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

Inputs
Shape Space
Joint Angle
Laptop
Eyeglasses
Stapler
(87°) (45°) (78°) (9°)
(45°, 15°) (5°, 45°) (45°, 45°) (5°, 5°)
(15°) (-9°) (-39°) (-60°)

MLP
45°
80°
63°
90°
MLP
Generated shapes at unseen articulations

Figure 1: We represent articulated objects with separate codes for encoding shape and articulation. During inference, given an unseen instance, our model first infers the shape and articulation codes via back-propagation. With the inferred shape code, we can generate shapes at unseen angles by only changing the articulation code.

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. Project page with code: https://jitengmu.github.io/A-SDF/.

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 [40, 82]. 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 requiring 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 [58]. While implicit functions have recently been widely applied in modeling static object shape with fine details [64, 65, 72], much less effort has been devoted to modeling general articulated objects. We observe that models with a single shape code input, such as DeepSDF [58], 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 embedding network 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 encourage 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. We visualize the generation process and results for a few objects in Figure 1. 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. Note that our unique model architecture with separate shape embedding and articulation network provides the opportunity to do so: the separation of shape embedding and articulation network ensures the disentanglement is maintained when the shape embedding network is adapted. We adapt the shape embedding network to the current test instance by updating its parameters, while fixing the parameters of the articulation network. This procedure allows A-SDF to reconstruct and generate better shapes aligning with the inputs.

To our knowledge, our work is the first paper tackling the problem of generic articulated object synthesis in the implicit representation context. We summarize the contributions of our paper as follows. First, we propose Articulated Signed Distance Functions (A-SDF) and a Test-Time Adaptation inference algorithm to model daily articulated objects. Second, the disentangled continuous representation allows us to control the articulation code and generate corresponding shapes as output on unseen instances with unseen joint angles. Third, the proposed representation shows significant improvement on interpolation, extrapolation, and shape synthesis. More interestingly, our model can generalize to real-world depth images from the RBO dataset [45] and we quantitatively demonstrate superior performance over the baselines.

2. Related Work

**Neural Shape Representation.** A large body of work [80, 19, 9, 11, 41, 63, 17, 78, 77, 89, 13, 1, 15] has focused on investigating efficient and accurate 3D object representations. Recent advances suggest that representing 3D objects as continuous and differentiable implicit functions [18, 58, 46, 10, 35, 26, 84, 72, 54, 64, 48, 71, 76] can model various topologies in a memory-efficient way. The basic idea is to exploit neural networks to parameterize a shape as a decision boundary in 3D. Most of these work is limited to modeling static objects and scenes [18, 26, 84, 72, 54, 64, 48, 71, 76]. 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. Comparisons to implicit neural networks on deformable shapes will be discussed in depth in the following.

**Articulated Humans.** One line of work leverages parametric mesh models [43, 38, 92, 5] to estimate shape and articulation for faces [74, 62, 66], hands [16], humans bodies [60, 4, 86, 34, 28, 56, 85, 79], and animals [91, 29, 36, 90] by directly inferring shape and articulation parameters. However, such parametric models require substantial efforts from experts to construct and is hard to generalize to large-scale object categories. To address the challenges, another line of work [57, 53, 75, 12, 8, 55] employs neural networks to learn shapes from data. For example, Niemeyer et al. [53] learned an implicit vector field and deformed shapes in a spatial-temporal space. However, the design does not allow for controlling each part separately. Recently, Hoang et al. [75] defined shapes using patches and shapes can be deformed by manipulating the extrinsic parameters of each patch. Nevertheless, the learned patches do not correspond to parts and the method fails at large deformation. Deng et al. [12] modeled each human body part by a separate implicit function. However, the method is limited to instance-level and requires skinning weights for the learning process. In comparison, our method is category-level on general articulated objects and we assume no part label. Therefore, these previous approaches are not directly comparable to ours. Besides, we model articulated object poses with joint angles, which allows us to articulate each joint separately.

**Articulated General Objects.** Though reconstructing articulated humans has attracted lots of attention in the community, modeling 3D shapes of daily articulated objects is an under-explored field in terms of both data and approaches. Unlike modeling humans where lots of human priors are injected, modeling articulated daily objects poses additional challenges by assuming no prior part geometry and labels, articulation status, joint type, joint axis and joint location about the category. One recent popular paradigm of research [20, 31, 44, 14, 47, 82] focuses on estimating 6D poses of articulated objects. For example, Desingh et al. [14] proposed a instance-level factored approach to estimate the poses of articulated objects with articulation constraints. However, 6D pose information may not be suffi-
cient for tasks that require detailed shape information, such as robotic manipulation [32, 83, 50, 49]. In this work, we show that implicit functions are suitable for the daily articulated objects modeling. We demonstrate that once an implicit function is learned, shapes at unseen articulations can be generated by manipulating the articulation code.

Disentangled Representation. Disentangled representations focus on modeling complex variations in a low dimensional space, where individual factors control different types of variation. Previous work [37, 22, 7, 24, 33, 30, 23, 52, 67, 6, 88, 87, 68, 61, 42, 2, 59] has shown that disentangled representations are essential to learn meaningful latent space for 2D image synthesis. For example, Zhou et al. [87] proposed an auto-encoder architecture to disentangle human pose and shape. Different from previous work, we focus on modeling general articulated objects in 3D with interpretable pose codes.

Adaptation on Test Instance. Learning on test instance has been recently applied for adapting a trained model to out-of-distribution in multiple applications, including image recognition [25, 51, 73], super-resolution and synthesis [70, 69, 3], and mesh reconstruction and generation [39, 27]. For example, Li et al. [39] exploits training in test-time with self-supervision for consistent mesh reconstruction in a single video. Inspired by previous works, we develop a Test-Time Adaptation inference algorithm in implicit functions for better shape reconstruction and generation, where our disentanglement-based model architecture is the key to allow for Test-Time Adaptation.

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. We argue that, even though the shapes look quite different for different articulations of the same instance, a good representation should capture this variability in a low dimensional space, since the part geometry remains unchanged. An overview of our method is presented in Figure 2. As different categories vary a lot in terms of articulation, we train separate networks for each category.

Our model is based on DeepSDF [58]. DeepSDF is an effective and widely acknowledged baseline and its simple network design allows us to focus on the effectiveness of our key idea rather than designing complex architectures. Note that our approach is generic and it can be applied to other advanced architectures as well. For a fair comparison, our model is designed with similar model size as DeepSDF. Furthermore, compared to feed forward designs [46, 64], the optimization-based shape modeling is naturally compatible with Test-Time Adaptation to address the out-of-distribution data as described in Section 3.3.

In the following, we describe how to learn a model encouraging the disentanglement of shape and articulation. We also introduce a Test-Time Adaptation inference technique allowing for generating unseen shapes at unseen articulations with high quality.

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 2, an Articulated Signed Distance Function $f_\theta$ is finally defined with the auto-decoder architecture, which is composed of a shape embedding network $f_s$ and an articulation network $f_\alpha$.

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

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. The 3D shape is implicitly represented by the zero level-set $f_\theta(\cdot) = 0$.

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 2. The shape code is first concatenated with a sampled point $x$ to form vector of dimension $C+3$ and input to the shape embedding network. Similarly, the articulation code $\psi$ (joint angles) is
Figure 2: Overview of the proposed method. At training time, the articulation code and randomly sampled shape codes are first concatenated with a sampled point separately. The produced embeddings are then input to an articulated signed distance function \( f_\theta \) to regress SDF values (signed distance) and predict part labels (optional). Note that the same instance is associated with one shape code regardless of its articulation state, as illustrated with brown staplers. During inference, back-propagation is used to jointly infer the shape code and articulation code for an unseen instance. With the inferred shape code, the model can faithfully generate new shape at unseen articulations.

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,

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

where \( x_k \in X \) is a point of instance \( X \), \( s_k \) 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 as,

\[
\mathcal{L}_p(X, \phi, \psi) = \frac{1}{K} \sum_{k=1}^{K} \left\{ CE \left( f_\theta(x_k, \phi, \psi), p_k \right) \right\},
\]

where \( CE \) denotes the cross-entropy loss and \( p_k \) is the ground-truth label for the part containing \( x_k \). Intuitively, the part classification task helps the network disambiguate the object parts.

The full loss \( L(x, \phi, \psi) \) is defined as,

\[
L(X, \phi, \psi) = \mathcal{L}_1(X, \phi, \psi) + \lambda_p \mathcal{L}_p(X, \phi, \psi) + \lambda_\phi \| \phi \|^2_2.
\]

Following [58], we include a zero-mean multivariate-Gaussian prior per shape latent code \( \phi \) to facilitate learning a continuous shape manifold. Part coefficient \( \lambda_p \) and shape code coefficient \( \lambda_\phi \) are introduced to balance different losses. We optionally set \( \lambda_p = 0 \) when the part labels are not available.

At training time, the shape codes are randomly initialized with a Gaussian distribution at the very beginning of training. Each shape code is then optimized through the training steps and it is shared across all the shapes articulated from the same instance. 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,
\[
\arg \min_{\theta, \phi, \psi} \sum_{n=1}^{N} \sum_{m=1}^{M} L(X_{n,m}, \phi_n, \psi_m),
\]
where $\theta$ is the network parameters.

### 3.3. Inference

**Basic Inference.** In the inference stage, illustrated in the Inference Section of Figure 2, 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,
\[
\hat{\phi}, \hat{\psi} = \arg \min_{\phi, \psi} L(X, \phi, \psi).
\]

However, directly applying Equation 6 is hard, since both the shape and articulation spaces are non-convex. Gradient based optimization can converge to local minima, producing an inaccurate estimation.

To overcome the challenge, we first use Equation 6 to optimize both shape and articulation codes as our initial estimation. In practice, we observe that the articulation code usually converges to a good estimate but the inferred shape codes tend to lead to noisy outputs. 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.** Though fixing the parameters of decoder $f_\theta$ makes sense if the test distribution aligns well with the training distribution, it can be problematic given out-of-distribution observations. To generalize better to out-of-distribution data, the Test-Time Adaptation (TTA) for shape embedding network $f_s$ 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 embedding network $f_s$ using the following objective,
\[
\hat{f}_s = \arg \min_{f_s} L(X, \hat{\phi}, \hat{\psi}),
\]
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 embedding network and articulation network ensures the disentanglement is maintained when the shape embedding network is finetuned. After adaptation, the generation ability (Section 3.4) is still well preserved. Moreover, in practice, one can easily revert to the non-updated model before adapting to individual instance.

To this end, the proposed full Test-Time Adaptation Inference Algorithm is summarized in Algorithm 1. Since back-propagation is very efficient, the iterative optimization only introduces negligible additional computation.

**Algorithm 1 Test-Time Adaptation Inference Algorithm**

**Input:** Target shape $X$.

**Output:** shape code $\hat{\phi}$; articulation code $\hat{\psi}$, updated shape embedding network $\hat{f}_s$.

1: Init $\phi \sim \mathcal{N}(0, \sigma), \psi$
2: $\hat{\phi}, \hat{\psi} = \arg \min_{\phi, \psi} L(X, \phi, \psi)$ \(\triangleright \) Articulation Estimation
3: Re-init $\hat{\phi} \sim \mathcal{N}(0, \sigma)$
4: $\phi = \arg \min_{\phi} L(X, \phi, \hat{\psi})$ \(\triangleright \) Shape Code Estimation
5: $\hat{f}_s = \arg \min_{f_s} L(X, \hat{\phi}, \hat{\psi})$ \(\triangleright \) Test-time Adaptation

### 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. This is shown in Figure 2, Generation section. In this stage, the shape code and finetuned shape embedding network $\hat{f}_s$ obtained in the inference stage is fixed and new shapes are generated by simply inputting new joint angles to the network. If trained with part labels, the model can also output part labels for unseen shapes, as shown in Figure 6.

### 4. Experiment

We experiment with 3D reconstruction and demonstrate that the proposed method can successfully interpolate, extrapolate the articulation space, and synthesize new shapes. We also quantitatively show the proposed method generalize well to partial point clouds obtained from both synthetic and real depth images.

### 4.1. Experiment Setup

For all experiments, the mesh models used are from the Shape2Motion dataset [81]. More details about the network design and training are discussed in the supplementary materials. To evaluate the quality of generated shapes, Chamfer-L1 distance is used as the main evaluation metric, which is defined as the mean of an accuracy and a completeness metric. Following DeepSDF [58], 30,000 points are sampled for each shape for evaluation and Chamfer-L1 distances shown in the paper are multiplied by 1,000. For joint angle estimation, we follow [40] to evaluate the average joint angle error in degrees for each joint.
4.3. Interpolation and Extrapolation

The ability to interpolate and extrapolate between shapes is important for a good 3D articulated object representation. In this section, given two articulated shapes of the same instance, we ask the models to output shapes with articulations in between or beyond.

As shown in Table 2, we quantitatively demonstrate that the proposed model can reliably interpolate shapes. In both cases, the proposed method outperforms DeepSDF by a large margin. As visualized in Figure 4, the baseline generates unrealistic deformed shapes whereas the proposed method produces accurate shape estimations.

In the extrapolation setting on laptops, the proposed method also significantly outperforms DeepSDF. We compute the Chamfer-L1 distance for both methods. The proposed method achieves 0.280 whereas DeepSDF only gets 3.396. As illustrated in Figure 5, the proposed method is capable of extrapolating beyond the range seen during training while DeepSDF fails to produce valid shapes.

The insight is that, even in cases where the two given shapes (the same instance) are with large articulation difference, the inferred shape codes of the proposed method are still close in the high dimensional space. The difference between these shapes is mainly captured by the articulation codes. Therefore, the linear interpolation in the articulation space still produces valid shapes. However, the baseline method learns a non-structured shape space and the interpolation result could be a random point in a high dimensional space.
One main advantage of our learned disentangled representation is its generation ability as described in Section 4.4. One can easily control the articulation input to generate shapes of unseen instances with unseen joint angles.

To provide comparisons, we employ the DeepSDF Interpolation results described in Section 4.3 as baseline. 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 3. We demonstrate that applying Test-Time Adaptation reduces the error further, indicating that Test-Time Adaptation helps with inferring better shape while maintaining a disentangled representation. As visualized in Figure 6, note that one single network is capable of synthesizing fridges with different articulation patterns.

One additional advantage of the proposed method is that joint angles can be estimated simultaneously. We quanti-
Figure 7: Partial point clouds completion one-view setting. For a given depth image visualized as point clouds, we show a comparison of shape completion from our method against DeepSDF and Ground-truth.

Figure 8: 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.

<table>
<thead>
<tr>
<th>Reconstruction</th>
<th>Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepSDF [58]</td>
<td>4.65</td>
</tr>
<tr>
<td>Ours (w/o TTA)</td>
<td>2.53</td>
</tr>
<tr>
<td>Ours</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>3.22</td>
</tr>
</tbody>
</table>

Table 5: 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.

RBO dataset [45] 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 by applying Mask R-CNN [21] on the corresponding rgb images. We generate corresponding point clouds from real depth images following Section 4.5, and then exploit the ground-truth pose to align the point clouds to the canonical space defined by Shape2motion dataset [81].

In Table 5, we show both reconstruction and generation results. Note both models are not trained on real-world depth images. Given a real-world depth image, we obtain its corresponding point clouds, input it to the model trained on synthetic data to reconstruct its 3D shape, and evaluate the reconstruction performance as the one-way Chamfer-L1 distance from ground-truth depth to reconstructed shape. Next, we take the shape code from the previous reconstructed shape and change the articulation code to output shapes at multiple unseen articulation. We take the real depth images at these new articulation and use the generated corresponding point clouds as the ground-truth to evaluate the generation performance. As visualized in Fig 8, the proposed model reliably synthesize shapes at unseen articulation whereas DeepSDF does not have the ability to generate shapes. Table 5 results suggest that applying Test-Time Adapation reduces the error further on both reconstruction and generation.

5. Conclusions

We propose Articulated Signed Distance Functions (A-SDF) to model articulated objects with a structured latent space. A Test-Time Adaptation inference algorithm is introduced to infer shape and articulation simultaneously. We experiment on seven articulated object categories from the shape2motion dataset [81], and demonstrated improved shape reconstruction, interpolation, and extrapolation performance. Moreover, the method allows for controlling the articulation code to generate shapes for unseen instances with unseen joint angles. We also go beyond synthetic data and demonstrate the proposed method can reliably generate 3D shapes from real-world depth images in the rbo dataset [45].

Acknowledgements. This work was supported, in part, by grants from DARPA LwLL, iARPA (DIVA) D17PC00342, NSF IIS-1924937, NSF 1730158 CI-New: Cognitive Hardware and Software Ecosystem Community Infrastructure (CHASE-CI), NSF ACI-1541349 CC*DNI Pacific Research Platform, and gifts from Qualcomm, TuSimple and Picsart.
References


