

S. Kim, J. Ha, Y. H. Kim, and F. C. Park were supported in part by IITP-MSIT under Grant RS-2021-II212068 (SNU AI Innovation Hub), in part by IITP-MSIT under Grant 2022-220480 and Grant RS-2022-II220480 (Training and Inference Methods for Goal Oriented AI Agents), in part by IITP-MSIT under Grant RS-2024-00436680 (Collaborative Research Projects with Microsoft Research) through Global Research Support Program in the Digital Field program, in part by KIAT under Grant P0020536 (HRD Program for Industrial Innovation), in part by SRRC NRF under Grant RS-2023-00208052, in part by SNU-IPAI, in part by SNU-AIIS, in part by SNU-IAMD, in part by SNU Institute for Engineering Research, in part by Hyundai Motor Company and Kia, and in part by Microsoft Research Asia.

References

[1] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3706–3715, 2020.

[2] V. Zeng, T. E. Lee, J. Liang, and O. Kroemer. Visual identification of articulated object parts. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2443–2450. IEEE, 2021.

[3] A. Jain, R. Lioutikov, C. Chuck, and S. Niekum. Screwnet: Category-independent articulation model estimation from depth images using screw theory. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13670–13677. IEEE, 2021.

[4] J. Mu, W. Qiu, A. Kortylewski, A. Yuille, N. Vasconcelos, and X. Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13001–13011, 2021.

[5] W.-C. Tseng, H.-J. Liao, L. Yen-Chen, and M. Sun. Cla-nerf: Category-level articulated neural radiance field. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8454–8460. IEEE, 2022.

[6] A. Jain, S. Giguere, R. Lioutikov, and S. Niekum. Distributional depth-based estimation of object articulation models. In *Conference on Robot Learning*, pages 1611–1621. PMLR, 2022.

[7] F. Wei, R. Chabra, L. Ma, C. Lassner, M. Zollhöfer, S. Rusinkiewicz, C. Sweeney, R. Newcombe, and M. Slavcheva. Self-supervised neural articulated shape and appearance models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15816–15826, 2022.

[8] Z. Jiang, C.-C. Hsu, and Y. Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5616–5626, 2022.

[9] N. Heppert, T. Migimatsu, B. Yi, C. Chen, and J. Bohg. Category-independent articulated object tracking with factor graphs. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3800–3807. IEEE, 2022.

[10] N. Heppert, M. Z. Irshad, S. Zakharov, K. Liu, R. A. Ambrus, J. Bohg, A. Valada, and T. Kollar. Carto: Category and joint agnostic reconstruction of articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21201–21210, 2023.

[11] J. Lei, C. Deng, W. B. Shen, L. J. Guibas, and K. Daniilidis. Nap: Neural 3d articulated object prior. *Advances in Neural Information Processing Systems*, 36:31878–31894, 2023.

[12] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[13] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[14] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[15] B. Wen, J. Tremblay, V. Blukis, S. Tyree, T. Müller, A. Evans, D. Fox, J. Kautz, and S. Birchfield. Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 606–617, 2023.

[16] J. Liu, A. Mahdavi-Amiri, and M. Savva. Paris: Part-level reconstruction and motion analysis for articulated objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 352–363, 2023.

[17] Y. Weng, B. Wen, J. Tremblay, V. Blukis, D. Fox, L. Guibas, and S. 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*, pages 3141–3150, 2024.

[18] J. Kerr, C. M. Kim, M. Wu, B. Yi, Q. Wang, K. Goldberg, and A. Kanazawa. Robot see robot do: Imitating articulated object manipulation with monocular 4d reconstruction. *arXiv preprint arXiv:2409.18121*, 2024.

[19] Y. Liu, B. Jia, R. Lu, J. Ni, S.-C. Zhu, and S. Huang. Building interactable replicas of complex articulated objects via gaussian splatting. *arXiv preprint arXiv:2502.19459*, 2025.

[20] J. Guo, Y. Xin, G. Liu, K. Xu, L. Liu, and R. Hu. Articulatedgs: Self-supervised digital twin modeling of articulated objects using 3d gaussian splatting. *arXiv preprint arXiv:2503.08135*, 2025.

[21] K. M. Lynch and F. C. Park. *Modern robotics*. Cambridge University Press, 2017.

[22] R. Liu, A. Canberk, S. Song, and C. Vondrick. Differentiable robot rendering. *arXiv preprint arXiv:2410.13851*, 2024.

[23] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[24] R. Gal, O. Patashnik, H. Maron, A. H. Bermano, G. Chechik, and D. Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022.

[25] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.

[26] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org.

[27] L. Bolanos, S.-Y. Su, and H. Rhodin. Gaussian shadow casting for neural characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20997–21006, 2024.

[28] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 851–866. 2023.

[29] J. Lin, L. Zhang, K. Lee, J. Ning, J. Goldfeder, and H. Lipson. Autourdf: Unsupervised robot modeling from point cloud frames using cluster registration. *arXiv preprint arXiv:2412.05507*, 2024.

[30] V. Viswanath, K. Shivakumar, M. Parulekar, J. Ajmera, J. Kerr, J. Ichnowski, R. Cheng, T. Kollar, and K. Goldberg. Handloom: Learned tracing of one-dimensional objects for inspection and manipulation. In *Conference on Robot Learning*, pages 341–357. PMLR, 2023.

[31] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song. Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects. *The International Journal of Robotics Research*, 43(4):389–404, 2024.

[32] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen. Shape completion enabled robotic grasping. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2442–2447. IEEE, 2017.

[33] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3907–3916, 2018.

[34] S. Liu, S. Saito, W. Chen, and H. Li. Learning to infer implicit surfaces without 3d supervision. *Advances in Neural Information Processing Systems*, 32, 2019.

[35] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[36] M. Van der Merwe, Q. Lu, B. Sundaralingam, M. Matak, and T. Hermans. Learning continuous 3d reconstructions for geometrically aware grasping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11516–11522. IEEE, 2020.

[37] J. Ichnowski, Y. Avigal, J. Kerr, and K. Goldberg. Dex-nerf: Using a neural radiance field to grasp transparent objects. *arXiv preprint arXiv:2110.14217*, 2021.

[38] J. Kerr, L. Fu, H. Huang, Y. Avigal, M. Tancik, J. Ichnowski, A. Kanazawa, and K. Goldberg. Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects. In *6th annual conference on robot learning*, 2022.

[39] S. Kim, T. Ahn, Y. Lee, J. Kim, M. Y. Wang, and F. C. Park. Dsqnet: A deformable model-based supervised learning algorithm for grasping unknown occluded objects. *IEEE Transactions on Automation Science and Engineering*, 20(3):1721–1734, 2022.

[40] Q. Dai, Y. Zhu, Y. Geng, C. Ruan, J. Zhang, and H. Wang. Graspnerf: Multiview-based 6-dof grasp detection for transparent and specular objects using generalizable nerf. *arXiv preprint arXiv:2210.06575*, 2022.

[41] S. Kim, Y. H. Kim, Y. Lee, and F. C. Park. Leveraging 3d reconstruction for mechanical search on cluttered shelves. In *7th Annual Conference on Robot Learning*, 2023.

[42] Y. H. Kim, S. Kim, Y. Lee, and F. C. Park. T^2sqnet: A recognition model for manipulating partially observed transparent tableware objects. In *8th Annual Conference on Robot Learning*, 2024.

[43] Y. H. Kim, S. Kim, Y. Lee, and F. C. Park. Dreamgrasp: Zero-shot 3d multi-object reconstruction from partial-view images for robotic manipulation. *arXiv preprint arXiv:2507.05627*, 2025.

[44] B. Abbatematteo, S. Tellex, and G. Konidaris. Learning to generalize kinematic models to novel objects. In *Proceedings of the 3rd Conference on Robot Learning*, 2019.

[45] L. Le, J. Xie, W. Liang, H.-J. Wang, Y. Yang, Y. J. Ma, K. Vedder, A. Krishna, D. Jayaraman, and E. Eaton. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. *arXiv preprint arXiv:2410.13882*, 2024.

[46] S. Y. Gadre, K. Ehsani, and S. Song. Act the part: Learning interaction strategies for articulated object part discovery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15752–15761, 2021.

[47] C.-C. Hsu, Z. Jiang, and Y. Zhu. Ditto in the house: Building articulation models of indoor scenes through interactive perception. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3933–3939. IEEE, 2023.

[48] L. Ma, J. Meng, S. Liu, W. Chen, J. Xu, and R. Chen. Sim2real 2: Actively building explicit physics model for precise articulated object manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11698–11704. IEEE, 2023.

[49] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.

[50] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.

Appendix

A Extended Related Works

A.1 Articulated Object Recognition Using Supervised Learning

There has been considerable interest in recognizing articulated objects within a supervised learning framework, where deep neural networks are trained to predict articulation parameters (e.g., joint axes and joint angles) directly from raw visual input [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. Some of these methods also aim to simultaneously reconstruct the part-aware geometry of objects [5, 8, 10], which aligns with our objectives. While the problem of 3D recognition of rigid objects is well studied and has numerous established solutions [32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43], the recognition of articulated objects is significantly more challenging, as it requires inferring not only object geometry but also the underlying kinematic structure. Consequently, it remains an open problem.

Early studies often addressed articulated object recognition problem by employing category-driven approaches, using category information to assist in identifying the kinematic structure, and testing on unseen objects belonging to known categories [1, 4, 7, 44]. More recently, to handle a broader range of objects beyond those in known categories, category-agnostic recognition approaches have been proposed. These methods primarily aim to reconstruct part-level geometry and kinematic structures, which closely align with our objective. For example, CARTO [10] predicts the geometry as a defomable signed distance function (SDF) representations – similar to the category-specific A-SDF [4] but removes the reliance on object categories. Ditto [8] predicts articulable neural occupancy fields to generate digital twins of articulated objects from point cloud observations under two different articulation states. Articulate-Anything [45], a recent approach, integrates a vision-language foundation model to recognize part-aware geometries and kinematic structures, and incorporates interactable digital twins into simulators for sim-to-real robot learning. They demonstrate generalizability to unseen objects within similar categories, and even to unseen categories through foundation models. However, they inherently struggle to generalize to objects that differ significantly from the training categories.

A.2 Articulated Object Recognition Using Per-object Optimization Methods

Several works have attempted to recognize articulated objects by directly fitting 3D representations and kinematic structures to observations without any supervision. Since these methods typically perform optimization for each object individually, they are often referred to as per-object optimization methods. A representative early work in per-object optimization methods is PARIS [16], which presents a method based on neural radiance fields (NeRF). Specifically, PARIS defines separate radiance fields for movable and static parts, performing joint rendering and optimization to achieve part-level reconstruction and articulation discovery. However, this approach is only applicable to single-joint objects, i.e., articulated objects with only one movable part, with known joint types.

Recently, approaches capable of covering multi-joint objects have been proposed. A notable example is DTA [17], which first reconstructs two entire meshes using RGB-D data from observations under two configurations of articulated objects, and then determines the kinematic structure using a feature correspondence matching module. During the mesh reconstruction process, depth images are used, and in the feature correspondence matching step, the number of movable parts must also be known. DTA successfully infers the kinematic structure of articulated objects with multiple movable parts using this additional information. Subsequently, research utilizing Gaussian splatting [14], such as ArtGS [19] and ArticulatedGS [20], has emerged as an alternative to neural radiance fields. These approaches are somewhat similar to ours in that they leverage Gaussian splatting. ArtGS, like DTA, utilizes point correspondence matching and similarly requires depth images and knowledge of the number of movable parts. ArticulatedGS, however, does not rely on these assumptions but is still limited to discovering only one articulation information per optimization step.

While these methods demonstrate strong performance without supervision, the review above highlights several limitations. The most important limitation is that they rely on assumptions such as the articulated object has a single joint axis, or the user should know the number of articulated joints, or even predefined articulation types. Some works also rely on auxiliary depth inputs, which are often noisier than RGB images – particularly for transparent or reflective surfaces – thus limiting their robustness in real-world scenarios. Finally, it is important to emphasize that these methods often involve multi-stage pipelines with intermediate procedures like point correspondence matching, which not only increase overall complexity, but also contrasts with our simple, end-to-end framework that operates soly on RGB observations and does not rely on such assumptions.

A.3 Articulation Discovery via Robot-Object Interaction

Beyond observation-based recognition approaches, several works have explored active interaction strategies that allow a robot to interact with unknown articulated objects and collect additional observations for articulation reasoning [46, 47, 48]. These methods often rely on pre-trained networks to guide interaction, rather than some learning-free strategies that cannot extract joint parameters from static observations. For instance, [46] uses an RGB image as input to predict hold and push locations via a trained network. The predicted actions are then executed by a human to modify the articulated object's configuration, and subsequent observations are used to infer the kinematic structure. Similar methods such as [47] and [48] also leverage 3D point cloud inputs to predict interaction positions and directions. Using point cloud observation data as 3D geometric information enables the robot to interact with the object and actively acquire additional observations.

The observation-based recognition approaches described above have drawbacks compared to the interaction-based methods discussed here. One major limitation is that collecting such data typically requires manual manipulation, which is labor-intensive and difficult to automate. While interaction-based methods are not yet perfect – often still requiring additional human interaction or failing to produce appropriate interactions for novel articulated objects – we believe that integrating observation-based recognition with interaction-based approaches is a promising future research direction that could lead to more effective recognition methods for articulated objects.

B Implementation Details for ScrewSplatting

B.1 Details for Screw Theory

In this section, we describe the closed-form matrix exponential expressions for both revolute and prismatic screw axes [21]. For convenience, we briefly review the screw theory outlined in Section 3.1. A six-dimensional screw axis $\mathcal{S}$ is given by:

$$\mathcal{S} = \begin{bmatrix} \omega \\ v \end{bmatrix} \in \mathbb{R}^6. \tag{10}$$

For a revolute joint, the screw axis satisfies $||\omega|| = 1$ and $v = -\omega \times q$, where q is an arbitrary point on the screw axis. For a prismatic joint, $\omega = 0$ and $||v|| = 1$. Given a screw axis $\mathcal{S}$ and a joint angle θ, the motion of an arbitrary rigid body coordinate $T \in \mathrm{SE}(3)$ along the screw axis can be expressed using the matrix exponential:

$$T' = e^{[\mathcal{S}]\theta} T, \tag{11}$$

where $T' \in \mathrm{SE}(3)$ denotes the transformed rigid body coordinate, and $[\mathcal{S}]$ is the 4×4 matrix representation of the screw axis $\mathcal{S}$, defined as

$$[\mathcal{S}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times4}, \quad [\omega] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times3}, \tag{12}$$

where $\omega = (\omega_1, \omega_2, \omega_3)$.

In general, the matrix exponential $e^{[\mathcal{S}]\theta}$ is computed using its series expansion:

$$\begin{aligned} e^{[\mathcal{S}]\theta} &= I + [\mathcal{S}]\theta + [\mathcal{S}]^2\frac{\theta^2}{2!} + [\mathcal{S}]^3\frac{\theta^3}{3!} + \cdots & (13) \\ &= \begin{bmatrix} e^{[\omega]\theta} & G(\theta)v \\ 0 & 1 \end{bmatrix}, & (14) \end{aligned}$$

where $G(\theta)$ represents the function that generates the translational part of the motion, and is given by:

$$G(\theta) = I\theta + [\omega]\frac{\theta^2}{2!} + [\omega]^2\frac{\theta^3}{3!} + \cdots \tag{15}$$

For a revolute joint, $G(\theta)$ has a closed-form expression using the fact that $[\omega]^3 = [\omega]$:

$$G(\theta) = I\theta + (1 - \cos\theta)[\omega] + (\theta - \sin\theta)[\omega]^2. \tag{16}$$

Thus, the closed-form matrix exponential expression for the revolute joint is given by:

$$e^{[\mathcal{S}]\theta} = \begin{bmatrix} e^{[\omega]\theta} & \left(I\theta + (1 - \cos\theta)[\omega] + (\theta - \sin\theta)[\omega]^2\right) v \\ 0 & 1 \end{bmatrix}, \tag{17}$$

where

$$e^{[\omega]\theta} = I + \sin\theta[\omega] + (1 - \cos\theta)[\omega]^2. \tag{18}$$

The equation for $e^{[\omega]\theta} \in \mathrm{SO}(3)$ is known as *Rodrigues' formula*.

For a prismatic joint, the term $e^{[\omega]\theta}$ becomes the identity matrix I, and $G(\theta)$ simplifies to $I\theta$. Therefore, the closed-form matrix exponential expression for the prismatic joint is given by:

$$e^{[\mathcal{S}]\theta} = \begin{bmatrix} I & v\theta \\ 0 & 1 \end{bmatrix}. \tag{19}$$

B.2 Implementation Details for ScrewSplat

In this section, we describe the implementation details of ScrewSplat and its optimization process, including the initialization of core components, periodic re-initialization, selection of meaningful kinematic structures, and additional details related to the optimization process.

Initialization of Core Components. For the part-aware Gaussian primitives, we initially spawn 10,000 primitives. For each primitive $\mathcal{H}_i = (T_i, s_i, \sigma_i, c_i, m_i)$, we generally adopt the same initialization scheme as used in the original Gaussian Splatting method for the variables $(T_i, s_i, \sigma_i, c_i)$. It is worth noting that the position μ_i of the Gaussian primitive from the pose $T_i = [R_i, \mu_i]$ is sampled from a uniform distribution: in the simulation environment, it is sampled from $[-1, 1]^3$, , and in real-world experiments, it is sampled from $[-0.7, 0.7]^3$. We found that initializing the positions within the camera's range results in better performance. The part probability $m_i \in \Delta^{n_s}$ is initialized as a uniform distribution (i.e., an element-wise constant vector).

For the screw primitives, we initially spawn eight revolute joint axes and eight prismatic joint axes (i.e., a total of 16 joint axes). For each screw primitive $\mathcal{A}_j = (\mathcal{S}_j, \gamma_j)$, we first sample a six-dimensional real vector from a uniform distribution over $[-0.5, 0.5]^3$ and then normalize it to satisfy the constraints for both revolute and prismatic joints. Specifically, let the sampled six-dimensional vector be $[x, q]$, where $x \in \mathbb{R}^3$ and $q \in \mathbb{R}^3$. For a revolute joint $[\omega, v]$, ω is set to $x/||x||$ and v is set to $-\omega \times q$. For a prismatic joint $[\omega, v]$, ω is set to the zero vector, and v is set to $q/||q||$. The screw confidence γ_j is initialized to 0.9.

For the joint angle vectors, we generate as many joint angle vector variables as there are configurations used in the observations. The joint angle θ_k is initialized to the zero vector.

Periodic Re-initialization of Part-aware Components. We periodically reset all screw confidences γ_j and part probabilities m_i. This periodic reset helps ScrewSplat effectively discover meaningful screw primitives. All screw confidences γ_j and part probabilities m_i are periodically reset to 0.9 and uniform distributions, respectively. We note that the reset period should be asynchronous with the original Gaussian Splatting opacity reset period for effectiveness. If the iterations are synchronized, we observe that ScrewSplat deteriorates significantly during that iteration. Additionally, at the same interval, we also re-initialize the joint angles θ_k and the poses of all part-aware Gaussian primitives T_i. First, we randomly select one joint angle θ_m from the set $\{\theta_1, \cdots, \theta_{n_a}\}$, and then convert all joint angles θ_k by substracting θ_m, i.e., $\theta \leftarrow \theta_k - \theta_m$. For each pose T_i, we first select the part index j^* corresponding to the highest value in the m_i vector (i.e., the index j with the highest m_{ij} in m_i). Then, we convert T_i as follows:

$$T_i \leftarrow e^{[S_{j^*}]\theta_{mj^*}} T_i. \tag{20}$$

This re-initialization of joint angles and poses has the effect of moving all Gaussian primitives to a canonical pose. Combined with the reset of screw confidences and part probabilities, this process synergistically aids in the discovery of appropriate geometries and kinematic structures.

Selecting Meaningful Screw Primitives. Once ScrewSplat has converged to some extent, we eliminate meaningless screw primitives and fine-tune the ScrewSplat model using only the meaningful screws for a certain number of iterations. There are two main criteria for eliminating meaningless screw primitives. The first criterion involves screw primitives with a confidence γ_j below a certain threshold (we set this threshold to 0.1). These primitives have little impact on rendering and are therefore considered trivially eliminated. In this case, the part-aware Gaussian primitives associated with these screw primitives are also removed. The second criterion involves screw primitives $\mathcal{A}_j$ where the difference between the maximum and minimum values in the set $\{\theta_{1j}, \cdots, \theta_{n_a j}\}$ (i.e., the interval of the joint angle bounds) is below a certain threshold (we set this threshold to 0.1 for revolute joints and 0.03 for prismatic joints). A small interval indicates that the joint angles have converged to constant values, meaning the corresponding part-aware Gaussian primitives should originally belong to the static base. In this case, we delete the screw primitive but re-initialize the Gaussian primitives to belong to the static base. Specifically, we randomly select a θ from the set $\{\theta_{1j}, \cdots, \theta_{n_a j}\}$, and then we convert T_i as follows, similar to the re-initialization process described

above:

$$T_i \leftarrow e^{[S_j]\theta} T_i. \tag{21}$$

Learning Rates and Hyperparameters. For the variables $(T_i, s_i, \sigma_i, c_i)$, used in Gaussian Splatting, we use the same parameters as those previously employed. For the part probability m_i, a learning rate of 0.1 is applied before passing through the softmax. For the screw axis $\mathcal{S}_j$, a learning rate of 0.003 is used before normalization. The screw confidence γ_j uses a learning rate of 0.01 before passing through the sigmoid. The joint angle θ_k uses a learning rate of 0.01.

Efficient Rendering and Backpropagation. To speed up the rendering and backward process during optimization, ScrewSplat only considers screw primitives with a confidence γ_j greater than a certain threshold (we set this threshold to 0.1) for rendering the RGB.

B.3 Details for Articulated Object Manipulation

In this section, we describe the details of articulated object manipulation, including additional information on controlling joint angles with text prompts using ScrewSplat, as discussed in Section 4.2, details on the current state estimation step mentioned in Section 5.2, and a brief explanation of robot trajectory planning.

Controlling Joint Angles Using ScrewSplat. For the CLIP model, we use `openai/clip-vit-base-patch32` from Huggingface. For Bayesian optimization, We use the `gp_minimize` function from the `scikit-optimize` library to optimize the joint angles. The optimization is guided by the Expected Improvement (EI) acquisition function and is performed over 50 function evaluations, with the first 10 being random joint angle samples. For recognition, we calculate the element-wise min and max of the optimized joint angles θ_k to set the joint limits, and the search space is defined based on these joint limits. The rationale behind using directional CLIP loss instead of simple cosine similarity loss, and Bayesian optimization instead of gradient-based optimization, is discussed in detail in the Appendix D.3.

Current State Estimation Step. As discussed in Section 5.2, for a changed articulated object configuration after recognition, we additionally estimate the object's current joint angle. To achieve this, we first obtain additional RGB observations of the articulated object, and then minimize the rendering loss $\mathcal{L}_render$ used in Gaussian Splatting to find the current joint angle:

$$\mathcal{L}_{\text{estimate}} = \mathcal{L}_{\text{render}}. \tag{22}$$

For optimization, we also use Bayesian optimization, and the parameters employed are the same as those used in the text-guided object manipulation described above.

Robot Trajectory Planning for Real-world Articulated Object Manipulation. Given the current joint angle $\theta_c \in \mathbb{R}$ and target joint angle $\theta_t \in \mathbb{R}$, along with a screw primitive $\mathcal{A}_j = (\mathcal{S}_j, \gamma_j)$, we propose a simple robot trajectory planning approach for real-world articulated object manipulation. The trajectory planning consists of two main stages: first, planning the robot gripper's tip trajectory, and second, planning the gripper's $\mathrm{SE}(3)$ trajectory based on the tip trajectory.

To plan the tip's trajectory, we first identify an affordance point. This involves collecting the centers μ_i of valid part-aware Gaussian primitives $\mathcal{H}_i$. For revolute joints, we select a subset of centers that lie within the top 10th percentile, the farthest from the axis. For prismatic joints, we select the subset of centers closest to the robot base, within the 20th percentile along the axis dimension. From this subset, the center of the cluster is selected as the affordance point. Subsequently, the tip trajectory is generated by moving the affordance point from $\theta_c - \theta_o$, where θ_o is an offset designed to help avoid object-robot collisions, to θ_t along the screw axis $\mathcal{S}_j$.

After designing the tip trajectory, we then plan the gripper's $\mathrm{SE}(3)$ trajectory, ensuring that the gripper's tip follows the tip trajectory while maintaining a fixed orientation. Typically, we set the orientation so that the robot gripper faces toward the ground. Once the $\mathrm{SE}(3)$ trajectory is obtained, we solve the inverse kinematics to compute the final robot joint angle trajectory for articulated object manipulation.

B.4 Mixed-integer Optimization Formulation vs. ScrewSplat Formulation

In this section, we briefly compare the original mixed-integer optimization formulation for articulated object recognition with our relaxed formulation using ScrewSplat. As mentioned in the introduction, articulated object recognition is particularly challenging due to its mixed-integer optimization structure, which involves both continuous variables (e.g., 3D geometry and joint angles) and discrete, combinatorial variables (e.g., part segmentation labels, joint types, and joint counts). Formally, the general mixed-integer optimization formulation can be expressed as:

$$
\begin{aligned}
\min_{\{\mathcal{H}_i\},\{\mathcal{S}_j\},\{\theta_k\},n_s} \quad & \sum_k \mathcal{L}_{\text{render}}\big(\pi_{\text{GS}}(\theta_k;\{\mathcal{H}_i\},\{\mathcal{S}_j\}),I_{\text{gt}}\big) + \beta n_s \\
\text{subject to} \quad & \mathcal{H}_i = (T_i, s_i, \sigma_i, c_i, m_i), \quad m_i \in \{0,1,\dots,n_s\}, \quad i = 1,\dots,n_g, \\
& \mathcal{S}_j \in \mathbb{R}^6, \quad j = 1,\dots,n_s, \\
& \theta_k \in \mathbb{R}^{n_s}, \quad k = 1,\dots,n_a, \\
& n_s \in \mathbb{N},
\end{aligned}
$$

where

- $\mathcal{H}_i$ denotes a part-aware Gaussian primitive, where $(T_i, s_i, \sigma_i, c_i)$ are the parameters of the corresponding 3D Gaussian, and m_i is its segmentation label.
- $\mathcal{S}_j$ represents the screw axis of the j-th movable part in the articulated object.
- $\theta_k = (\theta_{k1},\dots,\theta_{kn_s}) \in \mathbb{R}^{n_s}$ is the joint angle vector for the k-th configuration of the articulated object.
- n_s is the number of screws, corresponding to the number of movable parts.
- π_{GS} is the standard rendering function used in Gaussian Splatting; during rendering, the i-th part-aware Gaussian primitive is transformed into a standard Gaussian with parameters $\big(e^{[\mathcal{S}_{m_i}]\theta_{km_i}} T_i, s_i, \sigma_i, c_i\big)$.

Several previous works simplify this problem by making certain assumptions. For example, some assume a known number of articulated components (i.e., n_s is no longer treated as an optimization variable), while others use intermediate procedures such as point correspondence matching or part clustering to obtain part segmentation labels a priori (i.e., m_i is no longer treated as an optimization variable).

Our goal is to address this optimization problem without relying on intermediate steps, auxiliary data, or prior knowledge of joint types or counts. Rather than solving the mixed-integer optimization problem directly, we reformulate it into a differentiable relaxed version (as described in Section 4), which enables effective optimization. Our formulation can be expressed as:

$$
\begin{aligned}
\min_{\{\mathcal{H}_i\},\{\mathcal{A}_j\},\{\theta_k\}} \quad & \sum_k \mathcal{L}_{\text{render}}\big(\pi_{\text{SS}}(\theta_k;\{\mathcal{H}_i\},\{\mathcal{A}_j\}),I_{\text{gt}}\big) + \beta \sum_j \sqrt{\gamma_j} \\
\text{subject to} \quad & \mathcal{H}_i = (T_i, s_i, \sigma_i, c_i, m_i), \quad m_i \in \Delta^{n_s}, \quad i = 1,\dots,n_g, \\
& \mathcal{A}_j = (\mathcal{S}_j, \gamma_j), \quad \mathcal{S}_j \in \mathbb{R}^6, \quad \gamma_j \in [0,1], \quad j = 1,\dots,n_s, \\
& \theta_k \in \mathbb{R}^{n_s}, \quad k = 1,\dots,n_a,
\end{aligned}
$$

where

- $\mathcal{H}_i$ denotes a part-aware Gaussian primitive, where $(T_i, s_i, \sigma_i, c_i)$ are the parameters of the corresponding 3D Gaussian, and m_i is a probability simplex over the movable parts.
- $\mathcal{A}_j$ represents the screw primitive of the j-th movable part, where $\mathcal{S}_j$ is the screw axis and γ_j denotes its confidence.
- $\theta_k = (\theta_{k1},\dots,\theta_{kn_s}) \in \mathbb{R}^{n_s}$ is the joint angle vector for the k-th configuration of the articulated object.
- π_{SS} is the rendering function of ScrewSplat, as described in Section 4.1 ("RGB Rendering Procedure with ScrewSplat").

C Experimental Details

C.1 Additional Details for Recognition Experiments

In this section, we provide further details on the evaluation dataset, baseline implementations, and evaluation metrics.

Dataset. We first select ten single-joint objects and three multi-joint objects from distinct categories in the PartNet-Mobility dataset [25]. For the camera parameters, we use the same intrinsic parameters and resolution as those of the Intel RealSense D435, and sample 48 camera positions uniformly distributed over a hemisphere of radius 1, centered on each articulated object. The camera orientations are set such that they face the center of the object. To ensure that the objects are fully visible, we rescale each object.

For each articulated object, we generate observation data for five different configurations (i.e., joint angles). For the single-joint objects, we use evenly spaced joint angles within the joint limits, while for the multi-joint objects, we randomly sample joint angle vectors. For each object and configuration, we render both RGB and depth images (which are used for optimizing PARIS* and DTA) from the 48 camera poses. We use Blender [26] to render the RGB and depth images.

Baseline Methods. For brief description of baselines including *PARIS* [16] and *DTA* [17], we refer to Appendix A.3. It is important to note that for *PARIS**, which also incorporates depth information, we use an additional depth rendering loss with depth supervision during the optimization step. These baseline methods currently accept input data for only two object configurations, so we use only the observations corresponding to two of the five configurations generated in our dataset. For the single-joint objects, we compare against PARIS, PARIS*, and DTA, and we select the two middle configurations from the evenly spaced set (specifically, the observation data from the 2nd and 4th configurations are used). For the multi-joint objects, we compare only against DTA, and since the five configurations are randomly sampled, we randomly select two configurations for use.

ScrewSpawn is an ablation model that follows the ScrewSplat framework but spawns only a single screw (with a known joint type). Specifically, in ScrewSpawn, only one screw primitive, $\mathcal{A}_1 = (\mathcal{S}_1, \gamma_1)$, is spawned, and the confidence value γ_1 is fixed at 1.0 (i.e., it is not treated as an optimization variable). Thus, optimization is performed only over the screw axis variable $\mathcal{S}_1$ among the variables of $\mathcal{A}_1$. Since there is only a single screw, the parsimony loss is also not applied. All other optimization details follow those of ScrewSplat. We use all observations corresponding to the five configurations generated in our dataset for optimization.

Evaluation Metrics. We again note that we adopt three types of metrics, including *geometry*, *motion*, and *appearance*, for articulated object recognition experiment. For *geometry* metric, we first reconstruct meshes from the recognized models. Specifically, for PARIS, PARIS*, and DTA, we use the marching cubes method [49]. For ScrewSpawn and ScrewSplat, we render depth images from multiple designated camera views and fuse them using the Truncated Signed Distance Function (TSDF). The final mesh is then extracted using the marching cubes method from the TSDF. From each reconstructed mesh, we uniformly sample 2,048 points to obtain a point cloud and compute the bi-directional Chamfer-l_2 distances as described in Section 5.1.

For *motion* metric, the calculation for single-joint objects follows the procedure described in Section 5.1. For multi-joint objects, given n ground-truth screw axes and m recognized screw axes, we first perform bipartite matching between the two sets based on angular error and axis position error. Then, for each of the n ground-truth axes, we compute the motion metrics accordingly.

For *appearance* metric, we use the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM), computed on rendered images under unseen joint angles. These unseen joint angles correspond to the midpoint values between the joint angles of the object configurations used during optimization. At these intermediate joint angles, we render RGB images from 48 camera views, following the same procedure described in the dataset section, and use them as ground-truth images. We then compute the PSNR and SSIM scores between the rendered outputs of the

recognized models and these ground-truth images. Since DTA does not render RGB images and focuses solely on estimating geometry and kinematic structure, we exclude it from the appearance metric evaluation.

Computational Aspect. Optimizing ScrewSplat over 30,000 iterations takes approximately 7-9 minutes on average, depending on the object, measured on a GeForce RTX 4090 GPU. On the same hardware, PARIS and DTA take about 3 and 15 minutes on average, respectively. ScrewSplat occupies approximately 30MB per object, with peak GPU memory usage of approximately 2.8GB, as measured by `torch.cuda.max_memory_allocated`.

C.2 Additional Details for Real-world Manipulation Experiments

We use the 7-DoF Franka Emika Panda robot equipped with a parallel-jaw gripper and an Intel RealSense D435 camera mounted on the gripper. We sample 24 camera positions uniformly distributed over a *partial* hemisphere of radius 0.85 (i.e., providing only a partial view), where the camera orientations are set to face the center of the workspace. Among these, we use 16 camera poses for which the robot has valid inverse kinematics (IK) solutions, as the camera is mounted on the robot arm.

Multi-view RGB observations are then collected under five object joint configurations, which are manually set by a human operator. The collected RGB images are processed into masked object images using the pretrained segmentation network SAM [50]. These masked images are used as input for recognition with ScrewSplat. Additionally, in the real-world experiment, we set the weight of the parsimony loss to 0.005. The same 16 camera poses are also used when estimating the current object state.

D Additional Experimental Results

D.1 Additional Results for Articulated Object Recognition Experiments

Single-joint Objects. Figure 11 presents additional examples of single-joint articulated object recognition. The overall trend is similar to that shown in Figure 5 of the main text. While PARIS and PARIS* successfully recognize the geometry and kinematic structure of some objects, they also exhibit several failure cases. DTA generally succeeds in recognizing all the additional objects. ScrewSpawn fails to recognize all objects except the scissor. ScrewSplat, on the other hand, successfully recognizes all objects and predicts more accurate and precise geometry, as well as more accurate kinematic structures, compared to DTA.

Table 3 reports the object-wise quantitative recognition results. We demonstrate that ScrewSplat generally achieves the highest overall performance across geometry, motion, and visual appearance. In particular, compared to the existing baselines, ScrewSplat consistently outperforms all others in predicting the geometry of the movable parts and joint axes. Although ScrewSpawn outperforms ScrewSplat for specific objects such as the scissor, we additionally observe that optimization often falls into local minima for most other objects.

Multi-joint Objects. Figure 10 presents an additional example of multi-joint articulated object recognition. In this object, DTA also achieves a reasonably accurate recognition. Along with Figure 6, ScrewSplat demonstrates superior performance over DTA by predicting more precise geometry and more accurate joint axes. Table 4 shows the object-wise quantitative results for multi-joint articulated object recognition. From the geometry perspective, while DTA slightly outperforms ScrewSplat in predicting the geometry of the static and whole parts, ScrewSplat significantly outperforms DTA in predicting the geometry of the movable parts. From the kinematic structure perspective, ScrewSplat generally shows better performance. In particular, for the 3-joint object, DTA completely fails by predicting an incorrect joint axis, whereas ScrewSplat successfully identifies the correct one.

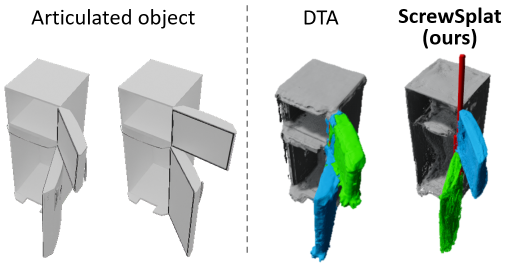

Figure 10: Additional results showing reconstructed meshes and screw axes for a multi-joint object using each method.

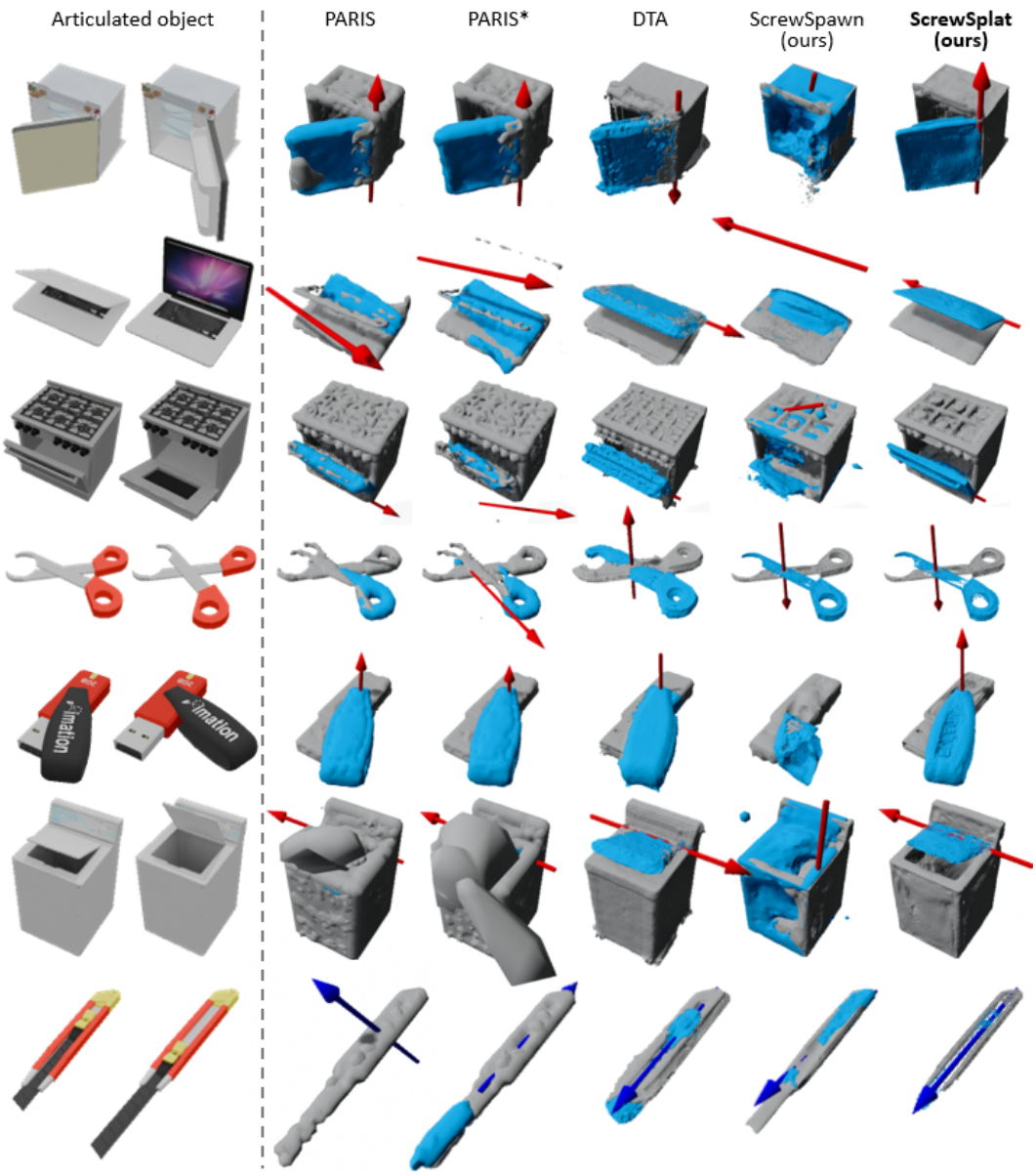

Figure 11: Additional results showing reconstructed meshes and screw axes for all single-joint objects used in the experiment (excluding the folding chair, storage box, and stapler shown in Figure 5), using each method. Static parts are shown in gray, movable parts in cyan, revolute joints in red, and prismatic joints in blue.

Table 3: Object-wise recognition performance for single-joint objects.

		Geometry (↓)			Motion (↓)		Appearance (↑)	
OBJECT	**METHOD**	CD-s	CD-m	CD-w	Ang.	Pos.	PSNR	SSIM
FoldChair	PARIS [16]	8.285	0.223	6.359	2.472	0.277	26.831	0.970
	PARIS* [16]	3.244	0.206	1.756	1.302	0.105	28.558	0.973
	DTA [17]	0.551	0.224	0.367	0.262	0.037	-	-
	ScrewSpawn	0.700	0.220	0.170	0.498	0.013	29.440	0.982
	ScrewSplat	**0.052**	**0.051**	**0.090**	**0.058**	**0.017**	**32.970**	**0.991**
Fridge	PARIS [16]	6.746	0.324	4.365	1.722	1.501	34.084	0.981
	PARIS* [16]	4.341	0.304	2.070	2.252	0.900	33.927	0.981
	DTA [17]	0.469	0.221	0.358	0.287	0.024	-	-
	ScrewSpawn	0.635	15.428	1.881	27.423	0.111	27.590	0.984
	ScrewSplat	**0.256**	**0.117**	**0.289**	**0.231**	**0.004**	**40.110**	**0.995**
Laptop	PARIS [16]	239.870	113.542	192.582	22.119	1.246	24.790	0.958
	PARIS* [16]	293.719	19.709	194.828	13.482	2.149	24.557	0.957
	DTA [17]	0.926	0.541	**0.341**	0.227	0.166	-	-
	ScrewSpawn	0.076	0.211	0.209	0.062	2.375	27.070	0.977
	ScrewSplat	**0.322**	**0.170**	0.347	**0.071**	**0.015**	**38.260**	**0.994**
Oven	PARIS [16]	15.400	9.209	9.332	6.547	4.285	29.170	0.952
	PARIS* [16]	21.212	11.606	16.419	31.119	2.760	27.305	0.947
	DTA [17]	**0.561**	0.242	**0.512**	0.251	0.097	-	-
	ScrewSpawn	0.834	23.083	1.223	33.489	1.451	26.650	0.968
	ScrewSplat	0.617	**0.204**	0.536	**0.125**	**0.007**	**35.010**	**0.983**
Scissor	PARIS [16]	3.630	0.675	0.198	19.904	0.994	29.972	0.975
	PARIS* [16]	7.625	1.438	2.195	59.327	0.896	28.656	0.971
	DTA [17]	0.337	0.299	0.339	0.136	0.029	-	-
	ScrewSpawn	**0.047**	**0.046**	**0.068**	**0.099**	**0.004**	**39.770**	**0.997**
	ScrewSplat	**0.047**	0.054	0.070	0.109	0.016	38.990	0.996
Stapler	PARIS [16]	151.146	6.510	85.308	4.426	0.064	21.053	0.954
	PARIS* [16]	99.810	20.018	61.348	5.205	2.306	21.198	0.954
	DTA [17]	0.320	1.167	0.181	0.222	2.058	-	-
	ScrewSpawn	0.139	3.256	0.320	1.195	2.253	21.940	0.975
	ScrewSplat	**0.127**	**0.685**	**0.126**	**0.054**	**0.005**	**36.850**	**0.995**
USB	PARIS [16]	0.200	0.236	0.215	0.688	3.502	28.068	0.973
	PARIS* [16]	0.207	0.228	0.200	0.989	0.048	26.999	0.970
	DTA [17]	0.586	0.369	0.288	0.172	0.023	-	-
	ScrewSpawn	0.404	5.137	1.543	3.159	0.168	22.220	0.969
	ScrewSplat	**0.236**	**0.105**	**0.234**	**0.047**	**0.001**	**35.300**	**0.993**
Washer	PARIS [16]	95.739	8.080	89.933	16.599	4.287	32.424	0.981
	PARIS* [16]	54.112	0.625	49.712	5.279	4.778	35.970	0.986
	DTA [17]	**0.435**	0.527	**0.447**	0.397	0.026	-	-
	ScrewSpawn	1.197	29.681	0.781	88.232	0.844	33.890	0.992
	ScrewSplat	0.714	**0.335**	0.449	**0.079**	**0.014**	**41.510**	**0.996**
Knife	PARIS [16]	7.438	F	4.554	61.590	-	30.554	0.988
	PARIS* [16]	3.611	32.524	3.663	1.879	-	30.161	0.988
	DTA [17]	0.355	0.410	0.359	0.047	-	-	-
	ScrewSpawn	1.088	28.176	2.439	5.993	-	31.230	0.994
	ScrewSplat	**0.039**	**0.038**	**0.046**	**0.031**		**41.090**	**0.998**
Storage	PARIS [16]	11.698	23.487	9.072	40.492	-	29.482	0.968
	PARIS* [16]	9.181	25.644	6.367	42.036	-	29.220	0.968
	DTA [17]	0.842	1.281	**0.407**	2.372	-	-	-
	ScrewSpawn	1.056	10.426	0.823	88.541	-	31.340	0.983
	ScrewSplat	**0.783**	**0.355**	0.421	**0.030**	-	**40.650**	**0.992**

Table 4: Object-wise recognition performance for multi-joint objects.

		Geometry (↓)					Motion (↓)						Appearance (↑)	
OBJECT	**METHOD**	CD-s	CD-m_1	CD-m_2	CD-m_3	CD-w	$Ang_1.$	$Pos_1.$	$Ang_2.$	$Pos_2.$	$Ang_3.$	$Pos_3.$	PSNR	SSIM
Fridge-2	DTA [17]	**0.402**	5.886	0.466	-	**0.424**	0.733	0.005	0.733	0.005	-	-	-	-
	ScrewSplat	0.517	**0.057**	**0.093**	-	0.531	**0.120**	**0.001**	**0.091**	**0.003**	-	-	36.65	0.990
Storage-2	DTA [17]	**0.524**	3.604	0.269	-	**0.500**	**0.114**	0.114	0.480	-	-	-	-	-
	ScrewSplat	1.033	**0.303**	**0.083**	-	1.009	0.136	**0.000**	**0.109**	-	-	-	37.20	0.988
Storage-3	DTA [17]	0.790	2.984	7.211	25.308	0.466	1.017	0.084	12.474	0.220	48.522	259.235	-	-
	ScrewSplat	**0.476**	**0.044**	**0.081**	**0.068**	**0.459**	**0.074**	**0.003**	**0.123**	**0.001**	**0.173**	**0.002**	36.43	0.983

Table 5: Recognition performance of ScrewSplat optimized using observations from two and five object configurations.

		Geometry (↓)			Motion (↓)		Appearance (↑)	
OBJECT	**METHOD**	CD-s	CD-m	CD-w	Ang.	Pos.	PSNR	SSIM
FoldChair	2-config	0.070	0.066	0.094	0.212	**0.006**	**34.39**	**0.993**
	5-config	**0.054**	**0.061**	**0.091**	**0.058**	0.017	33.44	0.992
Fridge	2-config	0.302	0.236	0.308	**0.153**	0.007	39.34	0.993
	5-config	**0.253**	**0.108**	**0.278**	0.231	**0.004**	**41.09**	**0.995**
Laptop	2-config	**0.069**	**0.081**	**0.097**	**0.066**	**0.003**	37.13	0.993
	5-config	0.322	0.127	0.347	0.071	0.015	**39.01**	**0.994**
Oven	2-config	**0.563**	0.474	**0.595**	**0.033**	0.013	**35.24**	**0.983**
	5-config	0.607	**0.274**	0.619	0.125	**0.007**	34.65	0.982
Scissor	2-config	1.915	**0.054**	**0.064**	**0.045**	**0.008**	36.66	0.995
	5-config	**0.047**	0.055	0.067	0.109	0.016	**39.46**	**0.997**
Stapler	2-config	0.154	6.199	1.195	F	1.519	21.46	0.971
	5-config	**0.122**	**0.577**	**0.127**	**0.054**	**0.005**	**36.54**	**0.995**
USB	2-config	0.276	0.151	0.262	0.084	0.104	28.08	0.979
	5-config	**0.237**	**0.106**	**0.225**	**0.047**	**0.001**	**36.97**	**0.994**
Washer	2-config	0.824	0.132	0.662	0.208	0.028	41.08	**0.996**
	5-config	**0.717**	**0.092**	**0.617**	**0.079**	**0.014**	**41.86**	**0.996**
Knife	2-config	0.044	0.070	0.048	0.045	-	41.14	**0.998**
	5-config	**0.039**	**0.038**	**0.047**	**0.031**	-	**41.23**	**0.998**
Storage	2-config	0.811	0.919	0.409	0.180	-	40.39	0.991
	5-config	**0.784**	**0.343**	**0.393**	**0.030**	-	**40.49**	**0.992**

D.2 Recognition Performance of ScrewSplat from Two-Configuration Observations

When optimizing ScrewSplat, we use RGB observations collected from five different configurations of the articulated object, whereas the other baselines only take observations from two configurations as input. In fact, to put it differently, the other baselines cannot handle observations from an arbitrary number of configurations, while ScrewSplat can. This makes ScrewSplat a more flexible algorithm capable of processing a richer set of information at once. To verify whether ScrewSplat still performs well when given limited input (i.e., observations from limited configurations), we conduct an ablation study using only two configurations as input.

Table 5 shows the recognition performance of ScrewSplat optimized using observations from two and five object configurations. Before analysis, we confirm that the performance was slightly sensitive to the weight of the parsimony loss when only using two configurations. Therefore, we optimize ScrewSplat over 10 different weights, ranging from 0.001 to 0.010, and evaluate the model with the best performance. From the table, it can be seen that there is little performance difference for most objects. In particular, for examples like the laptop and scissor, using only two configurations actually yields better performance in terms of geometry and motion metrics. However, even with the best parsimony loss weight set for two configurations, recognition failed for specific objects like the stapler and USB. In conclusion, ScrewSplat can recognize objects to some extent using observations from two configurations of articulated objects, and as discussed in the limitations and future directions, improvements in optimization techniques could lead to better performance in these cases.

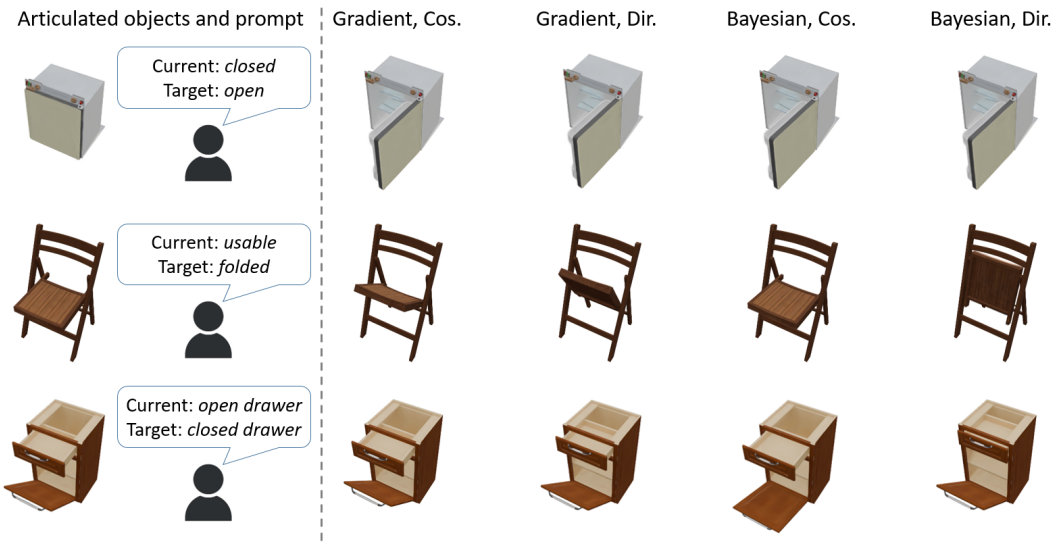

Figure 12: Comparison of directional CLIP loss versus cosine similarity loss, and Bayesian optimization versus gradient-based optimization for text-guided articulated object manipulation.

D.3 Why We Use Bayesian Optimization with Directional CLIP Loss

As described in Section 4.2, we optimize the target joint angles for text-guided articulated object manipulation using directional CLIP loss [24] through Bayesian optimization. However, there are more trivial alternatives. For instance, one could use a simple cosine similarity loss defined as follows [23]:

$$\mathcal{L}_{\text{CLIP-sim}} = 1 - e_I(\pi(\theta)) \cdot e_T(t_p), \tag{23}$$

where the notations are consistent with those used in Section 4.2. Furthermore, since our rendering function π is differentiable, this would allow the use of gradient-based optimization [22]. In this section, we compare directional CLIP loss versus cosine similarity loss, and Bayesian optimization versus gradient-based optimization. We provide a qualitative comparison of the performance for text-guided manipulation.

Figure 12 shows the visual appearance of articulated objects based on the optimized joint angles for each case. Overall, the refrigerator successfully reaches the target in all cases, while for other objects, we observe partial success or failure in cases where Bayesian optimization and directional CLIP loss are not used. In more detail, we can see that the directional CLIP loss leads to joint angles that align better with the target prompt than cosine similarity loss. This suggests that, for articulated objects, considering relative directions through a text description of the current state is more effective. Even when minimizing the directional CLIP loss, we observe partial success when using gradient-based methods. This implies that the loss landscape is complex and may lead to local minima, suggesting the need for a broader search space for joint angles. Currently, with joint angles having a maximum dimension of three, Bayesian optimization works efficiently. As the dimensionality increases, there is a need to design more appropriate optimization schemes.