Ditto: Building Digital Twins of Articulated Objects from Interaction

Zhenyu Jiang Cheng-Chun Hsu Yuke Zhu

Department of Computer Science, The University of Texas at Austin

{zhenyu,hsucc,yukez}@cs.utexas.edu

Abstract

Digitizing physical objects into the virtual world has the potential to unlock new research and applications in embodied AI and mixed reality. This work focuses on recreating interactive digital twins of real-world articulated objects, which can be directly imported into virtual environments. We introduce Ditto to learn articulation model estimation and 3D geometry reconstruction of an articulated object through interactive perception. Given a pair of visual observations of an articulated object before and after interaction, Ditto reconstructs part-level geometry and estimates the articulation model of the object. We employ implicit neural representations for joint geometry and articulation modeling. Our experiments show that Ditto effectively builds digital twins of articulated objects in a category-agnostic way. We also apply Ditto to real-world objects and deploy the recreated digital twins in physical simulation. Code and additional results are available at https://ut-austin-rpl.github.io/Ditto/

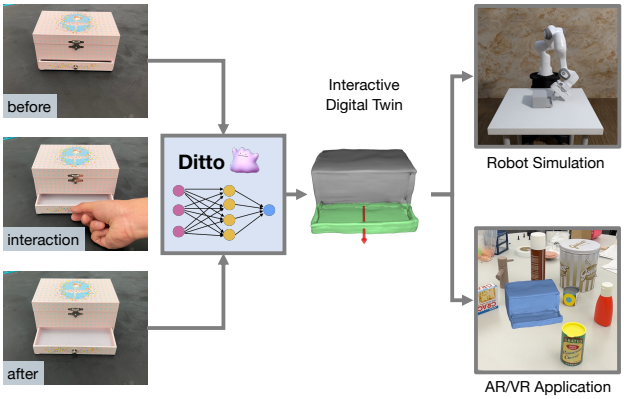

Figure 1. We build digital twins of articulated objects through interactive perception. Given visual observations before and after interaction, our method jointly reconstructs the part-level geometry and articulation model of the object. Our recreated digital twins can be spawned in physics engines and are fully interactive in robot simulation and AR/VR applications.

1. Introduction

Synthetic data has played a steadily more vital role in fueling emerging AI applications, from training and prototyping computer vision models [21, 45] to teaching robots to perform physical tasks [2, 30, 61]. As modern AI models become larger and increasingly data-hungry, virtual platforms and synthetic datasets supply a massive amount of cheap training data. For vision models to benefit from synthetic data, realism is key — the distribution mismatch between the real and virtual worlds hinders the generalization of models trained in simulation. A promising path towards closing the reality gap is digitizing physical objects and recreating them in virtual environments. Research on 3D vision and SLAM [3, 13, 36, 37, 56] has made significant advances in capturing realistic objects and scenes with static 3D models. Nonetheless, the burgeoning body of embodied AI and mixed reality research calls for *interactive digital twins* of physical objects that can be spawned in simulated environments and interact with virtual agents. Building digital twins of *articulated objects* is particularly challenging as it requires not only a good understanding of its overall geometry but the part compositions as well as the kinematic relations between the parts.

Recent efforts in embodied AI platforms [24, 25, 53] have incorporated interactive articulated objects, such as cabinets and drawers, in simulated household environments and employed them for training virtual agents. Even so, they heavily rely on graphics designers and engineers to author and curate the object models, limiting the scalability of the asset acquisition process. Developing vision-based methods to automate the estimation [1, 18] and reconstruction [35] of articulated objects has been an active line of research, accelerated by new tools developed from the 3D vision community, including geometric deep learning [20, 39, 42] and implicit neural representations [10, 38]. The majority of prior work focuses on solving individual components of the problem rather than constructing a full-fledged model. Several recent works [26, 55] have studied the joint learning of

part segmentation and joint estimation. However, they infer part-level geometry on the point cloud which cannot be used for physical simulation, because physical simulation requires compact geometry of the object such as the mesh for collision computation.

Departing from prior work, we seek the full-fledged virtual recreation of articulated physical objects from unknown categories. These digital twins of articulated objects represent the geometry and physics of individual object parts and their articulation relations (*e.g.*, prismatic or revolute joints). Category-agnostic articulation estimation from a single image is inherently ambiguous. The parts may move along a prismatic axis or rotate around a revolute axis depending on the underlying kinematic joints. Following pioneer work on the interactive perception of articulated objects [15, 31], we propose to infer the digital twins from visual observations collected before and after articulated motions (see Fig. 1). This task comprises three intimately connected challenges: object part segmentation based on motion cues, part reconstruction from a partial point cloud, and articulation estimation of unknown joint types.

We introduce **Ditto** (**Di**gital **t**win of ar**t**iculated **o**bjects), an implicit neural representation-based model that jointly predicts part-level geometry and kinematic articulation between the parts. We employ implicit neural representations [6,33,40,48] to encode continuous and high-resolution 3D information. Ditto is built on top of ConvONets [43], which learns local implicit fields based on convolutional feature grids. The input to Ditto is partial point cloud observations of an articulated object before and after interaction with one of its parts. The key technical challenge is to establish correspondences between these two partial observations. To achieve this, we encode the point clouds with PointNet++ [44] into two sets of subsampled point features. Then we fuse these two sets of point features with a self-attention layer [54] and decode the fused subsampled features into dense point features. We construct structured feature grids from the decoded point features. A local feature can be computed from the feature grids at a query 3D coordinate. We learn an implicit occupancy decoder and an implicit segmentation decoder that maps from a 3D coordinate and its local feature to the occupancy/part segmentation label at that coordinate to reconstruct part-level geometry. We use another set of implicit decoders that densely predict the relative joint parameters at each query point to estimate articulated joints. Such dense articulation prediction brings forth more robust articulation estimation than predicting the joint parameters globally. All the implicit decoders can be trained end-to-end.

We evaluate our approach on two datasets of articulated objects [1,55]. Our results demonstrate that Ditto accurately reconstructs part-level geometry and articulation model in a category-agnostic fashion. Ditto achieves superior results across all datasets and metrics compared with the baselines. Furthermore, we apply our method to real-world articulated objects for recreating digital twins. We provide examples of instantiating digital twins in simulation for a virtual robot to interact and transfer the interaction back to the real world.

2. Related Work

Articulation Model Estimation. Probabilistic methods [8, 49–52] are first used to estimate the articulation relationships between different parts of an articulated object. As articulation can be ambiguous to infer from a single observation, interactive perception methods [4, 12, 15, 22, 31, 32, 60] have been employed to estimate articulation from action-generated visual stimuli. Conventional methods take a series of sensory observations as input and rely on markers or handcrafted features to track the mobile parts. Recently, deep learning methods have been developed for articulation estimation from raw sensory data [1, 17, 18, 28]. Most of these works primarily focus on predicting the articulation parameters. In contrast, our method jointly reconstructs the full 3D geometry and estimates the articulation model.

3D Reconstruction of Articulated Objects. 3D models of articulated objects encode articulation and geometry properties of the objects. Pioneer work from Huang *et al*. [16] uses structure from motion to reconstruct the full point cloud of the object and segment the point cloud using feature-based correspondence. More recently, learning-based methods [26, 55] are developed to predict part segmentation together with joint parameters. These works reason about the part-level geometry on the point cloud, which lacks the mesh information required for physical simulation. A series of methods on reconstructing deformable object [5, 58, 59] use articulated bones to represent articulation. These representations loosely constrain the motions of object parts. In contrast, the digital twins require accurate part-level geometry and precise articulation modeling to be simulated in a physics engine.

Closest to our work is A-SDF [35], which learns a deep signed function for articulated objects from which a 3D mesh can be extracted. It uses a separate latent code to model the articulation state implicitly. Instead, our method builds full 3D meshes for each part and models their articulations explicitly. The resultant digital twin can spawn in virtual environments for physical interaction.

Implicit Neural Representations. Our method builds on top of recent work on implicit neural representations [6, 33, 40]. These works encode a 3D shape with the iso-surface of an implicit function. These implicit models are parametrized with deep networks so that they are capable of representing complex shapes smoothly and continuously in high resolution. Aiming at better scalability and finer details, several approaches [14, 27, 43] learn local implicit

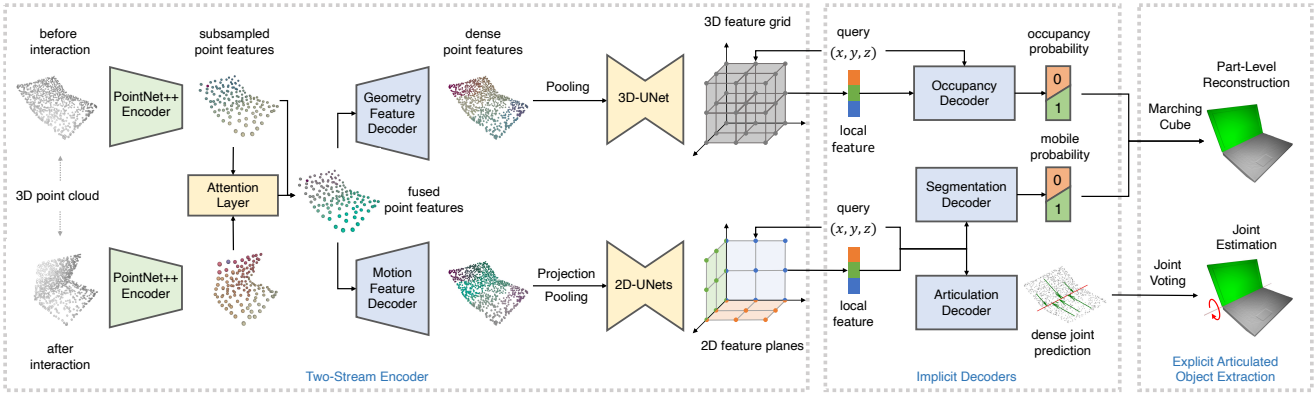

Figure 2. Model architecture of Ditto. The input consists of point cloud observations before and after interaction. After a PointNet++ [44] encoder, we fuse the subsampled point features with a simple attention layer. Then we use two independent decoders to propagate the fused point features into two sets of dense point features for geometry reconstruction and articulation estimation separately. We construct feature grid/planes by projecting and pooling the point features and query local features from the constructed feature grid/planes. Conditioning on local features, we use different decoders to predict occupancy, segmentation, and joint parameters with respect to the query points.

decoders and condition the implicit representations on local features instead of a global shape feature. Specifically, our model extends ConvONets [43] with a stronger encoder and a fusing module for processing two input data streams.

Physical Simulation with Articulated Objects. Physical simulators have become a vital tool for embodied AI research. A growing trend is shifting from static 3D scenes for visual navigation [24,47,61] to interactive environments that support physical interaction between the robot and the objects [9,25,53]. Interactive 3D assets are the key elements to construct these simulators. Existing interactive 3D assets are mostly authored and refined by 3D artists [34,53,55,57] or procedurally generated [1]. Our method builds interactive digital twins of daily articulated objects directly from visual observations. It has the potential to accelerate the acquisition of realistic interactive 3D assets.

3. Problem Formulation

We study the problem of recreating interactive digital twins of articulated objects from a pair of sensory observations before and after an interaction. Digital twins are commonly represented in standard 3D formats, such as URDF[1], such that they can be imported into physics engines. To enable physical interaction in the virtual world, a digital twin of an articulated object constitutes a *kinematic tree*, where the nodes define the geometry and physical properties (*e.g.* mass and friction) of individual parts and the edges define the kinematic joints between the parts. This work focuses on estimating part geometry and kinematic articulation while setting the physical properties to default values based on real-world statistics.

Given an articulated object from an unknown category, we interact with the object to change the articulation state. Without the loss of generality, we assume only one part is moved after the interaction, which we call the mobile part. The input to our method is a pair of point cloud observations $\mathcal{P}_1, \mathcal{P}_2 \in \mathbb{R}^{N\times 3}$ of the articulated object before and after an interaction. N is the number of input points. The objective is to segment and reconstruct the 3D geometry for static and mobile parts, estimate the joint parameters that connect these two parts, and relative change of the joint states.

For articulation estimation, we consider the 1D revolute joints and 1D prismatic joints. We follow Li *et al*. [26] and parameterize the two types of joints as follows. The parameters of a prismatic joint consist of the direction of the translation axis $\mathbf{u}^p \in \mathbb{R}^3$ and the joint state c^p. The joint state c^p is defined as the relative translation distance between the two observations. The parameters of a revolute joint consist of the direction of the revolute axis $\mathbf{u}^r \in \mathbb{R}^3$, a pivot point $\mathbf{q} \in \mathbb{R}^3$ on the revolute axis and the joint state c^r. The joint state c^r is defined as the relative rotation angle between the two observations.

4. Method

We now present Ditto, a learning framework that builds digital twins of articulated objects through interactive perception. Ditto jointly learns part-level geometry reconstruction and articulation model estimation with structured feature grids and unified implicit neural representations. Fig. 2 illustrates the overall model architecture. Ditto consists of a two-stream encoder that fuses two input point clouds and multiple implicit decoders for geometry and articulation. The model is jointly optimized with a combination of loss functions on geometry reconstruction and articulation estimation. Upon inference, we extract explicit models of articulated objects from the implicit decoders.

[1] http://wiki.ros.org/urdf

4.1. Two-Stream Encoder

To jointly learn the 3D reconstruction and articulation model estimation, we need to extract features that fuse the information from the input pair of point clouds. We build our encoder based on ConvONets [43], the state-of-the-art implicit representation-based 3D reconstruction method.

We use an attention layer [54] to fuse the two sets of point features of two input point clouds. The complexity of attention operations exhibits quadratic growth with respect to the number of points. To process more dense point clouds which capture finer details of the object, we use a PointNet++ [44] encoder μ_{enc} to obtain two sets of subsampled point features $f_1 = \mu_{\text{enc}}(\mathcal{P}_1)$ and $f_2 = \mu_{\text{enc}}(\mathcal{P}_2)$, where $f_1, f_2 \in \mathbb{R}^{N' \times d_{\text{sub}}}$, $N' < N$ is the number of the subsampled points, and d_{sub} is the dimension of the subsampled point features. A scaled dot-product attention operation is applied to these subsampled feature points

$$Attn_{12} = \text{softmax}(\frac{f_1 f_2^T}{\sqrt{d_{\text{sub}}}}) f_2, \quad f_{12} = [f_1, Attn_{12}]. \quad (1)$$

The fused subsampled point features $f_{12} \in \mathbb{R}^{N' \times 2d_{\text{sub}}}$ is the concatenation of the f_1 and the output of attention. Then we use two PointNet++ decoder ν_{geo} and ν_{art} to propagate the fused subsampled point features into dense features aligned with the original points

$$f_{\text{geo}} = \nu_{\text{geo}}(f_{12}) \quad \text{and} \quad f_{\text{art}} = \nu_{\text{art}}(f_{12}), \quad (2)$$

where $f_{\text{geo}}, f_{\text{art}} \in \mathbb{R}^{N \times d_{\text{dense}}}$ are d_{dense}-dim point features aligned with $\mathcal{P}_1$. We use two separate sets of dense point features because geometry reconstruction mainly exploits the static observation, while the articulation estimation relies more on the correspondence between two observations. Also, these features are processed separately. f_{art} is projected into 2D feature planes and f_{geo} is projected into voxel grids as in the ConvONets [43]. The points that fall into the same pixel cell or voxel cell are aggregated together via max pooling. This projection operation greatly reduces the computation cost while keeping the spatial distribution of feature points. We apply the projection to three canonical planes $\mathbf{c}$ and a coarse voxel grid $\mathbf{v}$. The resulting feature planes and grid are processed with independent 2D and 3D UNets [46]. The output voxel feature grid is used for the geometry implicit decoder, and the feature planes are used for the articulation implicit decoders. Geometry reconstruction requires dense feature grids for fine-grained and local reasoning, while sparse feature planes are sufficient for articulation estimation. Therefore we choose this separate feature representation.

4.2. Implicit Decoders

Motivated by recent works that demonstrate the continuity and versatility of implicit neural representations [11, 19, 33, 40], we design implicit decoders for both geometry and articulation reasoning. As both tasks require reasoning about fine-grained geometry details, we condition the implicit decoders on local features. These local features can be computed from the feature grid/planes using trilinear/bilinear sampling given a query 3D coordinate $\mathbf{p} \in \mathbb{R}^3$.

4.2.1 Geometry Implicit Decoder

Our geometry implicit decoder is a mapping from a coordinate $\mathbf{p} \in \mathbb{R}^3$ to the occupancy probability $o(\mathbf{p})$ at the coordinate. The occupancy $o(\mathbf{p})$ should be 1 if the point $\mathbf{p}$ is occupied by the object and 0 otherwise. We query the local feature $\psi_{\mathbf{p}}^{\mathbf{v}}$ from the feature grid $\mathbf{v}$ using trilinear sampling. Conditioned on the query point coordinate $\mathbf{p}$ and local feature $\psi_{\mathbf{p}}^{\mathbf{v}}$, our geometry implicit decoder predicts the occupancy probability:

$$f_{\theta_o}(\mathbf{p}, \psi_{\mathbf{p}}^{\mathbf{v}}) \rightarrow o(\mathbf{p}). \quad (3)$$

4.2.2 Articulation Implicit Decoders

Our articulation implicit decoders map from an arbitrary point $\mathbf{p}_{\text{in}}$ inside the object to the segmentation label and joint parameters with respect to this point. We only consider the space inside the object because articulation is only meaningful for points in this space. We query the local feature $\psi_{\mathbf{p}_{\text{in}}}^{\mathbf{c}}$ from the feature planes $\mathbf{c}$ using bilinear sampling.

Segmentation. Since we assume that only one joint's state is changed due to the interaction, we can segment the object into the static and mobile parts during each interaction. Therefore we predict a binary segmentation label $s(\mathbf{p}_{\text{in}})$ where 0 stands for the static part and 1 stands for the mobile part. Our segmentation implicit decoder predicts the segmentation probability conditioning on the local feature:

$$f_{\theta_{\text{seg}}}(\mathbf{p}_{\text{in}}, \psi_{\mathbf{p}_{\text{in}}}^{c}) \rightarrow s(\mathbf{p}_{\text{in}}). \quad (4)$$

Joint Parameters. Even though the joint is a global property of an articulated object, we use a per-point representation to better utilize our structured feature representation and get a more robust estimation through voting. We share the feature planes for articulation and segmentation prediction since the articulation can be inferred from motion cues like segmentation. First, we use an implicit decoder to predict joint type $p_{j_{\text{type}}}$:

$$f_{\theta_{\text{type}}}(\mathbf{p}_{\text{in}}, \psi_{\mathbf{p}_{\text{in}}}^{c}) \rightarrow p_{j_{\text{type}}}(\mathbf{p}_{\text{in}}). \quad (5)$$

Then we use two implicit decoders to predict parameters and states of prismatic joints and revolute joints. The prismatic joint is defined by its translation axis direction, a 3D unit vector $\mathbf{u}^p$. The joint state is the translation distance c^p resulting from the interaction. Revolute joint parameters include the rotation axis direction $\mathbf{u}^r$. Different from the

prismatic joint, the position of the revolute joint axis also matters. We follow Li *et al*. [26] and define the joint position with respect to point $\mathbf{p}_{\text{in}}$ as the projection of $\mathbf{p}_{\text{in}}$ to the axis, represented by a 3D unit vector for the projection direction $\mathbf{d}^r_{\mathbf{p}_{\text{in}}}$ and a scalar $h^r_{\mathbf{p}_{\text{in}}}$ for the projection distance. The joint state of the revolute joint is the rotation angle c^r resulting from the interaction. We directly predict these parameters with the implicit decoders:

$$\begin{aligned} f_{\theta_{\text{param}_p}}(\mathbf{p}_{\text{in}}, \psi^c_{\mathbf{p}_{\text{in}}}) &\rightarrow [\mathbf{u}^p, c^p], \\ f_{\theta_{\text{param}_r}}(\mathbf{p}_{\text{in}}, \psi^c_{\mathbf{p}_{\text{in}}}) &\rightarrow [\mathbf{u}^r, \mathbf{d}^r_{\mathbf{p}_{\text{in}}}, h^r_{\mathbf{p}_{\text{in}}}, c^r]. \end{aligned} \tag{6}$$

4.3. Training

Our method does not assume known joint types during inference. Therefore, the model can be trained with data from different categories. The loss for training consists of two parts: the geometry loss and the joint loss. The geometry loss optimizes the part-level geometry reconstruction, and the joint estimation loss optimizes joint estimation.

Geometry Loss. We apply standard binary cross-entropy losses on occupancy and segmentation predictions,

$$\begin{aligned} \mathcal{L}_{\text{occ}} &= \sum_{\mathbf{p}} BCE(o(\mathbf{p}), \hat{o}(\mathbf{p})), \\ \mathcal{L}_{\text{seg}} &= \sum_{\mathbf{p}_{\text{in}}} BCE(s(\mathbf{p}_{\text{in}}), \hat{s}(\mathbf{p}_{\text{in}})), \end{aligned} \tag{7}$$

where $\hat{o}(\mathbf{p})$ is the ground truth occupancy at $\mathbf{p}$ and $\hat{s}(\mathbf{p}_{\text{in}})$ is the ground truth segmentation label at $\mathbf{p}_{\text{in}}$.

Joint Loss. We have three implicit decoders that predict joint type, prismatic joint parameters, and revolute joint parameters respectively. For joint type prediction, we also apply the standard binary cross entropy loss. The joint type loss is denoted as $\mathcal{L}_{\text{type}} = \sum_{\mathbf{p}_{\text{in}}} BCE(p_{j_{\text{type}}}(\mathbf{p}_{\text{in}}), \hat{t})$, where $\hat{t}$ is the ground truth joint type.

Prismatic Joint. We penalize the orientation difference between the estimated joint axis and the ground truth one with loss $\mathcal{L}_{\text{ori}_p} = \arccos(\mathbf{u}^p \cdot \hat{\mathbf{u}}^p)$. The state prediction is optimized with simple ℓ_1 loss $\mathcal{L}_{\text{state}_p} = |c^p - \hat{c}^p|$, where $\hat{c}^p$ is the ground truth joint state. Besides, we also minimize the different between the predicted displacement and ground truth one. The state prediction and parameter prediction can be jointly optimized with this loss $\mathcal{L}_{\text{disp}_p} = ||c^p\mathbf{u}^p - \hat{c}^p\hat{\mathbf{u}}^p||$. All together we have joint loss of the prismatic joint

$$\mathcal{L}_{\text{param}_p} = \sum_{\mathbf{p}_{\text{in}}} (\mathcal{L}_{\text{ori}_p} + \mathcal{L}_{\text{state}_p} + \mathcal{L}_{\text{disp}_p}). \tag{8}$$

Revolute Joint. The loss for axis orientation and joint state of the revolute joint is the same as the prismatic joint, denoted as $\mathcal{L}_{\text{ori}_r}$ and $\mathcal{L}_{\text{state}_r}$. We apply the same loss for orientation of the projection direction $\mathbf{d}^r_{\mathbf{p}_{\text{in}}}$ and projection distance $h^r_{\mathbf{p}_{\text{in}}}$, which are added together to form $\mathcal{L}_{\text{pos}_r}$. Thanks to our dense joint representation, displacement based loss can be also applied to revolute joint parameters prediction. For each point $\mathbf{p}_{\text{in}}$, we compute predicted rotation matrix $R_{\mathbf{p}_{\text{in}}}$ and ground truth one $\hat{R}_{\mathbf{p}_{\text{in}}}$ based on predicted and ground truth axis orientation and rotation angle. We also locate the estimated pivot point on the axis $\mathbf{q}_{\mathbf{p}_{\text{in}}} = \mathbf{p}_{\text{in}} + h^r_{\mathbf{p}_{\text{in}}}\mathbf{d}^r_{\mathbf{p}_{\text{in}}}$. Then the displacement can be computed as $\mathbf{l}_{\mathbf{p}_{\text{in}}} = R_{\mathbf{p}_{\text{in}}}(\mathbf{p}_{\text{in}} - \mathbf{q}_{\mathbf{p}_{\text{in}}}) + \mathbf{q}_{\mathbf{p}_{\text{in}}}$. The ground truth displacement $\hat{\mathbf{l}}_{\mathbf{p}_{\text{in}}}$ can be computed similarly with the ground truth parameters. And the displacement loss $\mathcal{L}_{\text{disp}_r} = ||\mathbf{l}_{\mathbf{p}_{\text{in}}} - \hat{\mathbf{l}}_{\mathbf{p}_{\text{in}}}||$. Moreover, we apply an extra loss on rotation matrix following ScrewNet [18] $\mathcal{L}_{\text{rot}_r} = ||\mathbf{I}_{3,3} - R_{\mathbf{p}_{\text{in}}}\hat{R}^T_{\mathbf{p}_{\text{in}}}||$. All together we have joint loss of the revolute joint

$$\mathcal{L}_{\text{param}_r} = \sum_{\mathbf{p}_{\text{in}}} (\mathcal{L}_{\text{ori}_r} + \mathcal{L}_{\text{state}_r} + \mathcal{L}_{\text{pos}_r} + \mathcal{L}_{\text{disp}_r} + \mathcal{L}_{\text{rot}_r}). \tag{9}$$

Since the joint type is unknown, we need to dynamically apply the prismatic or revolute joint loss based on the joint type. The full loss is

$$\mathcal{L} = \mathcal{L}_{\text{occ}} + \mathcal{L}_{\text{seg}} + \mathcal{L}_{\text{type}} + \mathbb{1}_p\mathcal{L}_{\text{param}_p} + \mathbb{1}_r\mathcal{L}_{\text{param}_r}, \tag{10}$$

where $\mathbb{I}_p$ and $\mathbb{I}_r$ are indicators of whether the ground truth joint type is prismatic or revolute.

4.4. Explicit Articulated Object Extraction

We need to extract the explicit part-level meshes and articulation model from the learned implicit representation to build interactive digital twins that can be spawned in a virtual environment.

Part-Level Mesh Extraction. To reconstruct part-level meshes, we mask the occupancy query results with segmentation query results. The occupancy query results of mobile part and static part are

$$\begin{aligned} o_m(\mathbf{p}) &= \mathbb{1}[o(\mathbf{p}) > t_{\text{occ}}]\mathbb{1}[s(\mathbf{p}) > t_{\text{seg}}], \\ o_s(\mathbf{p}) &= \mathbb{1}[o(\mathbf{p}) > t_{\text{occ}}]\mathbb{1}[s(\mathbf{p}) \leq t_{\text{seg}}], \end{aligned} \tag{11}$$

where t_{occ} and t_{seg} are thresholds for the predicted occupancy and segmentation probability. Then we apply Multiresolution IsoSurface Extraction [33] and Marching Cube [29] to extract per-part surface meshes.

Global Articulation Model Extraction. We use a simple average voting strategy to aggregate the dense joint prediction. During the mesh extraction, we can sample numerous points inside the mesh with their predicted label. Because the object's motion determines the articulation model, we only let points inside the mobile part vote for the global joint. For joint axis direction and joint state of both types of joint, we average all mobile points' predictions. As for the position of the revolute axis, we compute the pivot point coordinate of each mobile point using the predicted projection direction and distance. Then we average the results of all mobile points and get the estimated pivot point on the axis.

Dataset	Method	Geometry		Joint		
		Whole Chamfer Distance ↓	Mobile Chamfer Distance ↓	Prismatic	Revoulute	
				Angle Err ↓	Angle Err ↓	Pos Err ↓
Synthetic Dataset [1]	A-SDF [35]	2.48	-	-	-	-
	Correspondence [7]	2.13	93.2	10.3	46.5	0.46
	Global Joint [18]	0.54	37.7	0.69	52.0	0.13
	Ditto (Ours)	**0.38**	**0.21**	**0.06**	**0.72**	**0.03**
Shape2Motion [55]	Correspondence [7]	2.22	35.7	15.2	45.5	0.28
	Global Joint [18]	0.90	64.9	1.36	79.8	0.17
	Share Feature	0.75	10.7	**0.07**	2.22	0.04
	Concat Fusion	0.97	3.09	0.17	3.13	0.03
	Share Decoder	**0.68**	3.30	0.19	1.93	**0.02**
	Ditto (Ours)	0.72	**0.42**	0.08	**1.36**	**0.02**

Table 1. Quantitative results of geometry reconstruction and articulation estimation on Shape2Motion [55] and synthetic [1] datasets.

5. Experiments

We examine Ditto's ability to recreate digital twins of articulated objects. We first perform systematic quantitative evaluations on two 3D asset datasets, showing that Ditto can accurately reconstruct the geometry and estimate the articulation model. We then qualitatively show that our method generalizes well to objects in the real world.

5.1. Datasets

We conduct experiments on two 3D articulated object datasets, the synthetic objects dataset provided by Abbatematteo *et al*. [1] and the Shape2Motion dataset [55]. The synthetic dataset contains procedurally generated articulated objects. Shape2Motion contains human-designed objects. We select four categories from each dataset. For Shape2Motion dataset, we choose four categories with more than 30 instances. We choose 4 out of 6 categories for the synthetic dataset because the other two are very similar to the chosen ones. During data generation, we randomly spawn an object in simulation and set the object into random start and end states to mimic articulated motions. In each state, we fuse multi-view depth images into point cloud observation. Even though we use multi-view depth images, the point cloud may still be incomplete due to the self-occlusion of the objects. We generate occupancy data points for each part separately for ground truth geometry and aggregate the samples to get shape-level occupancy and segmentation. Ground-truth articulation is directly acquired from the simulator and the ground-truth occupancy is queried from the ground truth mesh as in [43].

5.2. Baselines

A-SDF. A-SDF [35] is the closest work to ours, given that no existing method is designed specifically for the full-fledged virtual recreation of articulated objects. There are two main differences between A-SDF and our work. First, A-SDF is a category-level model that assumes the same kinematic tree structures of objects in the same category. Second, it estimates the articulation model implicitly rather than explicitly. Accordingly, we train one A-SDF model for each category and evaluate only the geometry reconstruction result on the synthetic dataset.

Correspondence. We first train an FCGF [7] feature extractor on the whole dataset. Then we use the extracted features to find point correspondence across the observations before and after the interaction. An articulation model is fitted based on correspondence and using the non-linear least square algorithm. Besides, we use correspondence to compute the moving distance of every point and segment the mobile points with a threshold of 0.02 on this distance. We reconstruct the mesh of segmented points using a ConvONet [43] trained for part reconstruction. This baseline has the same output as Ditto.

Global Joint. To validate our choice of dense joint representation, we modify our model and use decoders that predict joint parameters from a global feature. Since the pivot point of a revolute joint axis is ambiguous, *i.e*., it can move along the axis, we adopt the screw-based joint parametrization in ScrewNet [18]. We also apply the loss function of ScrewNet to train the model.

In addition to the external baselines above, we also use the following ablated versions of our model to validate our design choices:

Concat Fusion. Instead of the attention-based fusion, it directly concatenates the structured features of the pair of point clouds and conditions the local implicit decoders on the concatenated features.

Share Feature. We use 3D feature grids for occupancy prediction and 2D feature planes for segmentation and joint prediction in our current model. This ablated version shares

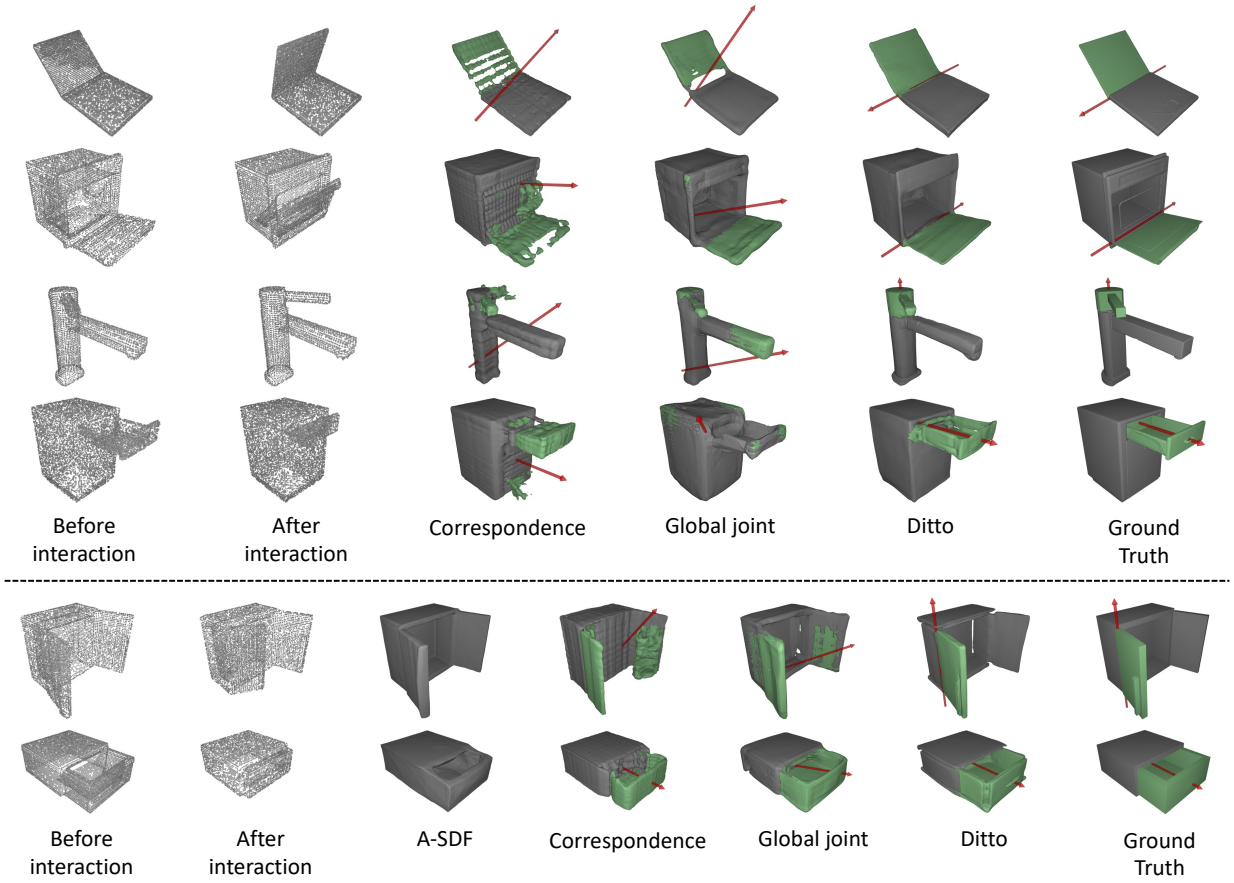

Figure 3. Reconstructed unseen articulated objects in Shape2Motion [55] (top) and synthetic [1] (bottom) dataset. Static parts are colored grey while mobile parts are colored green. We also visualize the estimated joint with the red arrow.

the 3D and 2D features for both geometry and articulation prediction.

Share Decoder. In our current model, we use two separate decoders in PointNet++ for geometry and articulation. This ablated version uses a shared decoder instead.

5.3. Evaluation Metrics

Part-level Geometry. To evaluate the quality of the reconstructed part-level mesh, we use Chamfer-ℓ_1 distance (CD) as the evaluation metric. Apart from the CD between the whole reconstructed mesh and the ground truth, we also evaluate the CD of the segmented mobile part because it is the only interactable region of the object. CD shown are multiplied by 1000 as in A-SDF [35].

Articulation Model. For both types of joints, we measure the axis orientation error (Angle Err). For the revolute joint, we also measure the axis position error (Pos Err) using the minimum distance between the predicted and ground truth rotation axis.

5.4. Articulated Object Reconstruction

The quantitative results are shown in Tab. 1. On both datasets, Ditto gets significantly better results on all metrics compared with the baselines. Both the Correspondence [7] and Global Joint [18] baselines perform poorly on articulation estimation. As shown in Fig. 3, while the baseline methods produce overall well-reconstructed shapes, the predicted mobile parts have many artifacts. In contrast, Ditto achieves precise part-level geometry reconstruction as well as accurate joint estimation.

Due to the two-stage design, the Correspondence baseline highly relies on a learned disentangled feature representation at the beginning. Bad initial feature representation is prone to inaccurate correspondence and articulation estimation. In comparison, Ditto does not suffer from such a bottleneck as an end-to-end method. The Global Joint baseline performs poorly mainly due to the high variance of direct global joint regression. Failure of joint estimation also harms segmentation prediction because the joint parameter decoders and the segmentation decoder share the same feature planes. Differently, Ditto predicts joint parameters with respect to each point. The dense predictions are aggregated

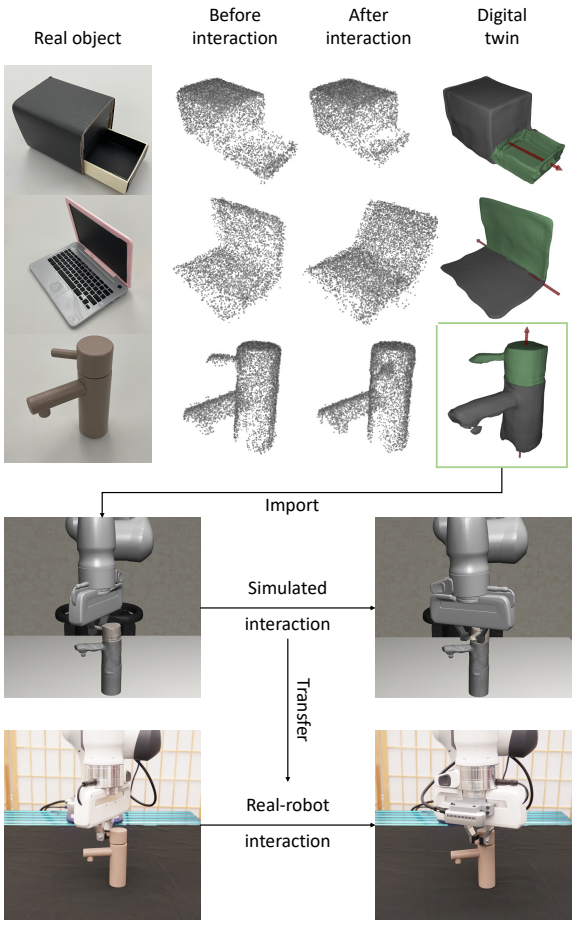

Figure 4. Real-world results. We use Ditto trained in simulated datasets to build the digital twin of these physical objects. The recreated faucet model is imported into a physical simulator. The robot interacts with the virtual faucet and transfers its actions back into the real world to manipulate the physical faucet.

into the final joint estimation and thus lead to a more robust and accurate result.

To compare with A-SDF [35], we provide the shape reconstruction results on the synthetic dataset. When it comes to the whole Chamfer distance, Ditto surpasses A-SDF by a notably large margin. As visualized in Fig. 3, A-SDF fails to reconstruct the shape details of unseen objects, especially the objects with prismatic joints. In contrast, Ditto accurately reconstructs the whole object and the fine-grained geometry detail like the handles of the cabinet door and drawer. A key difference between A-SDF and ours is that we use a feedforward model while A-SDF uses test-time optimization to find the articulation and shape codes. The inferior performance of A-SDF is due to the interference of articulation code and shape code in test-time optimization. Note that A-SDF requires separate training for each category while Ditto is a category-agnostic method. Thus, the task should be more challenging for Ditto. Furthermore, Ditto can extract explicit articulation and part-level geometry while A-SDF encodes the articulation model implicitly. More comparison with A-SDF is presented in the appendix.

5.5. Ablation Studies

As shown in Tab. 1, Ditto achieves superior or at least on-par performance on all metrics. The mobile Chamfer distance (CD) of Ditto is substantially lower (better) than the ablated versions. Mobile CD measures the quality of the reconstructed mobile part, which is vital for simulating interaction. Share Feature baseline has the worst performance in Mobile CD. We observe that using the same 3D and 2D features for geometry and articulation makes training unstable, and 2D features would harm the reconstruction due to the loss of spatial information after projection. Concat Fusion does not reason about correspondence explicitly and thus shows inferior performance on all metrics compared with Ditto. Finally, the Share Decoder baseline applies one decoder for both geometry and motion features. This decoder needs to reason about geometry and articulation simultaneously. Suffering from a limited capacity, this baseline obtains sub-optimal performance on Mobile-CD and joint angle errors. Qualitative results and analysis of ablation study are in the appendix.

5.6. Real-World Experiments

Finally, we use Ditto to recreate digital twins of real-world objects. We choose three daily objects, a toy cabinet, a laptop, and a faucet. The results are shown in Fig. 4. The results have some artifacts due to the noisy and incomplete input point clouds from depth cameras. Despite these artifacts, Ditto can generally reconstruct the geometry and articulation of these physical objects. Moreover, we import the digital twin of the faucet into Robosuite [62], a robot learning simulation framework. We use a simulated robot arm to interact with the digital twin and transfer the actions back to the real world after calibrating the simulated and real robot frames. The video of this experiment is provided on the project website. With Ditto we can recreate a real-world articulated object to the digital twin in a virtual environment and map the interactions with the digital twin back to actions in the real world.

5.7. Limitations

Kinematic tree. Currently Ditto only segments the object into two parts, the mobile and static ones. We hope to extend our method to reconstruct the full kinematic tree of a composite object with multiple joints and parts via consecutive interactions and aggregation of model inference after each interaction.

Active perception. We use interactions to create novel sensory data for inferring articulation. These interactions are either by setting the joint states (in simulation) or by human

(real world). We hope to develop algorithms that enable an agent to autonomously interact with objects to actively collect data.

6. Conclusion

We introduce Ditto, an implicit neural representation-based model for recreating digital twins of articulated objects through interactive perception. Ditto is an end-to-end model that jointly learns full-fledged geometry reconstruction and articulation estimation from two visual inputs before and after articulated motions. Results show that Ditto achieves significantly more accurate results on geometry and articulation reasoning over baselines. Furthermore, we demonstrate Ditto generalizes to real-world objects, and we can directly spawn the recreated digital twins in interactive simulation. These results manifest the potential of autonomous digital twin building in empowering embodied AI research and AR/VR applications.

Acknowledgments

We would like to thank Yifeng Zhu for his help with real robot experiments. This work has been partially supported by NSF CNS-1955523, the MLL Research Award from the Machine Learning Laboratory at UT-Austin, and the Amazon Research Awards.

References

[1] Ben Abbatematteo, Stefanie Tellex, and George Konidaris. Learning to generalize kinematic models to novel objects. In *Proceedings of the 3rd Conference on Robot Learning*, 2019. 1, 2, 3, 6, 7, 12, 13

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. 1

[3] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2614–2623, 2019. 1

[4] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017. 2

[5] Aljaz Bozic, Pablo Palafox, Michael Zollhofer, Justus Thies, Angela Dai, and Matthias Nießner. Neural deformation graphs for globally-consistent non-rigid reconstruction. In *CVPR*, pages 1450–1459, 2021. 2

[6] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 2

[7] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8958–8966, 2019. 6, 7, 12

[8] Anthony Dearden and Yiannis Demiris. Learning forward models for robots. In *IJCAI*, volume 5, page 1440, 2005. 2

[9] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, et al. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3164–3174, 2020. 3

[10] Boyang Deng, John P Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Nasa: Neural articulated shape approximation. In *European Conference on Computer Vision*, pages 612–628, 2020. 1

[11] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. *arXiv preprint arXiv:2109.00137*, 2021. 4

[12] Samir Yitzhak Gadre, Kiana Ehsani, and Shuran 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. 2

[13] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *2011 IEEE intelligent vehicles symposium (IV)*, pages 963–968. Ieee, 2011. 1

[14] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020. 2

[15] Karol Hausman, Scott Niekum, Sarah Osentoski, and Gaurav S Sukhatme. Active articulation model estimation through interactive perception. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3305–3312. IEEE, 2015. 2

[16] Xiaoxia Huang, Ian Walker, and Stan Birchfield. Occlusion-aware reconstruction and manipulation of 3d articulated objects. In *2012 IEEE International Conference on Robotics and Automation*, pages 1365–1371. IEEE, 2012. 2

[17] Ajinkya Jain, Stephen Giguere, Rudolf Lioutikov, and Scott Niekum. Distributional depth-based estimation of object articulation models. *arXiv preprint arXiv:2108.05875*, 2021. 2

[18] Ajinkya Jain, Rudolf Lioutikov, and Scott Niekum. Screwnet: Category-independent articulation model estimation from depth images using screw theory. *arXiv preprint arXiv:2008.10518*, 2020. 1, 2, 5, 6, 7

[19] Zhenyu Jiang, Yifeng Zhu, Maxwell Svetlik, Kuan Fang, and Yuke Zhu. Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. *arXiv preprint arXiv:2104.01542*, 2021. 4

[20] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and

pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7122–7131, 2018. 1

[21] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4551–4560, 2019. 1

[22] Dov Katz and Oliver Brock. Manipulating articulated objects with interactive perception. In *2008 IEEE International Conference on Robotics and Automation*, pages 272–277. IEEE, 2008. 2

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 12

[24] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 1, 3

[25] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021. 1, 3

[26] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3706–3715, 2020. 1, 2, 3, 5

[27] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *arXiv preprint arXiv:2007.11571*, 2020. 2

[28] Qihao Liu, Weichao Qiu, Weiyao Wang, Gregory D Hager, and Alan L Yuille. Nothing but geometric constraints: A model-free method for articulated object pose estimation. *arXiv preprint arXiv:2012.00088*, 2020. 2

[29] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 5

[30] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017. 1

[31] Roberto Martín-Martín and Oliver Brock. Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2494–2501. IEEE, 2014. 2

[32] Roberto Martín-Martín, Sebastian Höfer, and Oliver Brock. An integrated approach to visual perception of articulated objects. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5091–5097. IEEE, 2016. 2

[33] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 2, 4, 5

[34] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2019. 3

[35] Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. *arXiv preprint arXiv:2104.07645*, 2021. 1, 2, 6, 7, 8, 12

[36] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015. 1

[37] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE, 2011. 1

[38] Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Neural articulated radiance field. *arXiv preprint arXiv:2104.03110*, 2021. 1

[39] Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter Gehler, and Bernt Schiele. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In *2018 international conference on 3D vision (3DV)*, pages 484–494. IEEE, 2018. 1

[40] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven 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. 2, 4

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 12

[42] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 459–468, 2018. 1

[43] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceed-*

ings, Part III 16, pages 523–540. Springer, 2020. 2, 3, 4, 6

[44] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 2, 3, 4

[45] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *International Conference on Computer Vision (ICCV)*, 2021. 1

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and Computer-Assisted Intervention*, pages 234–241, 2015. 4

[47] Manolis Savva, Angel X Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931*, 2017. 3

[48] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[49] Jürgen Sturm, Christian Plagemann, and Wolfram Burgard. Adaptive body scheme models for robust robotic manipulation. In *Robotics: Science and systems*. Zurich, 2008. 2

[50] Jurgen Sturm, Christian Plagemann, and Wolfram Burgard. Unsupervised body scheme learning through self-perception. In *2008 IEEE International Conference on Robotics and Automation*, pages 3328–3333. IEEE, 2008. 2

[51] Jürgen Sturm, Vijay Pradeep, Cyrill Stachniss, Christian Plagemann, Kurt Konolige, and Wolfram Burgard. Learning kinematic models for articulated objects. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009. 2

[52] Jürgen Sturm, Cyrill Stachniss, and Wolfram Burgard. A probabilistic framework for learning kinematic models of articulated objects. *Journal of Artificial Intelligence Research*, 41:477–526, 2011. 2

[53] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. *arXiv preprint arXiv:2106.14405*, 2021. 1, 3

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 4

[55] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8876–8884, 2019. 1, 2, 3, 6, 7, 12, 13

[56] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, William T Freeman, and Joshua B Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. *arXiv preprint arXiv:1711.03129*, 2017. 1

[57] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He 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. 3

[58] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Huiwen Chang, Deva Ramanan, William T Freeman, and Ce Liu. Lasr: Learning articulated shape reconstruction from a monocular video. In *CVPR*, pages 15980–15989, 2021. 2

[59] Gengshan Yang, Minh Vo, Neverova Natalia, Deva Ramanan, Vedaldi Andrea, and Joo Hanbyul. Banmo: Building animatable 3d neural models from many casual videos. *arXiv preprint arXiv:2112.12761*, 2021. 2

[60] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. *arXiv preprint arXiv:1809.07417*, 2018. 2

[61] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3357–3364. IEEE, 2017. 1, 3

[62] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020. 8

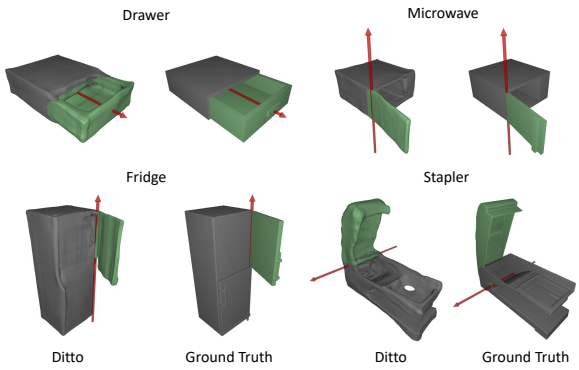

Figure 5. Qualitative results of generalizing to unseen categories.

A. Implementation Details

We use the Shape2Motion dataset [55] and the Synthetic dataset [1]. The Shape2Motion dataset is licensed under the GNU General Public License v3.0. We sample 8,192 points for each input point cloud. In each iteration, we sample 2,048 pairs of $\mathbf{p}$ and corresponding occupancy. We also 512 pairs of $\mathbf{p}_{in}$ inside the object and corresponding segmentation and joint parameters as query points and ground truths. $\mathbf{p}$ and $\mathbf{p}_{in}$ are input query points for geometry decoder and articulation decoders separately, as described in Sec. 4.2.

We implement the models with Pytorch [41] and train the models with the Adam [23] optimizer and a learning rate of 10^{-4} and batch sizes of 8.

B. Ablation Study

Ditto uses 3D feature grid for geometry reconstruction and 2D feature plane for articulation estimation. To validate the advantage of this design, we evaluate another ablated version where two 3D feature grid are used for geometry and articulation respectively. As in Tab. 2, this ablated version has similar performance to Ditto. But it requires around 20% more memory usage and training time compared with Ditto.

We show some qualitative results in Fig. 6. Our full Ditto model can recreate the articulated objects more accurately, especially the mobile part, benefiting from the attention-based fusion, separate decoders and features. In comparison, Concat Fusion and Share Feature are not able to reconstruct the smooth and complete surface. The ablated version with Share Feature uses 2D feature planes along with 3D feature grids for geometry reconstruction. This projection operation results in artifacts as in the faucet result in Fig. 6 (red circle in the second row, first column). The ablated version with Share Decoder has a problem segmenting the mobile and the static parts correctly. Overall, Ditto can achieve the best performance on the reconstruction of the articulated objects.

Method	Whole CD ↓	Mobile CD
A-SDF (oracle code) [35]	0.66	-
Ditto (3D+3D feature)	0.27	0.12
Ditto	0.25	0.16

Table 2. Quantitative results of reconstruction on the Synthetic [1] dataset.

Method	Chamfer Distance ↓
A-SDF [35]	3.57
Ditto	**0.37**

Table 3. Quantitative results of articulated motion synthesis on the Synthetic [1] dataset.

Method	Joint type accuracy (%)
Global Joint [7]	88
Share Feature	100
Concat Fusion	96
Share Decoder	100
Ditto (Ours)	100

Table 4. Quantitative results of joint type prediction accuracy on the Shape2Motion [55] dataset.

C. Comparison with A-SDF

To explore the possible reason behind the inferior performance in Tab. 1, we try fixing the A-SDF's articulation code to the ground-truth one. The reconstruction result is improved and close to Ditto as in Tab. 2. It indicates that the inferior performance of A-SDF is caused by the in- terference between articulation and shape codes in test-time optimization. For example, the shape code degrades when the articulation code is in a local minimum far from the ground truth. In contrast, the articulation and geometry predictions do not interfere with each other in our model.

A-SDF [35] can control the joint state by changing the articulation code. On the other hand, Ditto explicitly reconstructs the explicit part-level meshes and the articulation model, where the joint state can also be easily controlled. It is thus possible to compare the performance of articulated motion synthesis of A-SDF and Ditto. We first reconstruct the articulated object and then manipulate the articulated object to a new joint state. And we measure the whole Chamfer distance between manipulated results and the ground truth objects after such an articulated motion.

The quantitative results are in Tab. 3. Ditto achieves significantly better results compared with A-SDF. We also

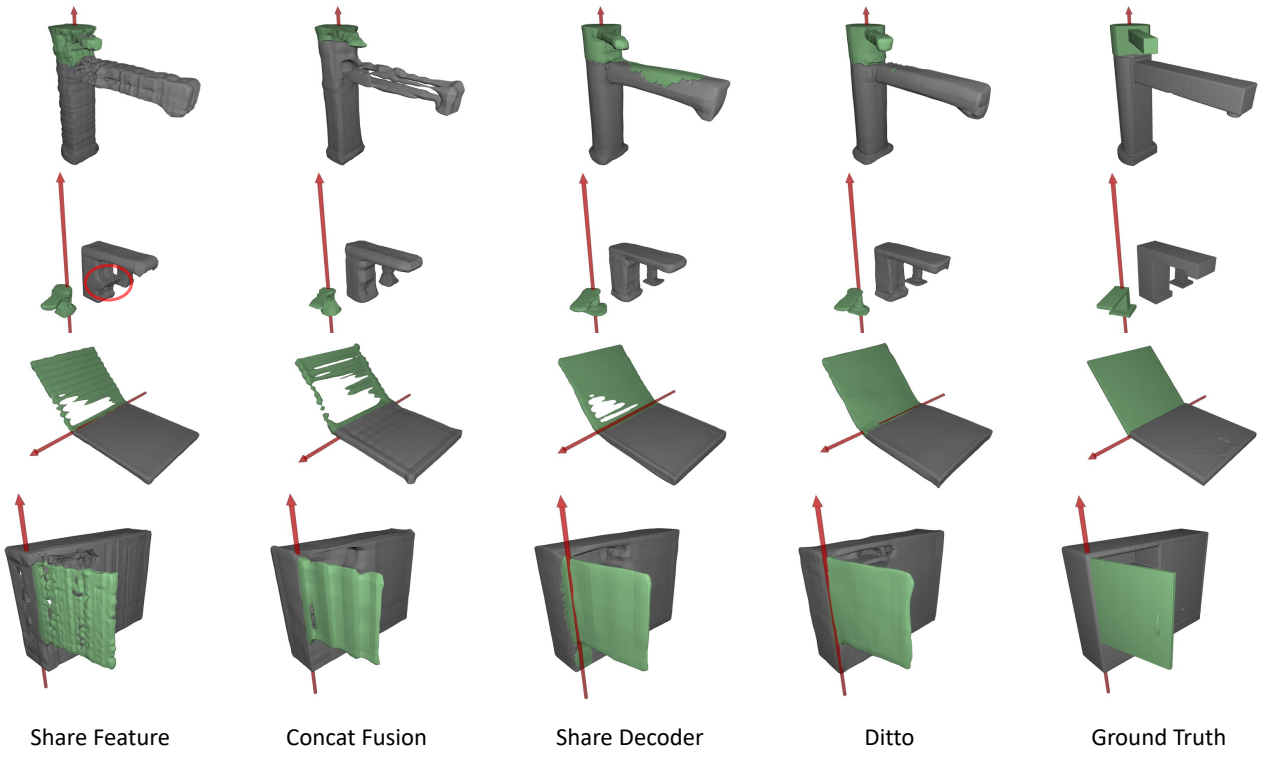

Figure 6. Reconstructed unseen articulated objects in the Shape2Motion [55] dataset of ablated versions. Static parts are colored grey while mobile parts are colored green. We also visualize the estimated joint with the red arrow.

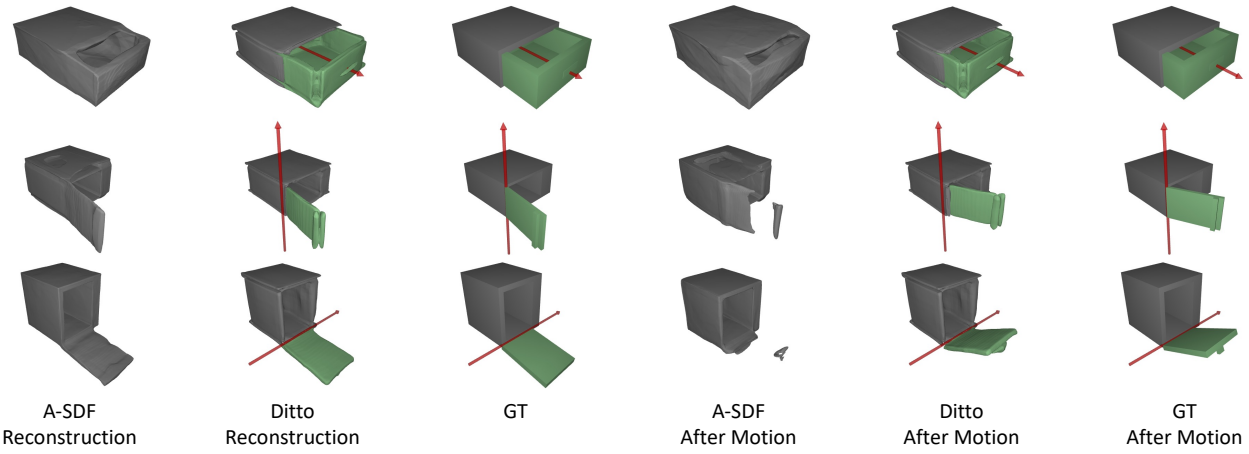

Figure 7. Objects after articulated motion on the Synthetic [1] dataset. Static parts are colored grey while mobile parts are colored green. We also visualize the estimated joint with the red arrow.

show some qualitative results in Fig. 7. Even though A-SDF can generally reconstruct the articulated object from the observation, the results after the articulated motion are not consistent with the initial state due to its latent representation of articulation. For example, the whole drawer body is widened after the motion. In contrast, Ditto explicitly extracts mobile part mesh and the corresponding joint parameters. Apart from the rigid transformation induced by articulated motion, there is no unexpected distortion after the motion.

D. Joint Type Prediction

Our model is also predicting the joint type. All methods give 100% joint type accuracy on the Synthetic dataset. We provide the results on the accuracy of joint type prediction on the Shape2Motion dataset in Tab. 4. Most methods also acquire 100% accuracy on this dataset except the global joint baseline and the ablated version with concat fusion.

E. Generalization to Unseen Categories

In our experiments, we trained our model for four categories altogether and evaluated it on the same four categories. To test generalization to novel categories, we run our model, trained on Shape2Motion, on four unseen categories (drawer, microwave, fridge, and stapler). Fig. 5 shows that our model generalizes robustly to geometrically similar categories (drawer, microwave, and fridge) but slightly worse on the new categories of more significant differences (stapler) as it learns shape priors from the training data.