

# Appendix for

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

Anonymous Author(s)

Affiliation

Address

email

### 1 A Extended Related Works

#### 2 A.1 Articulated Object Recognition Using Supervised Learning

3 There has been considerable interest in recognizing articulated objects within a supervised learning  
4 framework, where deep neural networks are trained to predict the underlying kinematic structure  
5 directly from raw visual input. Early studies often addressed this by employing category-driven ap-  
6 proaches, using category information to assist in identifying the kinematic structure, and testing on  
7 unseen objects belonging to known categories [1, 2, 3, 4]. More recently, to handle a broader range  
8 of objects beyond those in known categories, category-agnostic recognition approaches have been  
9 proposed. These methods primarily aim to reconstruct part-level geometry and kinematic structures,  
10 which closely align with our objective. For example, CARTO [5] predicts the geometry as a de-  
11 formable signed distance function (SDF) representations – similar to the category-specific A-SDF [2]  
12 but removes the reliance on object categories. Ditto [6] predicts articulable neural occupancy fields  
13 to generate digital twins of articulated objects from point cloud observations under two different ar-  
14 ticulation states. Articulate-Anything [7], a recent approach, integrates a vision-language foundation  
15 model to recognize part-aware geometries and kinematic structures, and incorporates interactable  
16 digital twins into simulators for sim-to-real robot learning. They demonstrate generalizability to  
17 unseen objects within similar categories, and even to unseen categories through foundation models.  
18 However, they inherently struggle to generalize to objects that differ significantly from the training  
19 categories.

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

21 Several works have attempted to recognize articulated objects by directly fitting 3D representations  
22 and kinematic structures to observations without any supervision. Since these methods typically  
23 perform optimization for each object individually, they are often referred to as per-object optimiza-  
24 tion methods. A representative early work in per-object optimization methods is PARIS [8], which  
25 presents a method based on neural radiance fields (NeRF). Specifically, PARIS defines separate ra-  
26 diance fields for movable and static parts, performing joint rendering and optimization to achieve  
27 part-level reconstruction and articulation discovery. However, this approach is only applicable to  
28 single-joint objects, i.e., articulated objects with only one movable part, with pre-determined joint  
29 types.

30 Recently, approaches capable of covering multi-joint objects have been proposed. A notable ex-  
31 ample is DTA [9], which first reconstructs two entire meshes using RGB-D data from observations  
32 under two configurations of articulated objects, and then determines the kinematic structure using  
33 a feature correspondence matching module. During the mesh reconstruction process, depth images  
34 are used, and in the feature correspondence matching step, the number of movable parts must also  
35 be known. DTA successfully infers the kinematic structure of articulated objects with multiple mov-  
36 able parts using this additional information. Subsequently, research utilizing Gaussian splatting [10],

37 such as ArtGS [11] and ArticulatedGS [12], has emerged as an alternative to neural radiance fields.  
38 These approaches are somewhat similar to ours in that they leverage Gaussian splatting. ArtGS, like  
39 DTA, utilizes point correspondence matching and similarly requires depth images and knowledge  
40 of the number of movable parts. ArticulatedGS, however, does not rely on these assumptions but is  
41 still limited to discovering only one articulation information per optimization step.

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

### 51 **A.3 Articulation Discovery via Robot-Object Interaction**

52 Beyond observation-based recognition approaches, several works have explored active interaction  
53 strategies that allow a robot to interact with unknown articulated objects and collect additional ob-  
54 servations for articulation reasoning [13, 14, 15]. These methods often rely on pre-trained networks  
55 to guide interaction, rather than some learning-free strategies that cannot extract joint parameters  
56 from static observations. For instance, [13] uses an RGB image as input to predict hold and push  
57 locations via a trained network. The predicted actions are then executed by a human to modify the  
58 articulated object’s configuration, and subsequent observations are used to infer the kinematic struc-  
59 ture. Similar methods such as [14] and [15] also leverage 3D point cloud inputs to predict interaction  
60 positions and directions. Using point cloud observation data as 3D geometric information enables  
61 the robot to interact with the object and actively acquire additional observations.

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

## 69 B Implementation Details for ScrewSplatting

### 70 B.1 Details for Screw Theory

71 In this section, we describe the closed-form matrix exponential expressions for both revolute and  
 72 prismatic screw axes [16]. For convenience, we briefly review the screw theory outlined in Section  
 73 3.1. A six-dimensional screw axis  $\mathcal{S}$  is given by:

$$\mathcal{S} = \begin{bmatrix} \omega \\ v \end{bmatrix} \in \mathbb{R}^6. \quad (1)$$

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

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

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

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

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

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

$$e^{[\mathcal{S}]\theta} = I + [\mathcal{S}]\theta + [\mathcal{S}]^2 \frac{\theta^2}{2!} + [\mathcal{S}]^3 \frac{\theta^3}{3!} + \dots \quad (4)$$

$$= \begin{bmatrix} e^{[\omega]\theta} & G(\theta)v \\ 0 & 1 \end{bmatrix}, \quad (5)$$

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

$$G(\theta) = I\theta + [\omega] \frac{\theta^2}{2!} + [\omega]^2 \frac{\theta^3}{3!} + \dots \quad (6)$$

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

$$G(\theta) = I\theta + (1 - \cos \theta)[\omega] + (\theta - \sin \theta)[\omega]^2. \quad (7)$$

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

$$e^{[\mathcal{S}]\theta} = \begin{bmatrix} e^{[\omega]\theta} & (I\theta + (1 - \cos \theta)[\omega] + (\theta - \sin \theta)[\omega]^2) v \\ 0 & 1 \end{bmatrix}, \quad (8)$$

86 where

$$e^{[\omega]\theta} = I + \sin \theta [\omega] + (1 - \cos \theta) [\omega]^2. \quad (9)$$

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

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

$$e^{[\mathcal{S}]\theta} = \begin{bmatrix} I & v\theta \\ 0 & 1 \end{bmatrix}. \quad (10)$$

## 90 B.2 Implementation Details for ScrewSplat

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

94 **Initialization of Core Components.** For the part-aware Gaussian primitives, we initially spawn  
 95 10,000 primitives. For each primitive  $\mathcal{H}_i = (T_i, s_i, \sigma_i, c_i, m_i)$ , we generally adopt the same initial-  
 96 ization scheme as used in the original Gaussian Splatting method for the variables  $(T_i, s_i, \sigma_i, c_i)$ . It  
 97 is worth noting that the position  $\mu_i$  of the Gaussian primitive from the pose  $T_i = [R_i, \mu_i]$  is sam-  
 98 pled from a uniform distribution: in the simulation environment, it is sampled from  $[-1, 1]^3$ , and  
 99 in real-world experiments, it is sampled from  $[-0.7, 0.7]^3$ . We found that initializing the positions  
 100 within the camera’s range results in better performance. The part probability  $m_i \in \Delta^{n_s}$  is initialized  
 101 as a uniform distribution (i.e., an element-wise constant vector).

102 For the screw primitives, we initially spawn eight revolute joint axes and eight prismatic joint axes  
 103 (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-  
 104 dimensional real vector from a uniform distribution over  $[-0.5, 0.5]^3$  and then normalize it to satisfy  
 105 the constraints for both revolute and prismatic joints. Specifically, let the sampled six-dimensional  
 106 vector be  $[x, q]$ , where  $x \in \mathbb{R}^3$  and  $q \in \mathbb{R}^3$ . For a revolute joint  $[\omega, v]$ ,  $\omega$  is set to  $x/\|x\|$  and  $v$  is set  
 107 to  $-\omega \times q$ . For a prismatic joint  $[\omega, v]$ ,  $\omega$  is set to the zero vector, and  $v$  is set to  $q/\|q\|$ . The screw  
 108 confidence  $\gamma_j$  is initialized to 0.9.

109 For the joint angle vectors, we generate as many joint angle vector variables as there are configura-  
 110 tions used in the observations. The joint angle  $\theta_k$  is initialized to the zero vector.

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

$$T_i \leftarrow e^{[S_{j^*}] \theta_{m_{j^*}}} T_i. \quad (11)$$

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

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

139 above:

$$T_i \leftarrow e^{[S_j]\theta} T_i. \quad (12)$$

140 **Learning Rates and Hyperparameters.** For the variables  $(T_i, s_i, \sigma_i, c_i)$ , used in Gaussian Splat-  
 141 ting, we use the same parameters as those previously employed. For the part probability  $m_i$ , a  
 142 learning rate of 0.1 is applied before passing through the softmax. For the screw axis  $S_j$ , a learning  
 143 rate of 0.003 is used before normalization. The screw confidence  $\gamma_j$  uses a learning rate of 0.01  
 144 before passing through the sigmoid. The joint angle  $\theta_k$  uses a learning rate of 0.01.

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

### 148 B.3 Details for Articulated Object Manipulation

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

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

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

$$\mathcal{L}_{estimate} = \mathcal{L}_{render}. \quad (13)$$

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

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

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

180 After designing the tip trajectory, we then plan the gripper’s SE(3) trajectory, ensuring that the  
 181 gripper’s tip follows the tip trajectory while maintaining a fixed orientation. Typically, we set the  
 182 orientation so that the robot gripper faces toward the ground. Once the SE(3) trajectory is obtained,  
 183 we solve the inverse kinematics to compute the final robot joint angle trajectory for articulated object  
 184 manipulation.

## 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 [17]. 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 [18] to render the RGB and depth images.

**Baseline Methods.** For brief description of baselines including PARIS [8] and DTA [9], 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  $\gamma_1$  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 [19]. 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

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

## 237 **C.2 Additional Details for Real-world Manipulation Experiments**

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

244 Multi-view RGB observations are then collected under five object joint configurations, which are  
245 manually set by a human operator. The collected RGB images are processed into masked object  
246 images using the pretrained segmentation network SAM [20]. These masked images are used as  
247 input for recognition with ScrewSplat. Additionally, in the real-world experiment, we set the weight  
248 of the parsimony loss to 0.005. The same 16 camera poses are also used when estimating the current  
249 object state.

## D Additional Experimental Results

### D.1 Additional Results for Articulated Object Recognition Experiments

**Single-joint Objects.** Figure 2 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 1 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 1 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 2 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.

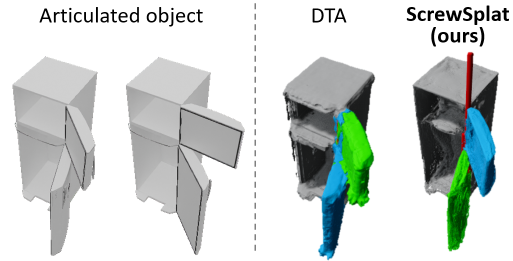


Figure 1: Additional results showing reconstructed meshes and screw axes for a multi-joint object using each method.



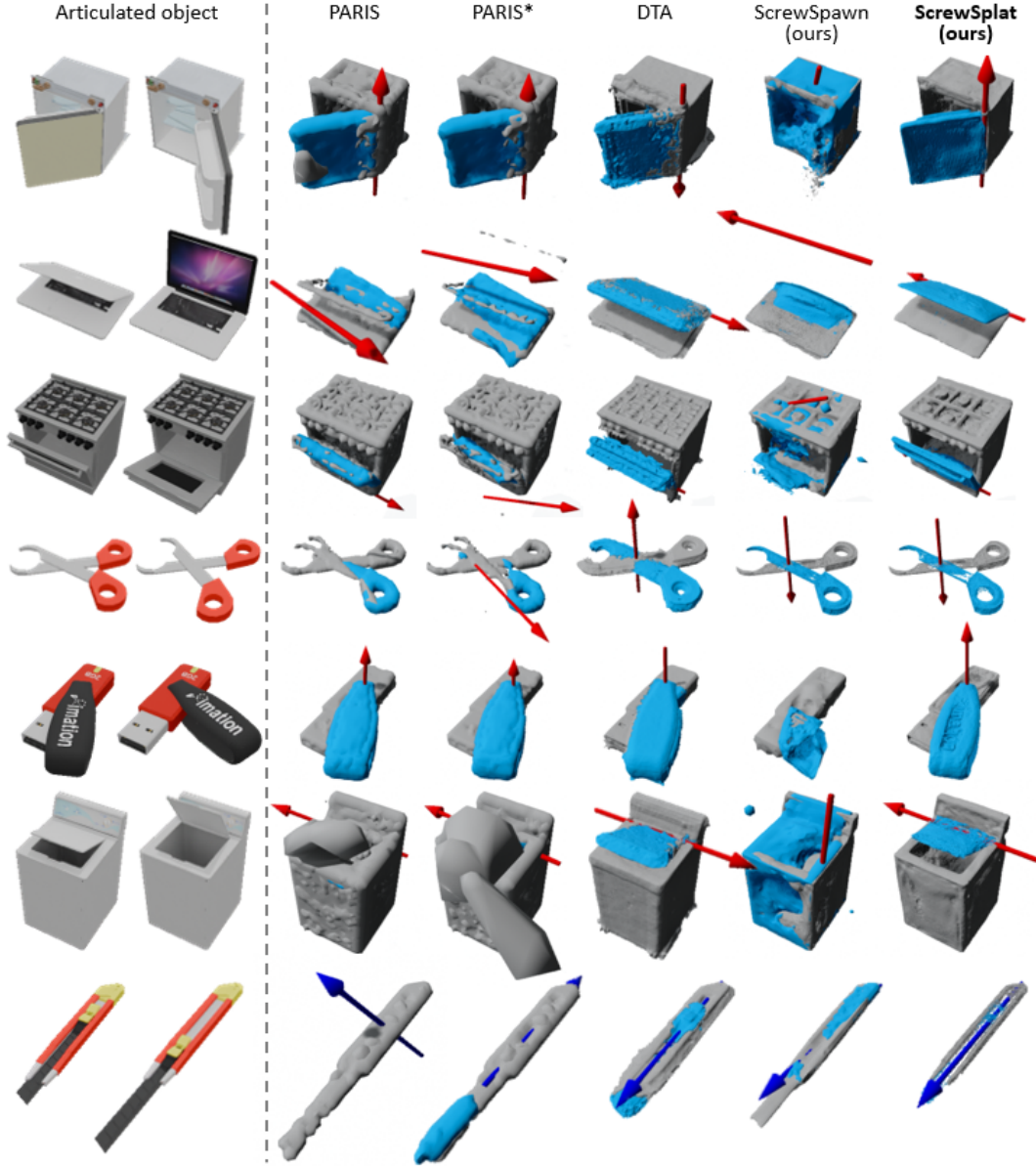


Figure 2: Additional results showing reconstructed meshes and screw axes for all single-joint objects used in the experiment (excluding the folding chair, storage box, and stapler shown in Figure 5), using each method. Static parts are shown in gray, movable parts in cyan, revolute joints in red, and prismatic joints in blue.

Table 1: Object-wise recognition performance for single-joint objects.

OBJECT	METHOD	Geometry ( $\downarrow$ )			Motion ( $\downarrow$ )		Appearance ( $\uparrow$ )	
		CD-s	CD-m	CD-w	Ang.	Pos.	PSNR	SSIM
FoldChair	PARIS [8]	8.285	0.223	6.359	2.472	0.277	26.831	0.970
	PARIS* [8]	3.244	0.206	1.756	1.302	0.105	28.558	0.973
	DTA [9]	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	<b>0.052</b>	<b>0.051</b>	<b>0.090</b>	<b>0.058</b>	<b>0.017</b>	<b>32.970</b>	<b>0.991</b>
Fridge	PARIS [8]	6.746	0.324	4.365	1.722	1.501	34.084	0.981
	PARIS* [8]	4.341	0.304	2.070	2.252	0.900	33.927	0.981
	DTA [9]	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	<b>0.256</b>	<b>0.117</b>	<b>0.289</b>	<b>0.231</b>	<b>0.004</b>	<b>40.110</b>	<b>0.995</b>
Laptop	PARIS [8]	239.870	113.542	192.582	22.119	1.246	24.790	0.958
	PARIS* [8]	293.719	19.709	194.828	13.482	2.149	24.557	0.957
	DTA [9]	0.926	0.541	<b>0.341</b>	0.227	0.166	-	-
	ScrewSpawn	0.076	0.211	0.209	0.062	2.375	27.070	0.977
	ScrewSplat	<b>0.322</b>	<b>0.170</b>	0.347	<b>0.071</b>	<b>0.015</b>	<b>38.260</b>	<b>0.994</b>
Oven	PARIS [8]	15.400	9.209	9.332	6.547	4.285	29.170	0.952
	PARIS* [8]	21.212	11.606	16.419	31.119	2.760	27.305	0.947
	DTA [9]	<b>0.561</b>	0.242	<b>0.512</b>	0.251	0.097	-	-
	ScrewSpawn	0.834	23.083	1.223	33.489	1.451	26.650	0.968
	ScrewSplat	0.617	<b>0.204</b>	0.536	<b>0.125</b>	<b>0.007</b>	<b>35.010</b>	<b>0.983</b>
Scissor	PARIS [8]	3.630	0.675	0.198	19.904	0.994	29.972	0.975
	PARIS* [8]	7.625	1.438	2.195	59.327	0.896	28.656	0.971
	DTA [9]	0.337	0.299	0.339	0.136	0.029	-	-
	ScrewSpawn	<b>0.047</b>	<b>0.046</b>	<b>0.068</b>	<b>0.099</b>	<b>0.004</b>	<b>39.770</b>	<b>0.997</b>
	ScrewSplat	<b>0.047</b>	0.054	0.070	0.109	0.016	38.990	0.996
Stapler	PARIS [8]	151.146	6.510	85.308	4.426	0.064	21.053	0.954
	PARIS* [8]	99.810	20.018	61.348	5.205	2.306	21.198	0.954
	DTA [9]	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	<b>0.127</b>	<b>0.685</b>	<b>0.126</b>	<b>0.054</b>	<b>0.005</b>	<b>36.850</b>	<b>0.995</b>
USB	PARIS [8]	0.200	0.236	0.215	0.688	3.502	28.068	0.973
	PARIS* [8]	0.207	0.228	0.200	0.989	0.048	26.999	0.970
	DTA [9]	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	<b>0.236</b>	<b>0.105</b>	<b>0.234</b>	<b>0.047</b>	<b>0.001</b>	<b>35.300</b>	<b>0.993</b>
Washer	PARIS [8]	95.739	8.080	89.933	16.599	4.287	32.424	0.981
	PARIS* [8]	54.112	0.625	49.712	5.279	4.778	35.970	0.986
	DTA [9]	<b>0.435</b>	0.527	<b>0.447</b>	0.397	0.026	-	-
	ScrewSpawn	1.197	29.681	0.781	88.232	0.844	33.890	0.992
	ScrewSplat	0.714	<b>0.335</b>	0.449	<b>0.079</b>	<b>0.014</b>	<b>41.510</b>	<b>0.996</b>
Knife	PARIS [8]	7.438	F	4.554	61.590	-	30.554	0.988
	PARIS* [8]	3.611	32.524	3.663	1.879	-	30.161	0.988
	DTA [9]	0.355	0.410	0.359	0.047	-	-	-
	ScrewSpawn	1.088	28.176	2.439	5.993	-	31.230	0.994
	ScrewSplat	<b>0.039</b>	<b>0.038</b>	<b>0.046</b>	<b>0.031</b>	-	<b>41.090</b>	<b>0.998</b>
Storage	PARIS [8]	11.698	23.487	9.072	40.492	-	29.482	0.968
	PARIS* [8]	9.181	25.644	6.367	42.036	-	29.220	0.968
	DTA [9]	0.842	1.281	<b>0.407</b>	2.372	-	-	-
	ScrewSpawn	1.056	10.426	0.823	88.541	-	31.340	0.983
	ScrewSplat	<b>0.783</b>	<b>0.355</b>	0.421	<b>0.030</b>	-	<b>40.650</b>	<b>0.992</b>

Table 2: Object-wise recognition performance for multi-joint objects.

OBJECT	METHOD	Geometry ( $\downarrow$ )					Motion ( $\downarrow$ )					Appearance ( $\uparrow$ )		
		CD-s	CD-m <sub>1</sub>	CD-m <sub>2</sub>	CD-m <sub>3</sub>	CD-w	Ang <sub>1</sub> .	Pos <sub>1</sub> .	Ang <sub>2</sub> .	Pos <sub>2</sub> .	Ang <sub>3</sub> .	Pos <sub>3</sub> .	PSNR	SSIM
Fridge-2	DTA [9]	<b>0.402</b>	5.886	0.466	-	<b>0.424</b>	0.733	0.005	0.733	0.005	-	-	-	-
	ScrewSplat	0.517	<b>0.057</b>	<b>0.093</b>	-	0.531	<b>0.120</b>	<b>0.001</b>	<b>0.091</b>	<b>0.003</b>	-	-	36.65	0.990
Storage-2	DTA [9]	<b>0.524</b>	3.604	0.269	-	<b>0.500</b>	<b>0.114</b>	0.114	0.480	-	-	-	-	-
	ScrewSplat	1.033	<b>0.303</b>	<b>0.083</b>	-	1.009	0.136	<b>0.000</b>	<b>0.109</b>	-	-	-	37.20	0.988
Storage-3	DTA [9]	0.790	2.984	7.211	25.308	0.466	1.017	0.084	12.474	0.220	48.522	259.235	-	-
	ScrewSplat	<b>0.476</b>	<b>0.044</b>	<b>0.081</b>	<b>0.068</b>	<b>0.459</b>	<b>0.074</b>	<b>0.003</b>	<b>0.123</b>	<b>0.001</b>	<b>0.173</b>	<b>0.002</b>	36.43	0.983

Table 3: Recognition performance of ScrewSplat optimized using observations from two and five object configurations.

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

## 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 3 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.

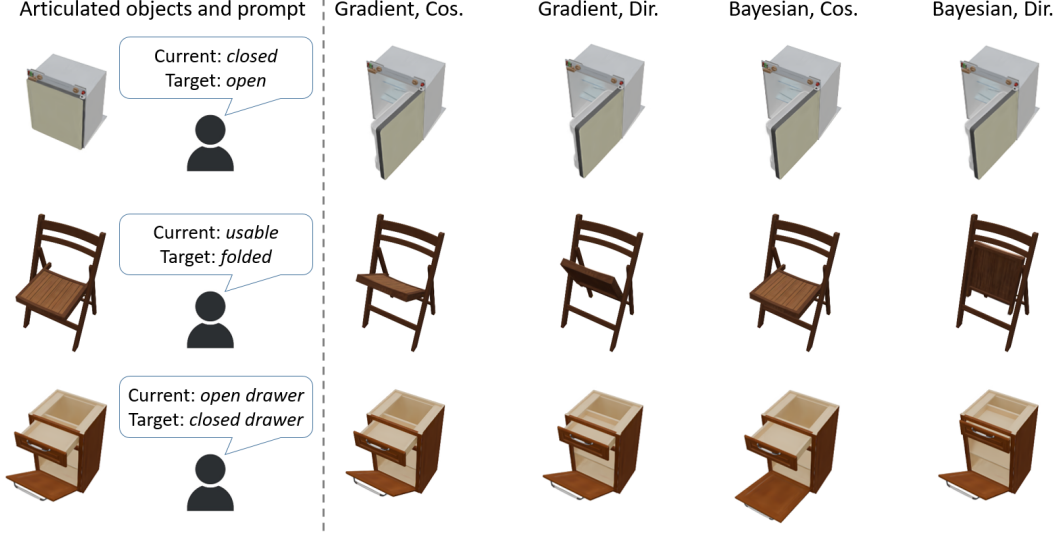


Figure 3: Comparison of directional CLIP loss versus cosine similarity loss, and Bayesian optimization versus gradient-based optimization for text-guided articulated object manipulation.

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

301 As described in Section 4.2, we optimize the target joint angles for text-guided articulated object  
 302 manipulation using directional CLIP loss [21] through Bayesian optimization. However, there are  
 303 more trivial alternatives. For instance, one could use a simple cosine similarity loss defined as  
 304 follows [22]:

$$\mathcal{L}_{\text{CLIP-sim}} = 1 - e_I(\pi(\theta)) \cdot e_T(t_p), \quad (14)$$

305 where the notations are consistent with those used in Section 4.2. Furthermore, since our rendering  
 306 function  $\pi$  is differentiable, this would allow the use of gradient-based optimization [23]. In this  
 307 section, we compare directional CLIP loss versus cosine similarity loss, and Bayesian optimization  
 308 versus gradient-based optimization. We provide a qualitative comparison of the performance for  
 309 text-guided manipulation.

310 Figure 3 shows the visual appearance of articulated objects based on the optimized joint angles  
 311 for each case. Overall, the refrigerator successfully reaches the target in all cases, while for other  
 312 objects, we observe partial success or failure in cases where Bayesian optimization and directional  
 313 CLIP loss are not used. In more detail, we can see that the directional CLIP loss leads to joint  
 314 angles that align better with the target prompt than cosine similarity loss. This suggests that, for  
 315 articulated objects, considering relative directions through a text description of the current state is  
 316 more effective. Even when minimizing the directional CLIP loss, we observe partial success when  
 317 using gradient-based methods. This implies that the loss landscape is complex and may lead to  
 318 local minima, suggesting the need for a broader search space for joint angles. Currently, with joint  
 319 angles having a maximum dimension of three, Bayesian optimization works efficiently. As the  
 320 dimensionality increases, there is a need to design more appropriate optimization schemes.

## 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] 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.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] 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.

- 366 [16] K. M. Lynch and F. C. Park. *Modern robotics*. Cambridge University Press, 2017.
- 367 [17] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, et al.  
 368 Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF*  
 369 *conference on computer vision and pattern recognition*, pages 11097–11107, 2020.
- 370 [18] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation,  
 371 Stichting Blender Foundation, Amsterdam, 2018. URL <http://www.blender.org>.
- 372 [19] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction  
 373 algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.
- 374 [20] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead,  
 375 A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.
- 376 [21] R. Gal, O. Patashnik, H. Maron, A. H. Bermano, G. Chechik, and D. Cohen-Or. Stylegan-nada:  
 377 Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)*,  
 378 41(4):1–13, 2022.
- 379 [22] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell,  
 380 P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervi-  
 381 sion. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.
- 382 [23] R. Liu, A. Canberk, S. Song, and C. Vondrick. Differentiable robot rendering. *arXiv preprint*  
 383 *arXiv:2410.13851*, 2024.