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

Jiteng Mu[1], Weichao Qiu[2], Adam Kortylewski[2], Alan Yuille[2],
Nuno Vasconcelos[1], Xiaolong Wang[1]
[1]UC San Diego, [2]Johns Hopkins University

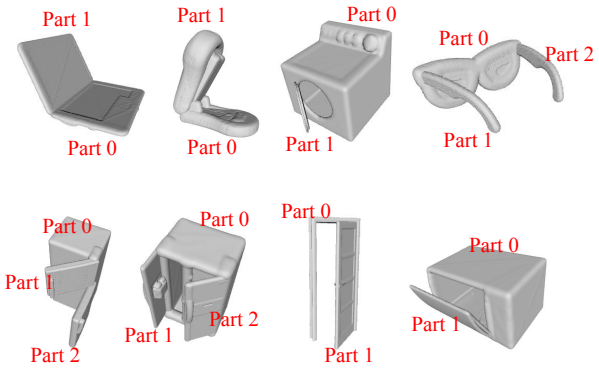

Figure 1: Dataset Part definition.

	Part0	Part1	Part2	Train/Test	Arts
Laptop	Base	Monitor	-	35/11	41
Stapler	Base	Handle	-	23/10	31
Washing Machine	Base	Door	-	24/8	31
Door	Base	Door	-	18/5	31
Oven	Base	Door	-	23/10	31
Eyeglasses	Lens	Left temple	Right temple	33/9	121
Fridge	Lens	Left/Upper door	Right/Lower door	21/7	121

Table 1: Dataset statistics. For each category, we show part definitions, number of training and test instances, and the number of articulations per instance. 'Arts' denotes the number of articulations per instance.

1. Dataset

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. Each instance is articulated to M poses. For each articulated shape, we follow DeepSDF to generate SDF samples. In addition to SDF values, we simultaneously generate part labels. Articulations for training are sampled in a systematic manner as detailed below.

We select seven categories for our experiments, which are laptop, stapler, washing machine, door, oven, eyeglasses, and fridge. The part definitions are illustrated in Figure 1 and Table 1. Besides, in Table 1, we also list the detailed training and test split and number of articulations per instance.

Each instance is articulated to M poses. For laptop, M is set to 41, which corresponds to joint angles $\{-90, -87, -84, \cdots, 24, 27, 30\}$ in degrees. For stapler, washing machine, door, and oven, M is set to 31, which corresponds to joint angles $\{0, 3, 6, \cdots, 84, 87, 90\}$ in degrees. For eyeglasses and fridge, M is set to 121, with each joint angle corresponding to $\{0, 5, 10, \cdots, 40, 45, 50\}$ in degrees.

Training and testing articulation angles are sampled in a systematic manner for different classes. We sample 6 angles for classes of one degree of freedom and 36 angles for classes of two degree of freedom. For laptop, angles used are $\{-72, -54, -36, -18, 0, 18\}$. For stapler, washing machine, door, and oven, angles used are $\{0, 18, 36, 54, 72, 90\}$. For each joint of eyeglasses and fridge, angles used are $\{0, 10, 20, 30, 40, 50\}$. Other articulation angles are used for interpolation, extrapolation, and shape generation evaluation purposes.

To generate SDF values, we first normalize each shape to fit a unit sphere. For all articulated shapes of the same instance, we use one set of normalization parameters to ensure that non-motion parts are aligned. For each shape, we generate $250,000$ SDF samples and its associated part labels. $15,000$ points are uniformly sampled in a unit sphere and $235,000$ points are sampled near to the surface. Half of the surface samples are perturbed with zero-mean Gaussian noise with variance 0.0025 and another half are perturbed with 0.00025.

2. Implementation Details

Network Architecture. The shape embedding network and articulation network architectures used are illustrated in Figure 2 and Figure 3.

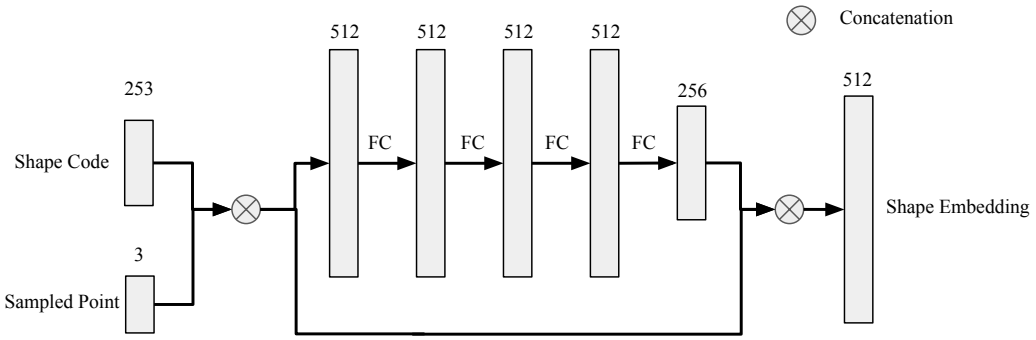

Figure 2: Shape Embedding Network Architecture.

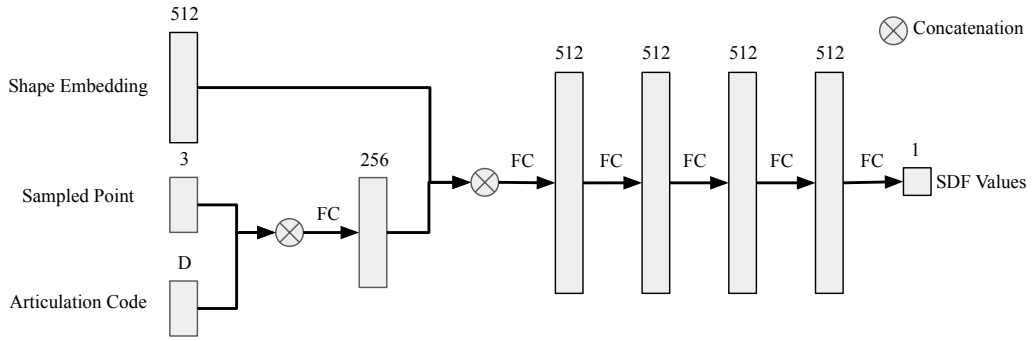

Figure 3: Articulation Network Architecture.

As illustrated in Figure 2, the shape embedding network takes as input the concatenated vector of shape code (with dimension C set to be 253) and a sampled point. The shape encoder is composed of 4 fully connected layers and output a shape embedding with a dimension of 512. $ReLU$ and $Dropout$ are used for all FC layers.

The articulation code (joint angles) with dimension D is first divided by 100 for statistic stability. Then the articulation code (joint angles) is concatenated with a sampled point to form a $D + 3$ dimensional vector. Then the $D + 3$ dimensional vector is forwarded to a FC layer to produce an articulation embedding. Then an MLP, composed of 5 fully connected layers, takes the concatenated shape embedding and articulation embedding to predict the SDF value for the input 3D point, as illustrated in Figure 3. $ReLU$ is used as non-linearities and $Dropout$ is applied after the first four layers of the MLP. A $tanh$ function is used at the end.

Training and Inference. For all experiments, shape codes are randomly initialized with $\mathcal{N}(0, 0.01^2)$. λ_p is set to 0.001(if part labels are used) and λ_ϕ is set to 0.0001. The predicted and ground-truth SDF values are clamped with 0.1 before calculating losses. The Marching Cubes algorithm is applied to extract an approximate iso-surface given the predicted SDF values.

During training, we randomly sample 8,000 positive points (outside the surface) and 8,000 negative points (inside the surface) for each shape in a batch. Network f_θ and shape codes are trained for 1,000 epochs. The learning rate for model parameters is set to be 0.0005 and for shape codes is set to be 0.001. Both learning rates reduce to half after every 250 epochs.

During inference, we randomly sample 4,000 positive points (outside the surface) and 4,000 negative points (inside the surface) for each shape. Articulation codes are initialized to be the middle joint angle of the observed training range. For the proposed Test-Time Adaptation optimization procedure, in the first step, the articulation code and shape code is optimized for 800 iterations. The learning rate for the articulation code is set to be 5 and decays to 0.5 after 400 iterations. The learning rate for the shape code is set to be 0.005 and decays to 0.0005 after 400 iterations. In the second step, the shape code is re-initialized and the articulation code is fixed. We optimize the shape code for 800 iterations. The learning rate for the shape code is set to be 0.005 and decays to 0.0005 after 400 iterations. In the third step, the inferred shape code, articulation code, and articulation network are fixed, we optimize the shape embedding network for 800 epochs. The learning rate for the shape embedding

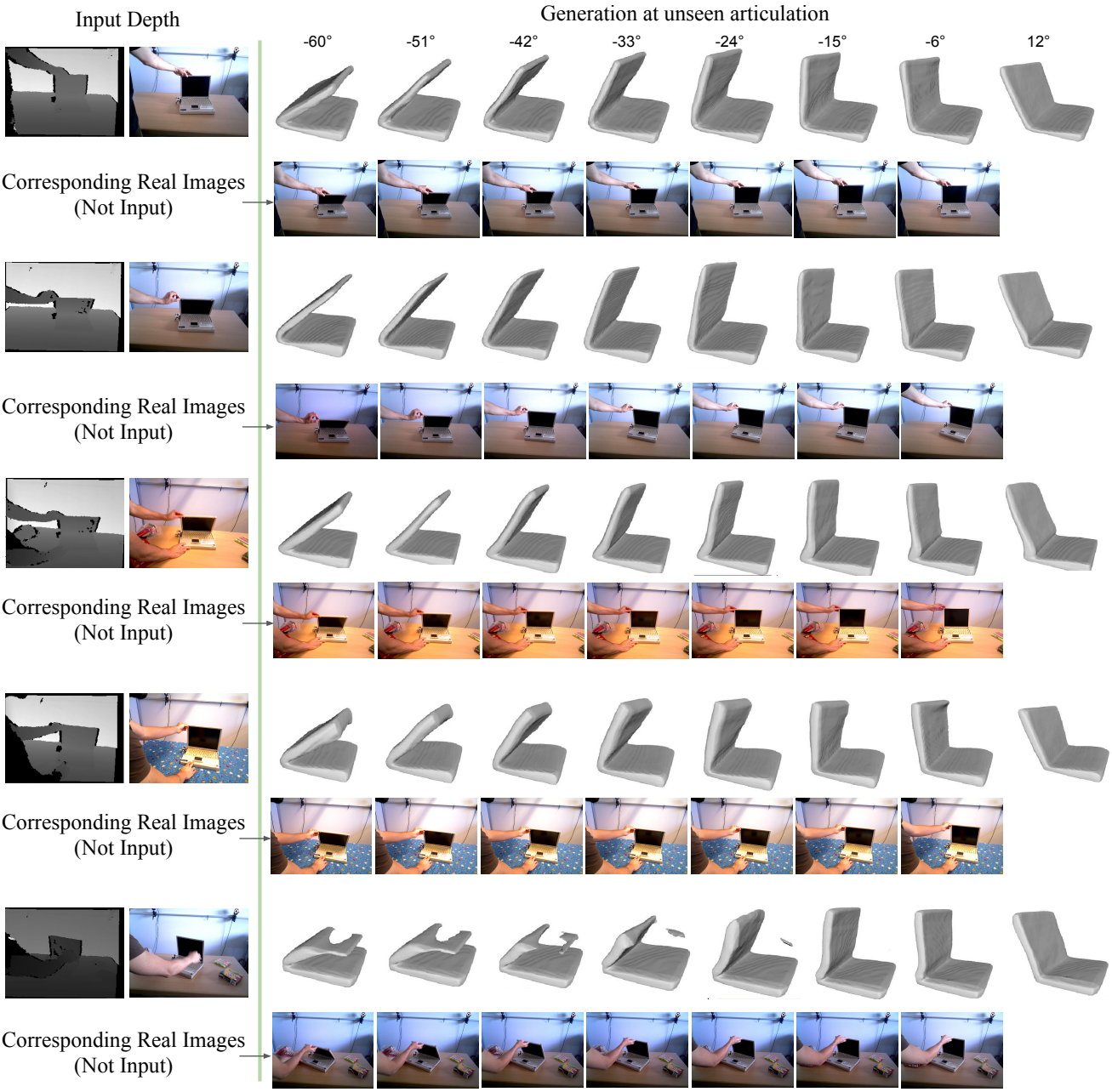

Figure 4: Test on real-world depth images. Given a single real-world depth image, our method generates articulated objects at unseen articulation. Corresponding RGB images are *not* used in the algorithm.

network is set to be 0.00005 and decays to 0.000005 after 400 iterations.

3. Test on Real-world Depth Images

The RBO dataset is composed of 358 RGB-D video sequences of articulated objects under varying conditions (light, perspective, background, interaction). All sequences are annotated with ground truth of the poses of the rigid parts and the joint angles of the articulated object obtained with a motion capture system. We take laptop as an example to test our algorithm. Specifically, we take depth images from sequences in the rbo dataset and crop laptops from depth images by applying Mask R-CNN on the corresponding rgb images. We select 10 frames for each sequence cov-

	Laptop	Stapler	Wash	Door	Oven	Eyeglasses	Fridge
DeepSDF	0.35	3.73	4.29	0.61	5.33	1.63	0.80
DeepSDF+art	0.21	3.94	2.97	0.45	4.97	1.46	0.75
Ours	0.13	2.55	2.13	0.17	1.83	1.16	0.69

Table 2: Reconstruction comparison under the Exp 4.2.

	Surface Sampling			Surface + Free space		
	Laptop	Door	Washing	Laptop	Door	Washing
DeepSDF	3.31	5.76	14.08	3.93	5.28	10.57
Ours	2.25	0.86	6.29	2.17	0.73	7.27

Table 3: Partial point cloud completion under the Exp 4.5.

ering a large angle range. We generate corresponding point clouds from the cropped real depth images. We exploit the ground-truth object pose and camera transformations in the rbo dataset to transform the depth into the object coordinate space. We then align the point cloud in the object coordinate space to the canonical space defined by Shape2motion dataset.

We test the model trained on synthetic laptops and directly apply the trained model to the real-world depth images. Visualizations are shown in Figure 4. Note that in the last row, even the laptop is heavily occluded, our method can still output reasonable shapes, which indicates that the trained model learns a good shape prior.

4. Ablation on Articulation Code

In this section, we explore how the articulation code influences the reconstruction results. Specifically, we implement the variant of DeepSDF using the same architecture but taking both shape code and articulation code as inputs (Table 2, *DeepSDF + art*). In addition, different from ours, shape code sharing is not implemented for this setting. As shown in Table 2, this improves DeepSDF but is still worse than our method, which indicates the effective design of embedding articulation code into the model is non-trivial.

5. Ablation on Partial Point Clouds Sampling

In this section, we explore how different sampling strategies influences reconstruction results from partial point clouds. As introduced in Exp 4.5., we approximate the SDF values to be η and $-\eta$ by perturbing each of them η distance away from the observed depth point along the computed surface normal direction. η is set to be 0.025 in our experiments. We refer to this sampling strategy as Surface Sampling as shown in Table 3. As introduced in DeepSDF, we also experiment with the setting where free-space sampling is incorporated. Free-space refers to the empty space between the object surface and camera. We sample points in free-space and enforce larger-than-zero constraints similar to DeepSDF.

We perform free-space sampling (right side) and show the reconstruction error on the right side of Table 3. We find the results are similar as when we only sample near surface (left side).