This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

CARTO: Category and Joint Agnostic Reconstruction of ARTiculated Objects

Nick Heppert[1,3], Muhammad Zubair Irshad[2], Sergey Zakharov[3], Katherine Liu[3],
Rares Andrei Ambrus[3], Jeannette Bohg[4], Abhinav Valada[1], Thomas Kollar[3]

[1]University of Freiburg [2]Georgia Institute of Technology
[3]Toyota Research Institute (TRI) [4]Stanford University

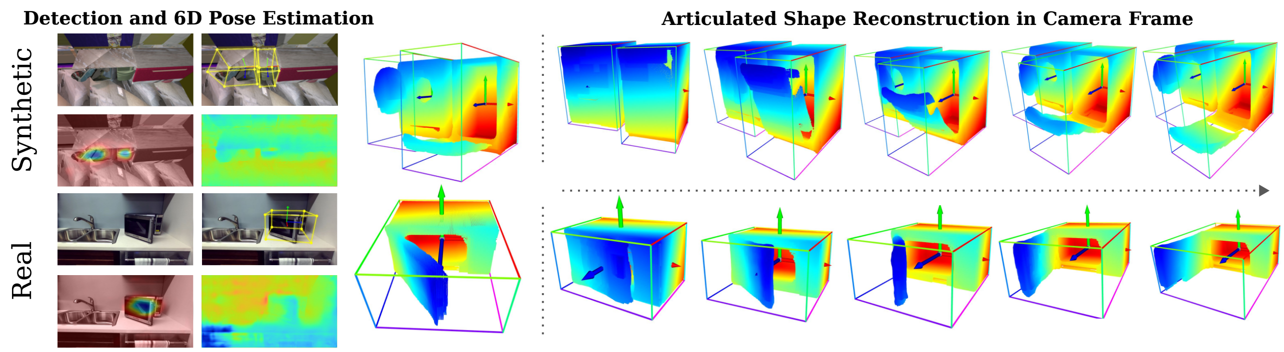

Figure 1. Visualization of CARTO on unseen object instances. We first use CARTO to jointly detect all objects in the scene and then articulate them while keeping the predicted shape code constant.

Abstract

We present CARTO, a novel approach for reconstructing multiple articulated objects from a single stereo RGB observation. We use implicit object-centric representations and learn a single geometry and articulation decoder for multiple object categories. Despite training on multiple categories, our decoder achieves a comparable reconstruction accuracy to methods that train bespoke decoders separately for each category. Combined with our stereo image encoder we infer the 3D shape, 6D pose, size, joint type, and the joint state of multiple unknown objects in a single forward pass. Our method achieves a 20.4% absolute improvement in mAP 3D IOU50 for novel instances when compared to a two-stage pipeline. Inference time is fast and can run on a NVIDIA TITAN XP GPU at 1 HZ for eight or less objects present. While only trained on simulated data, CARTO transfers to real-world object instances. Code and evaluation data is available at: `carto.cs.uni-freiburg.de`

1. Introduction

Reconstructing 3D shapes and inferring the 6D pose and sizes of objects from partially observed input observations remains a fundamental problem in computer vision with applications in Robotics [10, 11, 13, 20] and AR/VR [8, 44]. This object-centric 3D scene understanding problem is challenging and under-constrained since inferring 6D pose and shape can be ambiguous without prior knowledge about the object of interest. Previous work has shown that it is possible to perform category-level 3D shape reconstruction and 6D pose estimation in real-time [7], enabling the reconstruction of complete, fine-grained 3D shapes and textures. However, there are a wide variety of real-world objects that do not have a constant shape but can be articulated according to the object's underlying kinematics. There has been great progress in articulated object tracking [5,9,32,37] and reconstruction [10, 26] from a sequence of observations. However, a sequence of observations is cumbersome since it often requires prior interaction with the environment. In contrast, object reconstruction from a single stereo image through inferring latent information about an object a priori enables both grasping and manipulation of previously unknown articulated objects. Additionally, estimates from a single image can also serve as a good initial guess for object tracking approaches [37].

Previous approaches to articulated object reconstruction from a single observation use a two-stage approach [16] where first objects are detected using, e.g., Mask-RCNN [4]. Then, based on the detection output, object properties, e.g. part-poses and NOCS maps [35], are predicted and the object is reconstructed using backward optimization [24]. Such an approach is complex, error prone, does not scale across many categories, and does not run in real-time.

To mitigate the aforementioned challenges, building on ideas from [7] - a single-shot approach to output complete 3D information (3D shape, 6D pose, and sizes of multiple objects) on a per-pixel manner - we present "Category and Joint Agnostic Reconstruction of ARTiculated Objects" (CARTO). First, extending [24], we train a robust category- and joint-agnostic 3D decoder [3, 24, 28] by learning disentangled latent shape and joint codes. The shape code encodes the canonical shape of the object while the joint code captures the articulation state of the object consisting of the type of articulation (e.g., prismatic or revolute) and the amount of articulation (i.e. joint state). To disentangle both codes we impose structure among our learned joint codes by proposing a physically grounded regularization term. Second, in combination with our stereo RGB image encoder we can do inference in a *single-shot manner* to detect the objects' spatial centers, 6D poses and sizes as well as shape and joint codes. The latter two can then be used as input to our category- and joint-agnostic decoder to directly reconstruct all detected objects.

To evaluate CARTO, we first evaluate the reconstruction and articulation state prediction quality of our category- and joint-agnostic decoder and compare against decoders trained on a single category. In an ablation study we show the necessity of our proposed joint code regularization over naively training the joint codes. We then quantitatively compare our full pipeline to a two-stage approach on synthetic data and show qualitative results on a new real-world dataset.

Our main contributions can be summarized as follows:

- An approach for learning a shape and joint decoder jointly in a category- and joint-agnostic manner.
- A single shot method, which in addition to predicting 3D shapes and 6D pose, also predicts the articulation amount and type (prismatic or revolute) for each object.
- Large-scale synthetic and annotated real-world evaluation data for a set of articulated objects across 7 categories.
- Training and evaluation code for our method.

2. Related Work

Related work to CARTO includes neural fields for reconstruction, implicit reconstructions of non-rigid objects and articulated object detection, pose estimation, and reconstruction.

Neural Fields for Reconstruction: Neural fields, i.e. coordinate-based multi-layer perceptrons [39], have become a popular method for reconstruction in recent years. These methods encode continuous functions that model various scene properties, such as Signed Distance [28], radiance [23], and occupancy [21]. Variations of these include hybrid discrete-continuous representations that employ an external data structure, i.e. a grid or an octree, to partition the implicit function [29, 43]. The encoded shape can then be extracted via sphere tracing [17] after querying the implicit function repeatedly. Recent advances in differential rendering have enabled learning of shapes, as well as other scene properties such as appearance, only from images and without the need for 3D supervision [23]. Our approach falls into the paradigm of using neural fields for articulated object reconstruction and further learns a complete system for detection, pose estimation, and articulated shape reconstruction from a single observation.

Implicit Reconstruction of Non-Rigid Objects: Going beyond static scenes with rigid objects, [1] handle dynamic scenes while [27, 33] focus on reconstructing humans by leveraging their strong shape and kinematic prior as well as the amount of readily available datasets. [42] propose a general reconstruction framework to reconstruct any non-rigid entity (i.e. humans, animals, or objects) given only an RGB-video without requiring a category-specific shape template, while [14] focus on point cloud input data and split the prediction into a canonical shape and a deformation field. One downside of general reconstruction methods is that they do not leverage the rigidity and kinematic constraints of articulated objects. To explicitly use such priors, [26] propose a method that processes a multi-view sequence of a moving object and discovers its parts as well as their respective reconstruction and kinematic structure in an unsupervised way. Going one step further, [36] learn a shape and appearance prior for each category which allows them to model accurate reconstructions of articulated objects with only 6 given views. Similarly, [34] propose a Neural Radiance Field (NeRF) based method that can reconstruct objects on a category level given some images of an object. Also leveraging a learned shape prior over objects of the same category, [24] reconstructs articulated objects using only a single observation by optimizing for a latent shape code; similarly to [34], their method is only tested on revolute objects.

Our approach also represents objects through low-dimensional codes. However, by disentangling the shape from the articulation state in our code, our method becomes category- and joint-agnostic. Other multi-category models, such as [10], reconstruct objects given a point cloud in two different articulation states, which limits the approach to objects with a single joint; [25] uses many observations before and after articulation to reconstruct an object. Most similar to our disentangled representation, [40] learns a latent space in which part-pairs are close if one part-pair can be transformed into another through a valid joint transformation.

Articulated Object Detection, Pose Estimation and Reconstruction: Work in articulated object detection and pose estimation typically first requires the detection of individual parts and their respective poses from a sequence of images demonstrating the articulation [5, 9, 10, 32, 37] or from a single image [15, 16, 22]. [15] combines this part-level view of articulated objects with a holistic object-centric view as done for rigid objects [35]. Similarly, our work predicts the poses for articulated objects in the scene in a single pass from a single stereo or RGB-D image, without the need to detect individual parts first. Most similar to us, [7, 8] also detect the pose, shape and scale of multiple objects from an RGB-D observation via a single-stage approach. Both methods perform category-agnostic detection of unseen object instances at test time, however, they are limited to rigid objects, with [7] using a point cloud decoder for shape reconstruction, while [8] employs a latent shape and appearance prior. Our method also performs category-agnostic object detection and reconstruction, and we extend [7, 8] to handle articulated objects of multiple types in a single network forward pass, thus enabling fast and accurate articulate shape, pose and size estimation from a single stereo image.

3. Technical Approach

In this section, we detail our proposed single-shot detector for articulated objects. Our method consists of two individually learned components: an encoder that predicts latent object codes as well as poses in the camera frame and a decoder that reconstructs objects in a canonical frame that can be transformed into the camera frame through the predicted pose. An overview of our approach is shown in Fig. 2.

3.1. Encoder

Our encoder builds upon CenterSnap [7] and SimNet [13]. For each pixel in our input stereo image $I^{W \times H \times 6}$ we predict an importance scalar ψ, where a higher value indicates closeness to a 2D spatial center in the image of the object. The full output map of ψ represents a heatmap over objects. Additionally, we predict a dense pixel map of canonical 6D poses for each articulated object independent of their articulation state. Further, we extend [7], which predicted a shape code $\boldsymbol{z}_{\mathrm{s}} \in \mathbb{R}^{D_s}$, to also predict a joint code $\boldsymbol{z}_{\mathrm{j}} \in \mathbb{R}^{D_j}$ for each pixel. These codes can be used to predict the articulation state of the object.

Additionally, while not needed for our full pipeline, to guide the network towards important geometric object features, we also predict a semantic segmentation mask as well as 3D bounding boxes, again on a pixel-level. We use these predictions for constructing our baseline as described in Sec. 4.3. Last, we also predict a depth map $D^{W \times H}$. The full network architecture is given in Sec. S.1.1.

During inference of our full pipeline, given the predicted heatmap of importance values, we use non-maximum suppression to extract peaks in the image. At each peak, we then query the feature map to get the pose, shape, and joint code. We convert our 13-dimensional pose vector to a scale value $\in \mathbb{R}$ of the canonical object frame, a position $\in \mathbb{R}^3$ and using [2] to an orientation $\in \mathbb{R}^{3 \times 3}$ in the camera frame. We then use the shape and joint code to reconstruct each object in its canonical object frame using our decoder. After reconstruction, we use the predicted pose to place the object in the camera frame as shown in Fig. 2.

3.2. Decoder

Given a latent code, the decoder reconstructs object geometry, classifies the discrete joint type (i.e., as prismatic or revolute), and predicts the continuous joint state. To disentangle the shape of the object from its articulation state, we split the latent code in two separate codes: a shape and a joint code. We assign the same unique shape code $\boldsymbol{z}_{\mathrm{s}} \in \mathbb{R}^{D_s}$ to an object instance in different articulation states, where an articulation state is expressed through its own joint code variable $\boldsymbol{z}_{\mathrm{j}} \in \mathbb{R}^{D_j}$. We structure our decoder as two sub-decoders, one for reconstructing the geometry (Sec. 3.2.1) and the other for predicting the joint type *jt* and state q (Sec. 3.2.2). See Sec. S.1.2 for a full architecture description.

3.2.1 Geometry Decoder

The geometry decoder ϕ_{geom} reconstructs objects based on a shape code $\boldsymbol{z}_{\mathrm{s}}$ and joint code $\boldsymbol{z}_{\mathrm{j}}$. In principle, the approach is agnostic to the specific decoder architecture as long as it is differentiable with respect to the input latent codes. While there are many potential options such as occupancy maps [21] as adopted in [10], we use signed distance functions (SDFs) [28] due to the proven performance in [24, 28]. Specifically, in the case when using SDFs as our geometry decoder, the model takes as input a point in 3D space $\boldsymbol{x}$ as well as a shape $\boldsymbol{z}_{\mathrm{s}}$ and joint code $\boldsymbol{z}_{\mathrm{j}}$

$$\phi_{\text{geom}}(\boldsymbol{z}_{\mathrm{s}}, \boldsymbol{z}_{\mathrm{j}}, \boldsymbol{x}) = \hat{s}_{\boldsymbol{x}} \tag{1}$$

and predicts a value $\hat{s}_{\boldsymbol{x}}$ that indicates the distance to the surface of the object.

For faster inference, we implement a multi-level refinement procedure [8]. We first sample query points on a coarse grid and refine them around points that have a predicted distance within half of the boundary to the next point. This step can be repeated multiple times to refine the object prediction up to a level l. Eventually, we extract the surface of the objects by selecting all query points $\boldsymbol{x}$ for which $|\hat{s}_{\boldsymbol{x}}| < \epsilon$ holds. By taking the derivative

$$\boldsymbol{n}_{\boldsymbol{x}} = \frac{\partial \phi_{\text{geom}}(\boldsymbol{z}_{\mathrm{s}}, \boldsymbol{z}_{\mathrm{j}}, \boldsymbol{x})}{\partial \boldsymbol{x}} \tag{2}$$

and normalizing it we get the normal $\hat{\boldsymbol{n}}_{\boldsymbol{x}}$ at each point $\boldsymbol{x}$, which can then be used to project the points onto the surface of the object with $\hat{\boldsymbol{x}} = \boldsymbol{x} - \hat{s}_{\boldsymbol{x}} \hat{\boldsymbol{n}}_{\boldsymbol{x}}$.

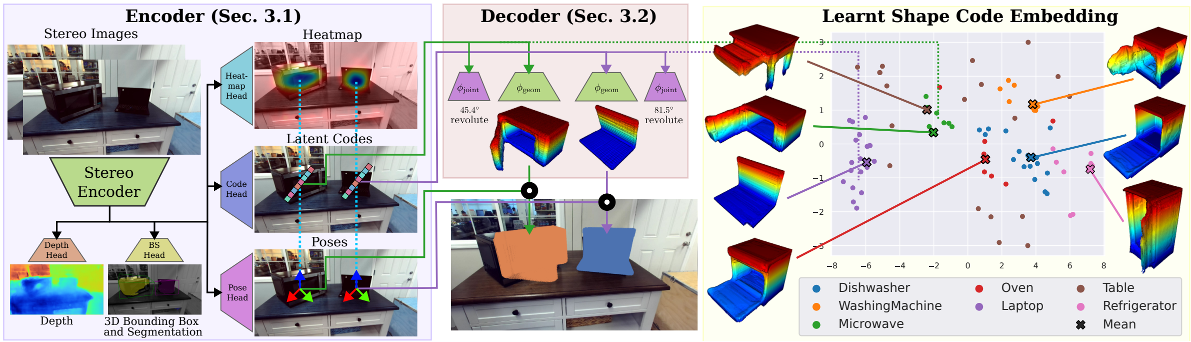

Figure 2. Overview of our proposed method. We first encode a stereo image using [13] to predict a depth and an importance value, a pose as well as a shape and joint code for each pixel using peak detection on the depth map allows us to detect objects which then can be reconstructed given the latent code. Last, to place the objects in camera frame we transform the reconstructed point cloud using the predicted poses at the peaks. On the right side we show the position of the predicted shape codes in a t-SNE visualization of the learned shape codes for the training used as input to our single category- and joint-agnostic decoder. We additionally project each categories mean shape code and use them to reconstruct the objects at the average prismatic and revolute joint state in the training set.

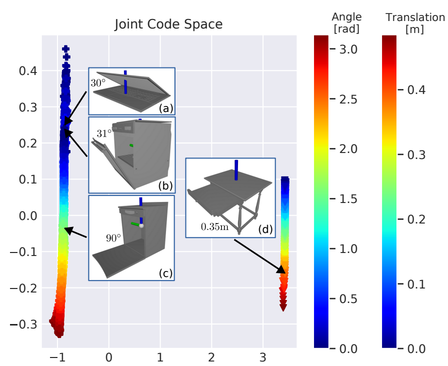

Figure 3. Intuition for Latent Space Regularization. Our main idea is that the joint codes of two similarly articulated objects should be close. We define the similarity first through the joint type *jt* and second through an exponential distance measure of the joint state q. Here, the laptop (a) and the oven (b) have a revolute joint and are similarly wide open around 30°. Compared to that, table (d) has a prismatic joint and thus should not be close to the revolute instances. Contrary to that, the dishwasher (c), has a revolute joint but is opened much more than the other revolute objects and therefore, should be relatively close. The visualization shows a lower dimensional projection of our learned latent joint space trained using our regularization.

3.2.2 Joint Decoder

As we represent the articulation state of the object implicitly through a joint code $\boldsymbol{z}_{\text{j}}$, we additionally introduce an articulation state decoder ϕ_{joint} to regress a discrete joint type $jt = \{\text{prismatic}, \text{revolute}\}$ and a continuous joint state q:

$$\phi_{\text{joint}}(\boldsymbol{z}_{\text{j}}) = \hat{jt}, \hat{q} \tag{3}$$

We use a multi-layer perceptron with 64 neurons in one hidden layer.

3.2.3 Backward Code Optimization

For the task of canonical object reconstruction from a set of SDF values at specific query points, we follow the optimization procedure from [24] to retrieve a shape and joint code. Different from [24], we utilize GPU parallelization to optimize multiple code hypotheses at once and pick the best one in the end. Additionally, we do not reset the codes as done in [24] but rather freeze them for some iterations. In early testing, we discovered that first optimizing for both codes jointly together gives a good initial guess. Freezing the joint code in a second stage of optimization helps such that the shape fits the static part and then last, freezing the shape code such that the joint code can do some fine adjustment to the articulation state of the object. To guide the gradient in the joint code space, we first transform the space using the singular value decomposition of the stacked training joint codes $\boldsymbol{z}_{\text{j}}^{n} \forall n \in 1, \ldots, N$. For further information, refer to Sec. S.2.

3.3. Training Protocol

To train CARTO, we first train the decoder as it provides ground truth training labels for the shape and joint code supervision of our encoder. Once shape and joint code labels are obtained for the objects in the dataset, we then train our encoder to predict the latent codes in addition to object pose. Thus, to adhere with our training procedure, we will first explain how to train our decoder and then our encoder.

3.3.1 Decoder

Given a training set of M objects each in N articulation states, we denote the m-th object in its n-th articulation state as $x_{m,n}$. As during training, a fixed association between each object and its latent codes is given, we can uniquely identify $x_{m,n} = (\boldsymbol{z}_{\text{s}}^{m}, \boldsymbol{z}_{\text{j}}^{n})$ as a tuple of both. This allows us to pass the gradient all the way to the codes themselves and thus, the embedding spaces rearrange them accordingly. Similar to [24], during training we regularize the codes through minimizing the L2-norm

$$\mathcal{L}_{\text{reg}}(\boldsymbol{z}) = \|\boldsymbol{z}\|, \tag{4}$$

where $\boldsymbol{z}$ is either $\boldsymbol{z}_{\text{s}}$ or $\boldsymbol{z}_{\text{j}}$.

The geometry decoder described in Sec. 3.2.1 is trained on a set of query points $\boldsymbol{x}$ close to the object surface sampled as in [28]. We define our reconstruction loss $\mathcal{L}_{\text{rec}}$ as in [28] but use a leaky clamping function

$$\text{clamp}_l(s|\delta, \alpha) = \begin{cases} s & |s| \leq \delta, \\ \alpha s & |s| > \delta \end{cases} \tag{5}$$

which is conceptually similar to a leaky ReLU by instead of hard clamping values above a threshold δ, we multiply it by a small factor α. Initial testing revealed a more stable training. Our reconstruction loss at one query point $\boldsymbol{x}$ is now given by:

$$\begin{aligned} &\mathcal{L}_{\text{rec}}\left(\boldsymbol{z}_{\text{s}}, \boldsymbol{z}_{\text{j}}, \boldsymbol{x}, s_{\boldsymbol{x}}\right) = \\ &\quad \left|\text{clamp}_l(\phi_{\text{geom}}(\boldsymbol{z}_{\text{s}}, \boldsymbol{z}_{\text{j}}, \boldsymbol{x})|\delta, \alpha) - \text{clamp}_l(s_{\boldsymbol{x}}|\delta, \alpha)\right|, \end{aligned} \tag{6}$$

where $s_{\boldsymbol{x}}$ is the ground truth distance to the surface.

The joint decoder introduced in Sec. 3.2.2 is jointly trained with the aforementioned geometry decoder. For the joint type loss $\mathcal{L}_{jt}$, we use cross entropy between the predicted joint type $\hat{jt}$ and ground truth jt and for the joint state loss $\mathcal{L}_q$ the L2-norm between the predicted joint state $\hat{q}$ and ground truth q.

Joint Space Regularization: One core contribution of our approach is how we impose structure in our joint code space during our decoder training. Here, we enforce the same similarity of latent codes as their corresponding articulations states have. A visualization of the underlying idea is shown in Fig. 3. Formally, given the joint codes $\boldsymbol{z}_{\text{j}}^k$ and $\boldsymbol{z}_{\text{j}}^l$ encoding two different articulation states $k, l \in 1, \ldots, N$, we define the similarity between them in latent space as

$$\text{sim}_{\text{latent}}\left(\boldsymbol{z}_{\text{j}}^k, \boldsymbol{z}_{\text{j}}^l\right) = \exp\left(-\frac{\|\boldsymbol{z}_{\text{j}}^k - \boldsymbol{z}_{\text{j}}^l\|}{\sigma}\right). \tag{7}$$

Similarly, we define the respective similarity in real joint space, considering the joint types jt^k and jt^l and the joint states q^k and q^l, through

$$\begin{aligned} &\text{sim}_{\text{real}}\left(\left(jt^k, q^k\right), \left(jt^l, q^l\right)\right) = \\ &\qquad \begin{cases} \exp\left(-\left(\frac{q^k - q^l}{\sigma_{jt}}\right)^2\right) & jt^k = jt^l \\ 0 & jt^k \neq jt^l, \end{cases} \end{aligned} \tag{8}$$

where σ_{jt} is a joint type specific scaling. By minimizing the L1-norm between both similarity measurements

$$\begin{aligned} &\mathcal{L}_{\text{jr}}\left(\boldsymbol{z}_{\text{j}}^k, \boldsymbol{z}_{\text{j}}^l\right) = \\ &\quad \left|\text{sim}_{\text{latent}}\left(\boldsymbol{z}_{\text{j}}^k, \boldsymbol{z}_{\text{j}}^l\right) - \text{sim}_{\text{real}}\left(\left(jt^k, q^k\right), \left(jt^l, q^l\right)\right)\right| \end{aligned} \tag{9}$$

we enforce that the latent similarities are similarly scaled as their real similarities.

We scale this formulation to all articulation states in the training set as described below. Calculating sim_{real} can be done once in a pre-processing step for all articulation state pairs $k, l \in 1, \ldots, N$ resulting in a matrix $\boldsymbol{S}_{\text{real}} \in \mathbb{R}^{N \times N}$. Similarly, calculating all $\text{sim}_{\text{latent}}$-pairs can be efficiently implemented as a vector-vector product. We denote the resulting matrix as $\boldsymbol{S}_{\text{latent}} \in \mathbb{R}^{N \times N}$. Eq. (9) now simplifies to

$$\mathcal{L}_{\text{jr}} = \frac{|\boldsymbol{S}_{\text{latent}} - \boldsymbol{S}_{\text{real}}|}{N^2}. \tag{10}$$

Through this efficient calculation, optimizing this loss term comes with almost no overhead during training. This concept of similarity can be extended for arbitrary kinematic graphs.

Pre-Training: Before we start training our full decoder, we only optimize our joint codes $\boldsymbol{z}_{\text{j}}^n \forall n \in 1, \ldots, N$. The pre-training helps with learning the full decoder as our joint codes are already more structured and thus, it is easier to learn the shape and joint code disentanglement. In the pre-training, we minimize

$$\mathcal{L}_{\text{pre}} = \delta_{\text{reg}, \boldsymbol{z}_{\text{j}}, \text{pre}} \mathcal{L}_{\text{reg}, \boldsymbol{z}_{\text{j}}} + \delta_{\text{jr,pre}} \mathcal{L}_{\text{jr}}, \tag{11}$$

where $\mathcal{L}_{\text{reg}, \boldsymbol{z}_{\text{j}}}$ is the default norm regularization from Eq. (4) and $\mathcal{L}_{\text{jr}}$ was introduced in Eq. (10).

Loss Function: Given an object $x_{m,n}$, we express our full decoder loss as

$$\begin{aligned} \mathcal{L} = &\delta_{\text{reg}, \boldsymbol{z}_{\text{s}}} \mathcal{L}_{\text{reg}, \boldsymbol{z}_{\text{s}}} + \delta_{\text{reg}, \boldsymbol{z}_{\text{j}}} \mathcal{L}_{\text{reg}, \boldsymbol{z}_{\text{j}}} + \delta_{\text{rec}} \mathcal{L}_{\text{rec}} \\ &+ \delta_{jt} \mathcal{L}_{jt} + \delta_q \mathcal{L}_q, \end{aligned} \tag{12}$$

where $\mathcal{L}_{\text{reg}, \boldsymbol{z}}$ are the shape and joint code regularization from Eq. (4), $\mathcal{L}_{\text{rec}}$ is the reconstruction loss introduced in Eq. (6), $\mathcal{L}_{jt}$ and $\mathcal{L}_q$ are the joint type and state loss. We jointly optimize $\mathcal{L}$ for the latent shape and joint code as well as the network parameters of the geometry decoder and joint decoder using ADAM [12] for 5000 epochs. Our new joint code regularizer loss $\mathcal{L}_{\text{jr}}$ introduced in Eq. (10) is minimized at the end of each epoch separately scaled by δ_{jr}. All δ variables are scalars to balance the different loss terms and are reported in Tab. S.1.

3.3.2 Encoder

Using [13] we generate a large-scale dataset in which we annotate each pixel with its respective ground truth value from the simulation as described in Sec. 3.1. For annotating the shape codes we directly use the results of our previous encoder training whereas for the joint code we use our inverse mapping explained in Sec. 3.3.3 to retrieve joint codes for arbitrary sampled articulation states.

3.3.3 Inverse Joint Decoder

To solve the inverse problem, given an articulation state for which we want to retrieve a joint code, we fit polynomial functions in the learnt joint code space. With the help of this mapping, we can retrieve arbitrary joint codes which then can be combined with a shape code to reconstruct objects in novel articulation states which have not been seen during the decoder training. Additionally, the mapping provides joint code training labels for the encoder.

We describe the full mapping as a function $\xi_{\text{code}}(jt, q) = \boldsymbol{z}_{\text{j}}$ that takes a joint type jt and joint state q as input and outputs a joint code $\boldsymbol{z}_{\text{j}}$. We leverage the fact that after decoder training, we learned a joint code $\boldsymbol{z}_{\text{j}}^{n}$ for each known training articulation state. We now define individual mappings for each joint type jt the following way. We will treat each latent dimension d separately. For each dimension d, we fit a polynomial function $\xi_{\text{code}}^{jt,d}(q)$ of varying degree p through all point tuples $(q^n, \boldsymbol{z}_{\text{j}}^{n}(d)) \forall n \in 1, \ldots, N$.

The final function

$$\xi_{\text{code}}(jt, q) = \begin{bmatrix} \xi_{\text{code}}^{jt,1}(q) \\ \vdots \\ \xi_{\text{code}}^{jt,D_j}(q) \end{bmatrix} \tag{13}$$

is then given by evaluating the polynomials individually and stacking the results into a vector. The exact choice of p is not important as long as the amount of joint codes to fit to is much higher than the potential dimensions of the polynomial $p \ll N$. Thus, we fixed $p = 5$ for all of our experiments. A visualization of our learned latent joint space and the fitted polynomials is given in Fig. S.3a.

4. Experiments

We conduct two main experiments, an object-centric canonical reconstruction task and a full scene reconstruction task. The first experiment is to evaluate the performance of our newly introduced decoder while the second experiment highlights the advantages of our single-forward pass method compared to a two-stage approach.

4.1. Object Set

For both experiments, we use 3D models from the PartNet-Mobility [38] object set to generate a training and test data set.

Categories: While PartNet-Mobility provides more than 2000 objects from 46 categories, as done in previous work [5, 9, 15, 24, 41] we only select a subset of all categories. From this subset of categories, we select objects with one fixed base part and one significant moving part (e.g. we filter out knobs, buttons etc.). To later create realistic room layouts, we further differentiate between three placement types for objects, stand-alone (SA), counter (C) and table-top (TT) objects. In Tab. 1 we list the number of objects per category we selected as well as the context in which they can be used.

Table 1. Overview of Selected Objects. We select a subset of the PartNet-Mobility [38] object set and report the amount of instances we selected per category for our training and test set. Our final object set has 92 objects instances for training and 25 for testing.

Category	Joint Type	SA	C	TT	Train	Test
Dishwasher	Rev.	✗	✓	✗	18	5
Laptop	Rev.	✗	✗	✓	20	5
Microwave	Rev.	✓	✗	✓	10	5
Oven	Rev.	✓	✓	✗	7	3
Refrigerator	Rev.	✓	✓	✗	10	2
Table	Pris.	✓	✗	✗	19	5
WashingMachine	Rev.	✓	✓	✗	8	2

SA = stand-alone, C = counter, TT = table-top
Rev. = Revolute, Pris. = Prismatic

Object Canonicalization: When tackling the task of reconstructing objects in a canonical object frame, usually, objects are canonicalized such that they either fit into the unit cube or unit sphere. This helps with the stability of learning and simplifies hyperparameter tuning. This approach fails for articulated objects as their outer dimensions change depending on the joint state. Blindly rescaling an articulated object such that it fits inside the unit cube or unit sphere results in an inconsistent part scaling across different joint states. To mitigate this problem, [15] proposed the NAOCS-space. Following their approach, first, we bring an object in its closed state (i.e. the lower joint limit) and put it in a canonical orientation such that for all objects, Z is pointing upwards and X back, Y is given through a right-hand coordinate frame system. Different from [15], we rescale and translate the closed object such that it fits in the unit cube and then backwards apply that same rescaling and translation to all objects of the same instance independent of the joint state of the object. It is important to note that rescaling an articulated object has no impact on revolute joint states (in $\deg$), but prismatic joint states (in m) which have to be rescaled accordingly.

4.2. Canonical Reconstruction Task

In our first experiment, we evaluate how well our decoders reconstruct the object's geometry and the articulation state. Thus, the task is not to reconstruct the object in the camera frame, but simply in its canonical frame. As described in Sec. 3.2.3, we will optimize the shape and joint code for each input SDF with ADAM [12] first jointly for 400 steps, then only the shape code for 100 steps and finally, only the joint code for 100 steps.

Dataset: To generate our dataset for the canonical reconstruction task, we first apply the aforementioned canonicalization to each object from our object set described in

Sec. 4.1. Here, the placement type does not matter. Then, we sample each object in 50 joint configurations uniformly spaced within the joint limits, make the meshes watertight using [6] and follow [28] to generate 100k ground truth SDF values. Lastly, we rescale the generated data by the largest extents across all meshes to a unit cube. As mentioned in Sec. 4.1 we also have to rescale prismatic joint states accordingly. While we do not consider this as a new dataset contribution, we make our generation code available to allow other researchers to adjust the selected object instances and categories and generate their own data.

Baselines and Ablations: Throughout this experiment, we will compare against the state-of-the-art for category-level object reconstruction method *A-SDF* [24]. As *A-SDF* is designed to work on a single category, first, to show that learning an implicit joint code does not have a negative impact rather than using the real joint state directly as input, we compare against *A-SDF* directly by training *CARTO* also only on a single category. Second, we will jointly train on all categories to highlight that *CARTO* is able to generalize to a wide variety of categories and articulations using one model. Third, we additionally perform an ablation study to understand the importance of our similarity regularization introduced in Sec. 3.3.1. In this ablation study, we remove the pre-training step and the post-epoch step. We call this model *CARTO-No-Enf*. And fourth, we extend *A-SDF* to also take the joint type as input which allows us to train it jointly on all categories.

Please note that we neglect *A-SDF*s proposed test-time adaption (TTA) technique as in real applications it would not be practical to keep network weights of all different instances encountered. Results using TTA are reported in Sec. S.5.1.

Metrics: To measure reconstruction quality, we report the bi-directional L2-Chamfer distance multiplied by 1000 between the ground truth points and the extracted points using the model's respective generation method. To quantify the articulation state prediction, we will report the joint type prediction accuracy as well as the joint state error measured in $\deg$ or m depending on the joint type for all correctly classified joint types.

Results: The results for the canonical reconstruction task are shown in Tab. 2. While no method clearly outperforms the other, it is notable though that CARTO and *A-SDF* trained on all categories are performing slightly better on average across all categories compared to our baselines. This shows that having disentangled joint and shape codes in CARTO can make the reconstruction category-agnostic.

4.3. Full Pipeline Task

In our second experiment, the full pipeline task, we want to investigate the advantages CARTO has over a two-stage

Table 2. Decoder Optimization Results. Each object is sampled in 50 different joint states for training as well as for testing. We average over all instances from our seven categories here. For a category-level comparison see Tab. S.3. † *means the model is trained only on one category, thus the joint type prediction is not applicable (N/A). The joint state error mean is only reported across the revolute categories, as there is only one prismatic category (reported in brackets).*

Method	CD (↓)	Joint State Error (↓)	Joint Type Accuracy (↑)
A-SDF [24] †	1.437	11.337° (0.094m)	N/A
CARTO †	1.190	12.474° (0.081m)	N/A
A-SDF [24]	**0.934**	16.139° (0.235m)	**0.962**
CARTO-No-Enf	2.246	35.892° (0.104m)	0.646
CARTO	1.192	**11.512°** (0.141m)	0.908

approach. To that end, we set up two experiments, one on simulated data and one on real-world data. For the full pipeline experiment, we use the trained decoders from the previous experiment.

Datasets: We evaluate on two datasets, quantitatively on a synthetic dataset that aligns with our synthetic training dataset and qualitatively on a newly collected real-world dataset.

Synthetic Data: For training our encoder, we use SimNet [13] to generate a large-scale dataset of indoor kitchen environments. We use the same articulated object instances from Tab. 1 we also used to train our decoders. Unlike the previous experiment, the placement type of the object matters here. For each randomly sampled articulated object in the scene, we randomly sample a joint state in its joint limits as well as a scale from a pre-defined scale for each category. To get ground truth joint codes for sampled articulation states we use the proposed method in Sec. 3.3.3. After sampling a scene, we generate noisy stereo images as well as non-noisy depth images (only used for evaluation of the baseline). To generate our synthetic test dataset we follow the same procedure with the only exception that we use the defined test-instances.

Real Data: Additionally, we evaluate the performance on a real-world test dataset we collected. Therefore, we select two real object instances from each of the following categories: knives, laptops, refrigerators, staplers, ovens, dishwashers, microwaves as well as one storage furniture and washing machine instance. We place these instances in common household environments. For each object, four different viewpoints were collected for each of the four articulation states for the object. We measure the real joint state and annotate the data using [30] with orientated 3D bounding boxes. In total, we collected 263 images. For collection we used a ZED 2 stereo camera. To get depth images we use state-of-the-art, off-the-shelf learned stereo depth methods to produce highly accurate depth images [31].

Comparison to other Datasets: To the best of our knowledge, the closest works to CARTO's dataset are the RBO [19] and BMVC [22] dataset. Both datasets do not provide large-scale synthetic stereo-RGB or RGB-D images and only half of the categories with readily available 3D furniture models from PartNetMobility. For a full comparison see Tab. S.2.

Baselines: We set up two baselines using A-SDF [24] and follow their proposed method to reconstruct objects in the camera frame. Since A-SDF assumes a given segmentation of the object as well as a pose that transforms the object from the camera frame to the object-centric frame, we will compare against two versions of A-SDF. One, where we use ground truth segmentation masks and poses which we call *A-SDF-GT* and one, where we use our model to predict segmentation masks whose center we then use to query our pixel-level pose map. We call this variant simply *A-SDF*. In both cases we approximate normals using the depth image, use the segmentation masks to extract the corresponding object point cloud from the depth image, transform the point clouds into the canonical object frame using the predicted pose, create SDF values, optimize for the SDF values as done in Sec. 4.2 and eventually reproject the reconstruction into the camera frame using the same transformation.

Metrics: We compare our reconstructions in the camera frame using two different metrics typically used for object pose prediction [35]. First, we compare the absolute error of the position and orientation by reporting the respective percentage below $10°10\mathrm{cm}$ and $20°30\mathrm{cm}$ combined. Second, we evaluate the average precision for various IOU-overlap thresholds (**IOU25** and **IOU50**) between the reconstructed bounding box and the ground truth bounding box. Both metrics serve as a proxy for articulation state and reconstruction quality. For evaluation results on these more fine-grained metrics, we refer to Sec. S.5.1.

Results: In Tab. 3 we report results using the aforementioned metrics as well as show a speed comparison of CARTO against the baselines.
Reconstruction: As visible in Tab. 3a *CARTO* shows superior performance over both variants of *A-SDF* for our full reconstruction task. Overall the performance of all methods is lower compared to similar experiments on category-level rigid object detection [35]. This can be attributed to the fact that in our kitchen scenarios we deal with heavy occlusions due to many objects being placed under the counter. Taking the occlusions into consideration, it becomes clear that for *A-SDF* it is very difficult to estimate the exact extents given only a front-showing partial point cloud of the object. Compared to that, *CARTO* benefits from its single-shot encoding step as the whole image is taken into consideration. We show qualitative results on our synthetic and real-world data in Fig. 1 as well as in Sec. S.5.3 where we also compare against an RGB-D version of CARTO.
Detection Speed: Aside from a lower pose and bounding box error, *CARTO* processes frames faster than *A-SDF*. Tab. 3b shows a reduction in inference time of more than 60 times while still persevering the same level of detail.

Table 3. Full Scene Reconstructions Results.

Method	IOU25 ↑	IOU50 ↑	$10°10\mathrm{cm}$ ↑	$20°30\mathrm{cm}$ ↑
A-SDF-GT	45.2	27.1	N/A	N/A
A-SDF	33.9	10.4	27.1	70.8
CARTO	**64.0**	**31.5**	**28.7**	**76.6**

(a) mAP Reconstruction Results.

Method	Sample Grid	Det.	Optim.	Recon.	Total
A-SDF	256	5.390	21.600	7.836	64.262
CARTO	256	**0.264**	N/A	0.414	1.092
CARTO	128	**0.264**	N/A	**0.097**	**0.458**

(b) Detection Speed of Approaches in [s]. We measure the speed of our approaches on a common desktop using a Nvidia Titan XP GPU. Sample grid defines how many points are sampled along each dimension. Total time assumes two detected objects. *Det. = Detection time per image, Optim. = Optimiziation time per object, Recon. = Reconstruction time per object*

5. Conclusion

We presented a novel method to reconstruct multiple articulated objects in a scene in a category- and joint-agnostic manner from a single stereo image. For reconstruction we learn a SDF-based decoder and show the necessity of regularization to achieve good performance. Our full single-shot pipeline improves over current two-stage approaches in terms of 3DIoU and inference speed.

Limitations: While CARTO is able to generalize to unseen instances, it still relies on a learned shape prior. Using test time adaption techniques such as done by [24] helps mitigating this issues, but is not sufficient to deal with categorically different objects. Additionally, while the single-forward pass is fast, jointly optimizing for pose, scale, codes like done in [8,18] could further improve results with the cost of added execution time.

Currently CARTO is only trained on objects with a single joint. To extend CARTO to objects with an arbitrary number of joints, we must be able to calculate pairwise similarity between two object states. While not explored in this paper, CARTO introduces a framework for future research to pursue this research question. A potential solution could leverage Eq. (8) and Hungarian matching of the cross-product of articulation states to obtain similarities measurements between arbitrary kinematic structures.

Acknowledgements: This work was partly funded by the Carl Zeiss Foundation with the ReScaLe project.

References

[1] Aljaz Bozic, Michael Zollhofer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d

reconstruction with semi-supervised data. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7002–7012, 2020. 2

[2] Romain Brégier. Deep regression on manifolds: a 3D rotation case study. In *Proc. Conf. 3D Vis.*, pages 166–174, 2021. 3

[3] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3D object reconstruction from a single image. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 605–613, 2017. 2

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2961–2969, 2017. 1

[5] Nick Heppert, Toki Migimatsu, Brent Yi, Claire Chen, and Jeannette Bohg. Category-independent articulated object tracking with factor graphs. In *Proc. IEEE Int. Conf. on Intel. Rob. and Syst.*, pages 3800–3807, 2022. 1, 2, 6

[6] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698*, 2018. 7

[7] Muhammad Zubair Irshad, Thomas Kollar, Michael Laskey, Kevin Stone, and Zsolt Kira. Centersnap: Single-shot multi-object 3d shape reconstruction and categorical 6d pose and size estimation. In *Proc. IEEE Int. Conf. on Rob. and Auto.*, pages 10632–10640, 2022. 1, 2, 3

[8] Muhammad Zubair Irshad, Sergey Zakharov, Rares Ambrus, Thomas Kollar, Zsolt Kira, and Adrien Gaidon. Shapo: Implicit representations for multi-object shape, appearance, and pose optimization. In *Proc. Springer Eur. Conf. Comput. Vis.*, pages 275–292, 2022. 1, 3, 8

[9] Ajinkya Jain, Rudolf Lioutikov, Caleb Chuck, and Scott Niekum. Screwnet: Category-independent articulation model estimation from depth images using screw theory. In *Proc. IEEE Int. Conf. on Rob. and Auto.*, pages 13670–13677, 2021. 1, 2, 6

[10] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5616–5626, 2022. 1, 2, 3

[11] Zhenyu Jiang, Yifeng Zhu, Maxwell Svetlik, Kuan Fang, and Yuke Zhu. Synergies between affordance and geometry: 6-Dof grasp detection via implicit representations. In *Proc. Rob.: Sci. and Syst.*, 2021. 1

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5, 6

[13] Thomas Kollar, Michael Laskey, Kevin Stone, Brijen Thananjeyan, and Mark Tjersland. Simnet: Enabling robust unknown object manipulation from pure synthetic data via stereo. In *Proc. Conf. on Rob. Learn.*, pages 938–948, 2021. 1, 3, 4, 5, 7

[14] Jiahui Lei and Kostas Daniilidis. Cadex: Learning canonical deformation coordinate space for dynamic surface representation via neural homeomorphism. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6624–6634, 2022. 2

[15] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3706–3715, 2020. 3, 6

[16] Liu Liu, Wenqiang Xu, Haoyuan Fu, Sucheng Qian, Qiaojun Yu, Yang Han, and Cewu Lu. Akb-48: a real-world articulated object knowledge base. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 14809–14818, 2022. 1, 3

[17] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2019–2028, 2020. 2

[18] Xingyu Liu, Gu Wang, Yi Li, and Xiangyang Ji. Catre: Iterative point clouds alignment for category-level object pose refinement. In *Proc. Springer Eur. Conf. Comput. Vis.*, pages 499–516, 2022. 8

[19] Roberto Martín-Martín, Clemens Eppner, and Oliver Brock. The RBO Dataset of Articulated Objects and Interactions, June 2018. 7

[20] Oier Mees, Maxim Tatarchenko, Thomas Brox, and Wolfram Burgard. Self-supervised 3d shape and viewpoint estimation from single images for robotics. In *Proc. IEEE Int. Conf. on Intel. Rob. and Syst.*, pages 6083–6089, 2019. 1

[21] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4460–4470, 2019. 2, 3

[22] Frank Michel, Alexander Krull, Eric Brachmann, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Pose estimation of kinematic chain instances via object coordinate regression. In *Proc. Brit. Mach. Vis. Conf.*, pages 181–1, 2015. 3, 8

[23] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Comm. of ACM*, 65(1):99–106, 2021. 2

[24] Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-SDF: Learning disentangled signed distance functions for articulated shape representation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 13001–13011, 2021. 1, 2, 3, 4, 6, 7, 8

[25] Neil Nie, Samir Yitzhak Gadre, Kiana Ehsani, and Shuran Song. Structure from action: Learning interactions for articulated object 3d structure discovery. *arXiv preprint arXiv:2207.08997*, 2022. 2

[26] Atsuhiro Noguchi, Umar Iqbal, Jonathan Tremblay, Tatsuya Harada, and Orazio Gallo. Watch it move: Unsupervised discovery of 3d joints for re-posing of articulated objects. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3677–3687, 2022. 1, 2

[27] Pablo Palafox, Nikolaos Sarafianos, Tony Tung, and Angela Dai. Spams: Structured implicit parametric models. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12851–12860, 2022. 2

[28] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 165–174, 2019. 2, 3, 5, 7

[29] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Proc. Springer Eur. Conf. Comput. Vis.*, pages 523–540, 2020. 2

[30] Christoph Sager, Patrick Zschech, and Niklas Kuhl. labelCloud: A lightweight labeling tool for domain-agnostic 3d object detection in point clouds. *Computer-Aided Design and Applications*, 19(6):1191–1206, 2022. 7

[31] Krishna Shankar, Mark Tjersland, Jeremy Ma, Kevin Stone, and Max Bajracharya. A learned stereo depth system for robotic manipulation in homes. *IEEE Rob. and Auto. Let.*, 7(2):2305–2312, 2022. 7

[32] Jürgen Sturm, Cyrill Stachniss, and Wolfram Burgard. A probabilistic framework for learning kinematic models of articulated objects. *J. Art. Intel. Res.*, 41:477–526, 2011. 1, 2

[33] Shih-Yang Su, Frank Yu, Michael Zollhoefer, and Helge Rhodin. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. In *Proc. Adv. Neural Inform. Process. Syst.*, 2021. 2

[34] Wei-Cheng Tseng, Hung-Ju Liao, Lin Yen-Chen, and Min Sun. CLA-NeRF: Category-Level Articulated Neural Radiance Field. In *Proc. IEEE Int. Conf. on Rob. and Auto.*, pages 8454–8460, 2022. 2

[35] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6D object pose and size estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2637–2646, 2019. 1, 3, 8

[36] Fangyin Wei, Rohan Chabra, Lingni Ma, Christoph Lassner, Michael Zollhofer, Szymon Rusinkiewicz, Chris Sweeney, Richard Newcombe, and Mira Slavcheva. Self-Supervised Neural Articulated Shape and Appearance Models. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 15816–15826, 2022. 2

[37] Yijia Weng, He Wang, Qiang Zhou, Yuzhe Qin, Yueqi Duan, Qingnan Fan, Baoquan Chen, Hao Su, and Leonidas J. Guibas. CAPTRA: CAtegory-level Pose Tracking for Rigid and Articulated Objects from Point Clouds. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 13209–13218, 2021. 1, 2

[38] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11097–11107, June 2020. 6

[39] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Comput. Graph. Forum*, 41(2):641–676, 2022. 2

[40] Xianghao Xu, Yifan Ruan, Srinath Sridhar, and Daniel Ritchie. Unsupervised kinematic motion detection for part-segmented 3d shape collections. In *Proc. Spec. Inter. Group CG. Inter. Tech. Conf.*, pages 1–9, Vancouver BC Canada, 2022. 2

[41] Han Xue, Liu Liu, Wenqiang Xu, Haoyuan Fu, and Cewu Lu. OMAD: Object Model with Articulated Deformations for Pose Estimation and Retrieval. In *Proc. Brit. Mach. Vis. Conf.*, page 366, 2021. 6

[42] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 15975–15984, 2021. 2

[43] Sergey Zakharov, Rares Andrei Ambrus, Katherine Liu, and Adrien Gaidon. Road: Learning an implicit recursive octree auto-decoder to efficiently encode 3d shapes. In *Proc. Conf. on Rob. Learn.*, 2022. 2

[44] Cheng Zhang, Zhaopeng Cui, Yinda Zhang, Bing Zeng, Marc Pollefeys, and Shuaicheng Liu. Holistic 3D scene understanding from a single image with implicit representation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8833–8842, 2021. 1