A-SDF: Learning Disentangled Signed Distance Functions for Articulated Shape Representation

Jiteng Mu¹, Weichao Qiu², Adam Kortylewski², Alan Yuille², Nuno Vasconcelos¹, Xiaolong Wang¹
¹UC San Diego, ²Johns Hopkins University

Abstract

Recent work has made significant progress on using implicit functions, as a continuous representation for 3D rigid object shape reconstruction. However, much less effort has been devoted to modeling general articulated objects. Compared to rigid objects, articulated objects have higher degrees of freedom, which makes it hard to generalize to unseen shapes. To deal with the large shape variance, we introduce Articulated Signed Distance Functions (A-SDF) to represent articulated shapes with a disentangled latent space, where we have separate codes for encoding shape and articulation. With this disentangled continuous representation, we demonstrate that we can control the articulation input and animate unseen instances with unseen joint angles. Furthermore, we propose a Test-Time Adaptation inference algorithm to adjust our model during inference. We demonstrate our model generalize well to out-of-distribution and unseen data, e.g., partial point clouds and real-world depth images.

1. Introduction

Modeling articulated objects has wide applications in multiple fields including virtual and augmented reality, object functional understanding, and robotic manipulation. To understand articulated objects, recent works propose to train deep networks for estimating per-part poses and the joint angle parameters of an object instance in a known category [19, 45]. However, if we want to interact with the articulated object (e.g., open a laptop), estimating its static state is not sufficient. For example, an autonomous agent needs to predict what the articulated object shape will be like after interactions for planning its action.

In this paper, we introduce Articulated Signed Distance Functions (A-SDF), a differentiable category-level articulated object representation, which can reconstruct and predict the object 3D shape under different articulations. A differentiable model is useful in applications which require back-propagation through the model to adjust inputs, such as rendering in graphics and model-based control in robotics.

We build our articulated object model based on the deep implicit Signed Distance Functions [30]. While implicit functions have recently been widely applied in modeling static object shape with fine details [34, 35, 38], much less effort has been devoted to modeling general articulated objects. We observe that models with a single shape code input, such as DeepSDF [30], cannot encode the articulation variation reliably. It is even harder for the models to generalize to unseen instances with unseen joint angles.

To improve the generalization ability, we propose to model the joint angles explicitly for articulated objects. Instead of using a single code to encode all the variance, we propose to use one shape code to model the shape of object parts and a separate articulation code for the joint angles. To achieve this, we design two separate networks in our model: (i) a shape encoder to produce a shape embedding given a shape code input; (ii) an articulation network which takes input both the shape embedding and an articulation code to deform the object shape. During training, we use the ground-truth joint angles as inputs and learn the shape code jointly with both model parameters. To enable the disentanglement, we enforce the same instance with different joint angles to share the same shape code.

During inference, given an unseen instance with unknown articulation, we first infer the shape code and articulation code via back-propagation. Given the inferred shape code, we can simply adjust the articulation code to generate the instance at different articulations. Note the part geometry remains the same as we fix the inferred shape code during generation. To generalize our model to out-of-distribution and unseen data, e.g., partial point clouds and real-world depth images, we further propose a Test-Time Adaptation (TTA) approach to adjust our model during inference.

2. Related Work

Neural Shape Representation. A large body of work [43, 10, 4, 6, 20, 33] has focused on investigating
efficient and accurate 3D object representations. Recent advances suggest that representing 3D objects as continuous and differentiable implicit functions [9, 30, 23, 5, 16] can model various topologies in a memory-efficient way. Most of these work is limited to modeling static objects and scenes [9, 12, 46, 38, 26, 34, 24, 37, 41]. Different from previous works, our method models articulated objects in a category-level by learning a disentangled implicit representation and we test our model on real depth images.

Articulated Humans. One line of work leverages parametric mesh models [21, 18, 51, 2] to estimate shape and articulation for faces [39, 32, 36], hands[8], humans bodies [31, 1, 48, 15, 13, 28, 47, 42], and animals [50, 14, 17, 49] by directly inferring shape and articulation parameters. However, such parametric models requires substantial efforts from experts to construct and thus is hard to generalize to large-scale object categories. To address the challenge, another line of work [29, 25, 40, 7, 3, 27] employs neural networks to learn shapes from data. In comparison, our method is category-level on general articulated objects and we assume no part label.

3. Method

We propose Articulated Signed Distance Functions (A-SDF), a differentiable category-level articulated object representation to reconstruct and predict the object 3D shape under different articulations. Our model takes sampled 3D point locations, shape codes, and articulation codes as inputs, and outputs SDF values (signed distance) that measure the distance of a point to the closest surface point. The key insight is that all shape codes of the same instance should be identical, independent of its articulation.

3.1. Formulation

Consider a training set of $N$ instance models for one object category. Each instance is articulated into $M$ poses, leading to a training set of $N \times M$ shapes of the category. Let $X_{n,m}$ denote the shape articulated from instance $n$ with articulation $m$, where $n \in \{1, \ldots, N\}$, $m \in \{1, \ldots, M\}$. Each shape $X_{n,m}$ is assigned with a shape code $\phi_n \in \mathbb{R}^C$, where $C$ denotes the latent dimension, and an articulation code $\psi_m \in \mathbb{R}^D$ with $D$ denoting the number of DoFs. The shape code $\phi_n$ is shared across the same object instance $n$ across different articulations. During training, we maintain and update one shape code for each instance. We use joint angles to represent the articulation code. For example, the articulation code of a 2-DoF object (e.g., eyeglasses) with both joints articulated to 45$^\circ$ is $\psi_m = (45^\circ, 45^\circ)$. The joint angle is defined as a relative angle to the canonical pose of the object.

Let $x \in \mathbb{R}^3$ be a sampled point from a shape. For notational simplicity, we omit the subscripts and denote $\phi$ and $\psi$ as the corresponding shape and articulation code of the shape. As shown in Figure 1, an Articulated Signed Distance Function $f_\theta$ is finally defined with the auto-decoder architecture, which is composed of a shape encoder $f_s$ and an articulation network $f_a$,

$$f_\theta(x, \phi, \psi) = f_s(x, \phi), x, \psi = s,$$  

where $s \in \mathbb{R}$ is a scalar SDF value (the signed distance to the 3D surface). The sign of the SDF value indicates whether the point is inside (negative) or outside (positive) the watertight surface.

3.2. Training

During training, given the ground-truth articulation code $\psi$, sampled points and their corresponding SDF values, the model is trained to optimize the shape code $\phi$ and the model parameters $\theta$.

The training process is illustrated in Figure 1. The shape code is first concatenated with a sampled point $x$ to form vector of dimension $C + 3$ and input to the shape encoder. Then the articulation network takes the shape embedding and articulation code to predict the SDF value for the input 3D point. When part supervision is available, a linear classifier is added to the last hidden layer of the articulation network to simultaneously output the part label.

The training loss functions are defined as following. Let $K$ be the number of sampled points per shape. The function $f_\theta$ is trained with the per-point $L_1$ loss function to regress SDF values,

$$L^s(\mathcal{X}, \phi, \psi) = \frac{1}{K} \sum_{k=1}^{K} \left\| f_\theta(x_k, \phi, \psi) - s_k \right\|_1,$$

where $x_k \in \mathcal{X}$ is a point of instance $\mathcal{X}$, $s_k$ the corresponding ground-truth SDF value, and $k \in \{1, \ldots, K\}$. When the object part labels are available, we include a complementary auxiliary part classification loss using cross-entropy.

The full loss $\mathcal{L}(x, \phi, \psi)$ is defined as,

$$\mathcal{L}(\mathcal{X}, \phi, \psi) = \mathcal{L}^s(\mathcal{X}, \phi, \psi) + \lambda_\phi ||\phi||_2^2.$$  

Following [30], we include a zero-mean multivariate-Gaussian prior per shape latent code $\phi$ to facilitate learning a continuous shape manifold.

At training time, the shape codes are randomly initialized with a Gaussian distribution at the very beginning of training. The articulation codes are constants given from the ground-truths. The objective is to optimize the loss function over all $N \times M$ training shapes, defined as follows,

$$\arg \min_{\theta, \phi_n} \sum_{n=1}^{N} \sum_{m=1}^{M} \mathcal{L}(X_{n,m}, \phi_n, \psi_m),$$

where $\theta$ is the network parameters.
3.3. Inference

Basic Inference. In the inference stage, illustrated in the Inference Section of Figure 1, an instance $X$ is given and the goal is to recover the corresponding shape code $\phi$ and the articulation code $\psi$. This can be done by back-propagation. The two codes are initialized randomly, the articulation network parameters are fixed, and the codes are inferred jointly by solving the optimization with the following objective,

$$\hat{\phi}, \hat{\psi} = \arg\min_{\phi, \psi} L(X, \phi, \psi).$$

(5)

We first use Equation 5 to optimize both shape and articulation codes as our initial estimation. So the estimated articulation code $\hat{\psi}$ is then kept and the shape code is discarded. In the second step, the shape code is re-initialized, the articulation code is fixed to $\hat{\psi}$, and the optimization is only solved for the shape code $\hat{\phi}$.

Test-Time Adaptation Inference. To generalize better to out-of-distribution data, the Test-Time Adaptation (TTA) for shape encoder $f_s$ is further introduced. It is built on the basic inference procedure with the estimated shape code $\hat{\phi}$ and articulation code $\hat{\psi}$. We fix both estimated codes and finetune the shape encoder $f_s$ using the following objective,

$$\hat{f}_s = \arg\min_{f_s} L(X, \hat{\phi}, \hat{\psi}),$$

(6)

where $\hat{\phi}$ and $\hat{\psi}$ are obtained as described in the basic inference. Note that our proposed model architecture is the key for TTA. The separation of shape encoder and articulation network ensures the disentanglement is maintained when the shape encoder is finetuned.

3.4. Articulated Shape Synthesis

A main advantage of the proposed disentangled continuous representation is that, once a shape code is inferred, it can be applied to synthesize shapes of unseen instances with unseen joint angles, by simply varying the articulation code.

4. Experiment

4.1. Datasets

For all experiments, the mesh models used are from the Shape2Motion dataset [44]. Shape2Motion is a large scale 3D articulated object dataset containing 2,440 instances. We select seven categories with sufficient number of instances per category, which are laptop, stapler, washing machine, door, oven, eyeglasses, and fridges.

4.2. Shape Synthesis and Part Prediction

One main advantage of our learned disentangled representation is its generation ability. We can easily control the articulation input to generate corresponding shapes of unseen instances with unseen joint angles. In this section, we
study the quality of generated shapes using the proposed generation method in Section 3.4.

Since DeepSDF does not have the ability to generate shapes, to provide comparisons, we employ the DeepSDF Interpolation results as baseline. Given two shapes, the target shape code is simply computed as a linear combination of the two inferred latent codes. Note that this is not a fair comparison as our method requires only one shape instead of two as for the baseline. Though relying on less information, the proposed method still yields much better results as shown in Table 1. We demonstrate that applying Test-Time Adaption reduces the error further, indicating that Test-Time Adaption helps with inferring better shape while maintaining a disentangled representation.

One additional advantage of the proposed method is that joint angles can be estimated simultaneously. We quantitatively evaluate joint angle prediction errors in degrees, as shown in brackets in Table 1. Results suggest that the proposed model can predict joint angles accurately during the inference stage. We also demonstrate that, if provided, part labels can further boost the performance. Models trained with part labels are denoted as Ours + part labels.

**4.3. Test on Real-world Depth Images**

We quantitatively show the proposed method generalizes better on real-world depth images, as shown in Table 2. The RBO dataset [22] is a collection of 358 RGB-D video sequences of humans manipulating articulated objects, with the ground-truth poses of the rigid parts annotated by a motion capture system. We take laptop depth images from different sequences in the dataset and crop laptops from depth images. We generate corresponding point clouds as the ground-truth to evaluate the generation performance. As visualized in Fig 2, the proposed model reliably synthesizes shapes at unseen articulation whereas DeepSDF does not have the ability to generate shapes. Table 2 results suggest that applying Test-Time Adaption reduces the error further on both reconstruction and generation.

<table>
<thead>
<tr>
<th></th>
<th>Laptop</th>
<th>Stapler</th>
<th>Washing</th>
<th>Door</th>
<th>Oven</th>
<th>Eyeglasses</th>
<th>Fridge</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepSDF [30] (Interpolation)</td>
<td>2.77</td>
<td>8.69</td>
<td>8.04</td>
<td>7.79</td>
<td>11.13</td>
<td>3.33</td>
<td>1.74</td>
</tr>
<tr>
<td>Ours (w/o TTA)</td>
<td>0.39 (1.39)</td>
<td>3.77 (3.30)</td>
<td>0.73 (1.09)</td>
<td>3.77 (7.08)</td>
<td>2.48 (2.58)</td>
<td>0.97 (3.47)</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>0.32 (1.59)</td>
<td>3.25 (3.53)</td>
<td>3.01 (8.44)</td>
<td>0.53 (0.95)</td>
<td>2.58 (6.79)</td>
<td>2.42 (2.84)</td>
<td>0.86 (4.19)</td>
</tr>
<tr>
<td>Ours (w/o TTA) + part label</td>
<td>0.32 (1.45)</td>
<td>3.08 (3.66)</td>
<td>2.16 (2.66)</td>
<td>0.38 (1.04)</td>
<td>5.19 (3.20)</td>
<td>2.03 (2.12)</td>
<td>0.85 (3.69)</td>
</tr>
<tr>
<td>Ours + part label</td>
<td>0.29 (1.48)</td>
<td>2.48 (3.34)</td>
<td>1.96 (2.03)</td>
<td>0.33 (1.67)</td>
<td>3.10 (2.98)</td>
<td>2.16 (2.18)</td>
<td>0.64 (2.98)</td>
</tr>
</tbody>
</table>

Table 1: Chamfer-L1 distance comparison for shape synthesis. Joint angle estimation errors of the proposed method in brackets (·).

<table>
<thead>
<tr>
<th></th>
<th>Reconstruction</th>
<th>Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepSDF [30]</td>
<td>4.65</td>
<td>-</td>
</tr>
<tr>
<td>Ours (w/o TTA)</td>
<td>2.53</td>
<td>5.09</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.76</strong></td>
<td><strong>3.22</strong></td>
</tr>
</tbody>
</table>

Table 2: Chamfer-L1 distance comparison on real-world depth images. The Chamfer-L1 distance here is from ground-truth depth to reconstructed shape. DeepSDF is not able to generate new shapes.

Note that the Ours generation are generated by only changing the articulation code. The shape code is inferred from the input depth of a laptop at a different articulation. RGB images and joint angles shown are only for visualization purposes and are not input to the model.

Figure 2: Test on real-world depth images. From left to right: Input depth, DeepSDF reconstruction, Ours reconstruction and generation. Note that the Ours generation are generated by only changing the articulation code. The shape code is inferred from the input depth of a laptop at a different articulation. RGB images and joint angles shown are only for visualization purposes and are not input to the model.
References


[22] Roberto Martín-Martín, Clemens Eppner, and Oliver Brock. The RBO dataset of articulated objects and interactions. Int. J. Robotics Res., 38(9), 2019. 4


[51] Silvia Zuffi, Angjoo Kanazawa, David W. Jacobs, and Michael J. Black. 3d menagerie: Modeling the 3d shape and pose of animals. In CVPR, pages 5524–5532, 2017. 2