CaDeX: Learning *Ca*nonical *De*formation *C*oordinate *S*pace for Dynamic Surface Representation via Neural Homeomorphism Supplementary Document

Jiahui Lei
University of Pennsylvania
leijh@seas.upenn.edu

Kostas Daniilidis
University of Pennsylvania
kostas@cis.upenn.edu

In this supplementary document, we first provide additional proof for the volume conservation property in Sec. S.1. Then, we provide more details of our architecture and implementation in Sec. S.2, and more details of the experiments and quantitative results in Sec. S.3, as well as more qualitative results and comparisons in Sec. S.5. Additional discussions and results are in Sec. S.4.

S.1. Volume Conservation

As discussed in Sec.3.3, if the homeomorphism is implemented by NICE [3], using an example split pattern of $[x, y]; [z]$, each coupling block is:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z + t(x, y|c) \end{bmatrix}. \tag{1}$$

The determinant of the Jacobian of this mapping is:

$$|J| = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{\partial t}{\partial x} & \frac{\partial t}{\partial y} & 1 \end{vmatrix} = 1, \tag{2}$$

which indicates that the mapping preserves the spatial volume.

S.2. Implementation Details

S.2.1. Deformation Encoder

As introduced in Sec.3.2, the deformation embedding c_i for each frame is output by the deformation encoder and we demonstrate three encoder choices for two types of inputs as illustrated in Fig. S1.

PF-encoder The most straightforward approach is to directly use a standard PointNet [9] to process each input frame separately. Fig S1-Top shows the diagram of this per frame (*PF*) encoder. As discussed in Sec.4.1, when predicting the deformation embedding for each frame, no information from other frames can be considered. Therefore, we use this type of encoder only when the observation of each frame is relatively complete, as is the case in the sparse point cloud setup. But, surprisingly, as shown in Tab.2, the *PF* encoder achieves the highest performance since it would potentially result in a higher canonicalization level and avoid overfitting.

ST-encoder We utilize the ST-PointNet from LPDC [10] to process the sequence input as shown in Fig. S1-Middle. First, the 4D coordinates (x, y, z, t) of the input point clouds are processed by the Temporal-PointNet (marked red in Fig. S1) with both spatial and temporal pooling in each residual block and generate temporal features for each input frame. Second, the 3D coordinates (x, y, z) are processed by the Spatial PointNet (marked yellow in Fig. S1) just the same as the PF-encoder and produce the geometry features for each frame. Finally, the spatial and temporal features for each frame are fused by Fusion MLPs (marked orange in Fig. S1). We enhance the original Fusion MLPs from LPDC [10] by applying pooling across geometry features from different frames and fusing with an additional global geometry feature. As mentioned in Sec.5, the continuity across time of the c_i output by the ST-PointNet is not guaranteed. Therefore, we apply a Gaussian filter to the deformation embeddings output by the ST-PointNet on the time dimension when taking the depth observations as input. The *ST* encoder is the default configuration of our architecture since it can simultaneously fuse spatial and temporal information, so that it can handle more general inputs like the partial observations from the depth video.

SET-Encoder When the input is a set of deformed surfaces without explicit order, we develop a 2-Phase Encoder with a code query network to produce the deformation embeddings for different deformation states, as illustrated in Fig. S1-Bottom. In our particular application, the input set is the set of deformed surfaces of one articulated object at different articulation angles. First, we apply a standard PointNet [9] (PointNet-1, marked yellow in Fig. S1) to each input state separately and summarize each state as an embedding z_i. Then we regard these embeddings as a set of points in a higher dimensional space and apply another PointNet (PointNet-2, marked red in Fig. S1) to get a global

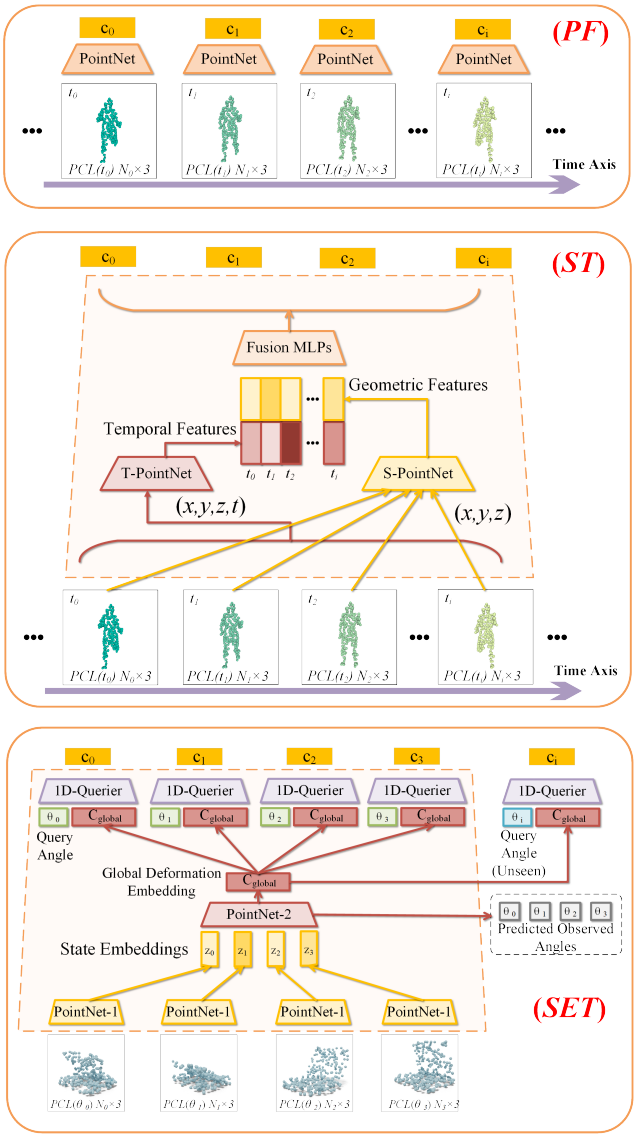

Figure S1. **Deformation Encoders**: Top: per-frame PointNet Encoder (*PF*). Middle: ST-PointNet Encoder [10] (*ST*). Bottom: The 2-Phase PointNet and code query network for set inputs (*SET*).

deformation embedding for this instance. To finally produce the deformation embedding for each desired articulation angle θ, we utilize an MLP as the code query network. It gets the query angle and global deformation embedding as input, and outputs the deformation code corresponding to the input query angle. Note that the angle θ can be the angle of the input deformation state (θ_0 - θ_3 in the figure) or an unseen angle (θ_i in the figure) for interpolation or generation. To provide a complete model, we also utilize a small prediction head on top of the last feature before global pooling in PointNet-2 to predict the deformation angle for the input observed states.

S.2.2. Homeomorphism Architecture

The basic idea of the invertible neural networks we are using [3, 4] is described in Sec.3.2. We provide more details of our implementation that helps to increase the expressivity in this section (Fig. S2). Each coupling block in our implementation has two subblocks, such that the second subblock has the complementary input split with the first subblock. For example, the first subblock (Coupling Block 0-A in Fig. S2) changes $[z]$ based on $[x, y]$ conditioning on the embedding, then, the second subblock (Coupling Block 0-B in Fig. S2) will immediately change $[x, y]$ based on the $[z]$ output from the first subblock conditioning on the embedding. Additionally, we apply a code projector (Code MLP BLK0 in Fig. S2) for each block to project the input deformation embedding to a block-specific condition. Note that the code projector is not shared across blocks. We also provide an optional global explicit affine and scaling transformation as in [8] at the beginning of the network to exclude the modelling of the trivial motion. The rotation, translation, and scale factors are predicted directly from the input deformation embedding. To provide a more stable canonical space, the coordinates output by the neural homeomorphism are optionally compressed via a sigmoid function to a bounded cube before sending to the geometry encoder and decoder.

S.3. Experiment Details

S.3.1. Training and Testing Details

During training, our model is optimized by Adam optimizer with a starting learning rate of 0.0001. The learning rate has a step decay with a decay rate of 0.3; the epochs in which the learning rate decay will occur are adjusted to each dataset depending on the size of the dataset. The model is trained with the gradient clip. We periodically compute the validation IoU metric and select the best model based on the validation performance. During testing, the marching cubes algorithm is conducted based on the library from O-Flow [7] and O-Net [5]. We refer the reader to [7] for details of the evaluation metrics. Note that the chamfer distance and the correspondence error reported in our paper, as well as [7, 10], are multiplied by 10.

S.3.2. Modeling Human Bodies

We refer the reader to [7, 10] for details of the experiment settings and data generation. Our model is configured with 6 coupling blocks (Sec.S.2.2) in the neural homeomorphism. During training, the sequence has 17 frames evenly distributed in t dimension, and we randomly supervise the occupancy prediction of 8 frames. Each frame has 512 randomly uniformly sampled query positions. If the correspondence loss is enabled, we predict the corresponding positions in every other frame of the 100 randomly sam-

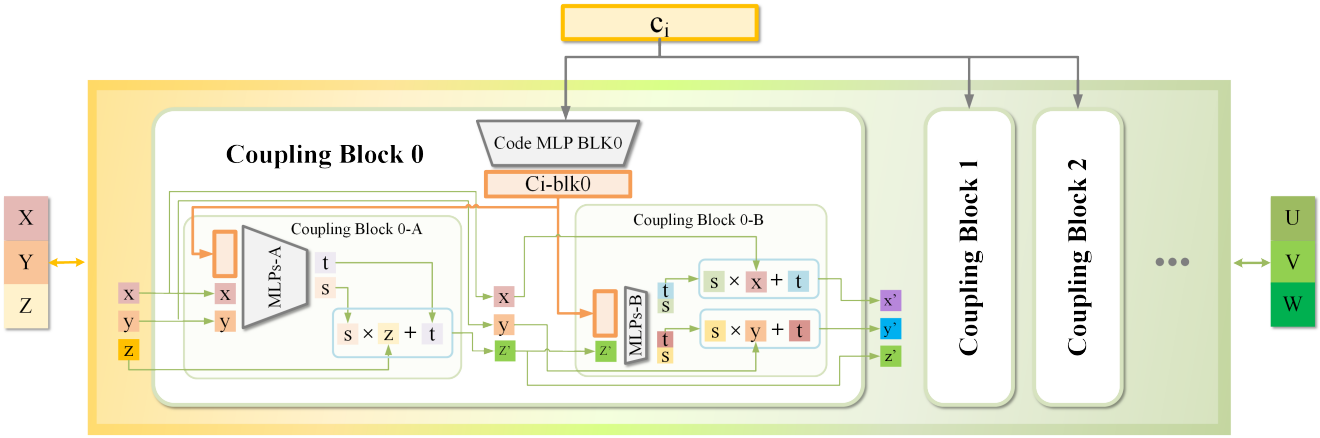

Figure S2. Detailed architecture of the invertible neural homeomorphism

pled surface points from the first frame and supervise with Eq.12 and error order $l = 1$. The model is trained with loss mixing weights $w_R = 1.0$ and $w_C = 1.0$ (if use $\mathcal{L}_C$).

S.3.3. Modeling Animals

We generate the dataset for Sec.4.2 based on the DeformingThings4D-Animals [12] (DT4D-A) dataset, which contains 1227 synthetic animations spreading 38 raw animal categories ("raw" means that the category is counted based on the name id prefix in the latest released data of [12]). Due to the uneven distribution of categories, we reserve 21 minor categories and generate data based on 17 raw animal categories named in DT4D-A as: bear, moose, fox, deer, puma, rabbit, elk, grizz, dragon, tiger, procy, doggie, huskydog, raccoon, bunny, bucks and canie. The dataset is split into training (75%), validation (7.5%), unseen motion testing (9.4%) and unseen individual testing (9.4%) split. Since the raw meshes in DT4D-A sometimes have ill-behaved areas (e.g., self-intersected or overlapped), we preprocess the meshes to resolve these issues and use the processed meshes to generate the occupancy samples, as well as to filter the surface correspondence point samples. The data format is the same as Sec.4.1 (S.3.2). Since the shapes in DT4D-A [12] are more complex, we generate the occupancy field supervision with 50% near-surface samples and 50% uniform samples. We also generate depth videos. For each animation, we render the depth video of two randomly posed (poses selected from the semi-sphere) static cameras and back project the depths to get the single-view point clouds.

We sample 512 points with noise std 0.001 from the point cloud trajectories (PCL) or depth back-projected point clouds (Dep) as input. Similar to Sec. S.3.2, the input sequence has a length of 17 and we supervise the reconstruction of 8 randomly sampled frames. The occupancy supervision has 256 near-surface samples and 256 uniform samples per frame. The correspondence is also supervised as in Sec. S.3.2. Our model is configured with ST Encoder and 6 NVP coupling blocks (Sec.S.2.2) in the neural homeomorphism. We train with loss mixing weights $w_R = 1.0$ and $w_C = 8.0$. Since animals have some motion, for example, turning around, which can be decomposed into trivial transformation components, we enable the global explicit rigid transformation (Sec. S.2.2) at the beginning of the neural homeomorphism. Additionally, as described in Sec. S.2.1, we apply a Gaussian filter on the temporal dimension of the output deformation embeddings from the ST-encoder with kernel size 5 when taking depth observations as input. The quantitative performance difference of the Gaussian filter under the depth input is shown in Tab. S1.

Mehtod	Seen individual			Unseen individual		
	IOU	CD	Corr	IOU	CD	Corr
un-filtered	70.8%	0.095	0.190	53.9%	0.183	0.323
filtered	**71.1%**	**0.094**	**0.186**	**55.7%**	**0.175**	**0.301**

Table S1. **Extension of Tab.4**: We report the results of the filtered version in the main paper for the depth inputs. We also provide the results of the un-filtered version here.

The baseline methods [7, 10] are fully trained under the same settings and we select their best models based on the validation metric. We found that the original LPDC [10] implementation will converge to a degenerated reconstruction (a thin animal shape-like plate) on the animal dataset. The potential reason is that the MLP which models the cross-frame deformation outputs the absolute deformed position, which leads to bad initialization when trained with complex shape and motion. We make LPDC [10] work on the animal dataset by changing the MLP's output to the relative deformation vectors instead of the absolute deformed positions.

Cate	Model	IoU↑	CD↓	Corr↓	t(s)	$\theta(deg)$
Lapt	A-SDF	65.0%	0.076	-	3.15	3.68
	LPDC	69.0%	0.077	0.101	0.47	2.99
	Ours	**76.2%**	**0.075**	**0.086**	0.83	2.65
Stap	A-SDF	59.6%	0.153	-	2.94	2.97
	LPDC	49.8%	0.201	0.313	0.45	2.70
	Ours	**61.7%**	**0.128**	**0.207**	0.82	2.95
Wash	A-SDF	42.3%	0.191	-	3.62	2.53
	LPDC	46.1%	0.201	0.242	0.62	3.48
	Ours	**48.3%**	**0.147**	**0.180**	1.33	2.67
Door	A-SDF	**45.0%**	0.069	-	3.41	1.49
	LPDC	21.5%	0.155	0.241	0.44	1.86
	Ours	42.4%	**0.064**	**0.093**	1.30	1.95
Oven	A-SDF	49.5%	0.144	-	4.05	3.46
	LPDC	51.1%	0.172	0.229	0.68	2.35
	Ours	**59.2%**	**0.123**	**0.164**	1.53	2.02
Glas	A-SDF	**55.5%**	**0.138**	-	3.16	3.12
	LPDC	43.9%	0.233	0.275	0.41	3.25
	Ours	55.3%	0.172	**0.256**	0.81	3.39
Frid	A-SDF	69.1%	0.122	-	3.74	6.42
	LPDC	63.4%	0.161	0.210	0.63	4.35
	Ours	**69.4%**	**0.118**	**0.135**	1.23	3.64
AVE	A-SDF	55.2%	0.127	-	3.44	3.38
	LPDC	49.2%	0.171	0.230	0.53	3.00
	Ours	**58.9%**	**0.118**	**0.160**	1.12	2.75

Table S2. Articulated objects per category performance with sparse point cloud input

Cate	Model	IoU↑	CD↓	Corr↓	t(s)	$\theta(deg)$
Lapt	A-SDF	64.3%	0.077	-	3.56	3.47
	LPDC	63.9%	0.089	0.112	0.44	3.13
	Ours	**71.0%**	**0.066**	**0.112**	1.01	3.40
Stap	A-SDF	**56.7%**	0.155	-	3.36	2.86
	LPDC	55.0%	0.140	**0.217**	0.40	3.29
	Ours	56.2%	**0.139**	0.228	0.94	2.98
Wash	A-SDF	45.6%	0.154	-	3.92	9.77
	LPDC	46.1%	0.181	0.181	0.77	8.49
	Ours	**49.2%**	**0.144**	**0.170**	1.54	8.24
Door	A-SDF	**41.4%**	0.078	-	3.46	2.10
	LPDC	18.1%	0.182	0.253	0.51	2.37
	Ours	37.2%	**0.076**	**0.116**	1.04	2.01
Oven	A-SDF	50.6%	0.147	-	4.33	7.03
	LPDC	47.8%	0.296	0.450	0.73	7.03
	Ours	**55.6%**	**0.129**	**0.168**	1.75	6.31
Glas	A-SDF	49.8%	**0.157**	-	3.05	4.44
	LPDC	34.7%	0.297	0.417	0.42	3.66
	Ours	**50.3%**	0.162	**0.233**	0.91	2.98
Frid	A-SDF	68.8%	0.124	-	3.87	5.77
	LPDC	59.4%	0.185	0.249	0.49	5.95
	Ours	**74.9%**	**0.093**	**0.100**	1.59	4.46
AVE	A-SDF	53.9%	0.127	-	3.65	5.06
	LPDC	46.4%	0.195	0.269	0.54	4.85
	Ours	**56.4%**	**0.116**	**0.161**	1.26	4.34

Table S3. Articulated objects per category performance with depth inputs

S.3.4. Modeling Articulated Objects

We generate the dataset of articulated objects based on the meshes provided by A-SDF [6] based on Shape2Motion [11] with 7 distinct articulated categories: laptop, stapler, door, washing machine, oven, eyeglasses, and refrigerator. Eyeglasses and refrigerators have two deformable angles and the others have one. We utilize the same training-testing split as [6], but we also split a small validation set out of the training set for model selection. The data format is the same as Sec. S.3.3.

Our model is configured with 3 NVP coupling blocks and uses the SET encoder (Sec. S.2.1). During training, 300 points are randomly sampled with noise std 0.003 from the sparse point cloud or depths as input. The input has 4 frames at randomly sampled articulation angles. The SET-encoder takes 4 input observations and produces a global deformation code, as well as 4 predicted articulation angles for the input frame (as illustrated in Fig. S1 and Sec. S.2.1). For completeness, the predicted angles are supervised by an additional MSE regression loss. Then 8 ground truth query articulation angles (4 for input frames, 4 for unseen angles) query the global deformation code through the code query network and produce 8 deformation embeddings. We supervise all 8 frame reconstructions, and each frame has 1024 near-surface samples and 1024 uniform samples. We also supervise the correspondence by predicting the corresponding positions of 256 randomly sampled surface points from one input frame to all 7 other frames with the loss error order $l = 2$. The losses are mixed as: $\mathcal{L} = w_R\mathcal{L}_R + w_C\mathcal{L}_C + w_\theta\mathcal{L}_\theta$, where L_θ is the angle regression loss, $w_c = w_\theta = 1.0$ and $w_C = 8.0$ for laptop, stapler and door categories, and $w_c = 32.0$ for the washing machine, oven, refrigerator and eyeglasses categories. The baseline methods [6, 10] are fully trained and we select their best models based on validation metrics.

During inference, the inputs are observations of 4 randomly sampled articulation angles, and the outputs are mesh reconstructions with the correspondence of 8 frames (4 for input, 4 for angle generation). The model weights are trained for each category separately. As mentioned in Tab.4 in the main paper, due to the 8-page limit, we report the per-category performance here in Tab. S2 for point cloud inputs and in Tab. S3 for depth inputs. The IoU, chamfer distance, and correspondence error are reported as averaged across seen and unseen angles, since we do not observe a

significant difference between them. To exclude the effect of the articulation angle prediction error, we report the metrics for the results based on the ground truth query angle and separately report the angle prediction error.

Our method guarantees to preserve the topology, but as mentioned in Sec.5 in the main paper, our method can not handle the topology changes (Fig. S3) in the current dataset we are using. We leave for future work to explore how to selectively preserve or alter the topology.

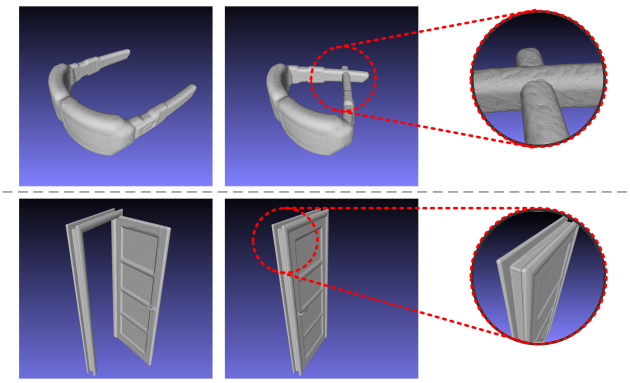

Figure S3. Topology changes in [6, 11] dataset. Top: unrealistic topology changes. Bottom: realistic local topology changes.

S.3.5. Ablation Study

In Sec.4.4, we conduct the ablation studies by replacing the 2-way bijection or removing the canonical geometry encoder. We provide more details and analysis here.

Bijection: In addition to the description in Sec.4.4, there is a remeshing in the ablative model after the per-frame marching cubes mesh extraction. The goal of this remeshing is to produce a vertex corresponding mesh sequence (with shared connectivity) since when the canonical map is not invertible, the deformation/correspondence function in Eq.3 and Eq.7 can not be established. The remeshing is achieved by warping the mesh of the first frame to every future frame. We project the vertices of the extracted canonical mesh of the first frame in CaDeX to every extracted canonical mesh of the future frame and find the barycentric weight. We then use the barycentric weight and the vertices from the extracted meshes in the future frames to produce the warped position. Therefore, the significant time consuming of this ablative model reported in Tab.5 can be attributed to the per-frame marching cubes and the remeshing, which are all resulted from the 1-way mapping.

Canonical Geometry Encoder: In this ablative model, the global shape embedding is directly obtained from the ST-Encoder's Fusion MLP (Sec. S.2.1). We observe a decrease in the reconstruction performance in Tab.5, but a slight improvement in the accuracy of the correspondence. One potential reason is that we are optimizing the fixed-capacity canonical map with respect to multiple gradients from different sources. In our full model, the canonical map is optimized to form a canonical shape that is easier for the canonical geometry encoder to summarize and better for the reconstruction accuracy and correspondence prediction. When we remove the canonical geometry encoder, more capacity of the canonical map can be allocated to optimize the correspondence prediction. However, the canonical geometry encoder is still worth doing because it brings a larger reconstruction improvement and is more helpful when dealing with partial observations.

S.4. Additional Discussion and Results

S.4.1. Interpreting Canonical Shapes

As shown in the qualitative results, the learned canonical shapes look different from the ones in the input space (the final reconstructions). For example, the canonical shape of the human body in Fig.4(left) amplifies the tissue fluctuations and has more stable poses across different pose sequences, helping the shape encoder-decoder to better express the local details. Such behavior is caused by our end-to-end framework and training supervision. The canonical shape and canonical maps can be jointly optimized during training so that the capacity of both the deformation module and the shape module is balanced.

On the other hand, the learned canonical shape is not oversimplified (e.g, a sphere), which might be a practical problem for learning the canonical template shape [13] from shape collections [2]. This might be caused by several facts: Our method is to model temporal sequences, where frames have closer correlation to each other than the discrete shape instances in the shape collection [2]. Additionally, our neural homeomorphism is initialized as a near-identity mapping, and hence the learning of the canonical shape starts from the mean shape. Finally, the end-to-end framework is unlikely to converge to the situation that the geometry module models oversimplified canonical shapes, but leaves most shape components expressed by the deformation module.

S.4.2. Efficiency of the NVP and NICE

	Training time (ms) per recon-frame			Testing Mesh Extracing Time (s)		
	Total	Forward	Backward&Optim	Total	First Frame MC	Rest Frames
O-Flow [7]	74.4	5.02	69.4	0.680	0.322	0.357
CaDeX	5.46	1.11	4.35	1.125	0.439	0.686

Table S4. Efficiency comparisons, all experiments are on the same gpu device and the gpu utility is near 100%.

We provide an additional discussion regarding the efficiency of the Neural ODE [7] deformation and our NVP/NICE deformation in Tab. S4. The advantage of our method over ODE is in the training phase. The forward and backward passes of our method are as simple as a standard MLP architecture, but the ODEs need to densely query

the whole trajectory to compute the gradient as well as to forward the integration. However, since O-Flow has a simpler velocity MLP and can do sequential prediction for every time step via one forward of the ODE during testing, it achieves slightly faster testing speed.

S.4.3. Other types of deformation

Figure S4. Other types of deformation: the left is the visualization of the learned canonical shape and the right are the input sparse point cloud and the output reconstruction sequence.

Beyond the deformation of human bodies, animals, and articulated objects, whose motion might be dominated by the global articulation, we also apply our method to fully non-rigid motions. We build a synthetic dataset from the sofa category of ShapeNet [2] similar to the warping cars dataset used in Tab.2 [7] in O-Flow paper (O-Flow's is not public yet). The sofas are randomly warped through a random space deformation field. Fig S4 shows the qualitative results and the evaluation metrics are as follows: IoU 73.2%, CD 0.077, Corr 0.099.

S.5. More Qualitative Results

Human Bodies Fig. S5 shows the qualitative comparison on D-FAUST [1] human bodies.

Animals Fig. S6 shows the qualitative comparison on DeformingThings4D-Animals [12] dataset with sparse point cloud inputs, and Fig. S7 shows the depth inputs.

Articulated Objects Fig. S8 shows the comparison on Shape2Motion [6, 11] articulated objects with sparse point cloud inputs, and Fig. S9 is for the depth inputs.

References

[1] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 6

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5, 6

[3] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 1, 2

[4] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 2

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

[6] Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. *arXiv preprint arXiv:2104.07645*, 2021. 4, 5, 6, 10

[7] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5379–5389, 2019. 2, 3, 5, 6

[8] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3204–3215, 2021. 2

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1

[10] Jiapeng Tang, Dan Xu, Kui Jia, and Lei Zhang. Learning parallel dense correspondence from spatio-temporal descriptors for efficient and robust 4d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6022–6031, 2021. 1, 2, 3, 4

[11] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8876–8884, 2019. 4, 5, 6

[12] Takafumi Taketomi Yang Li, Hikari Takehara, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. *arXiv preprint arXiv:2105.01905*, 2021. 3, 6

[13] Zerong Zheng, Tao Yu, Qionghai Dai, and Yebin Liu. Deep implicit templates for 3d shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1429–1439, 2021. 5

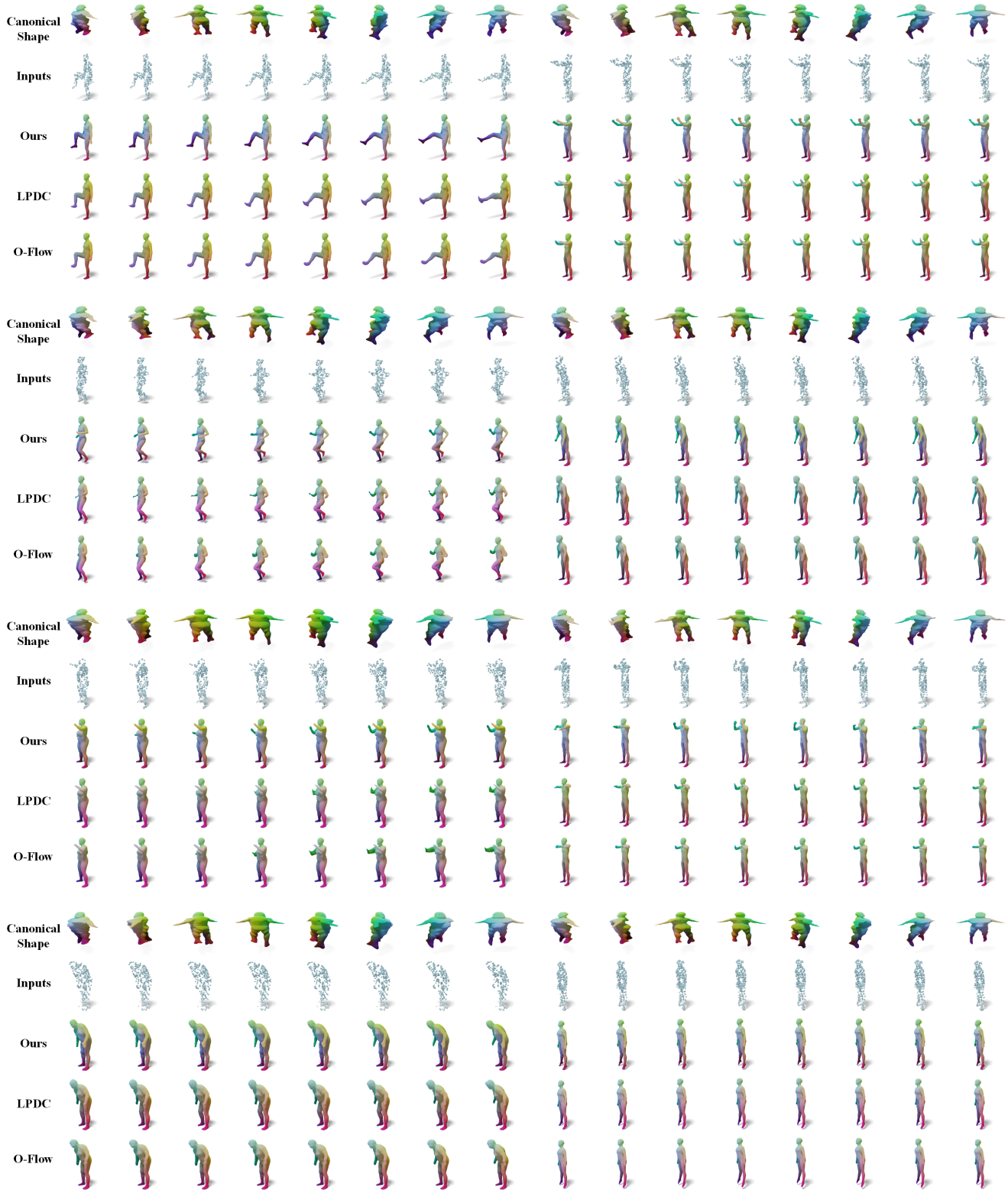

Figure S5. Modelling human bodies: each column corresponds to a time frame of the reconstruction or a visualization viewpoint of the learned canonical shape.

Figure S6. Modelling animals with sparse point cloud inputs: each column corresponds to a time frame of the reconstruction or a visualization viewpoint of the learned canonical shape.

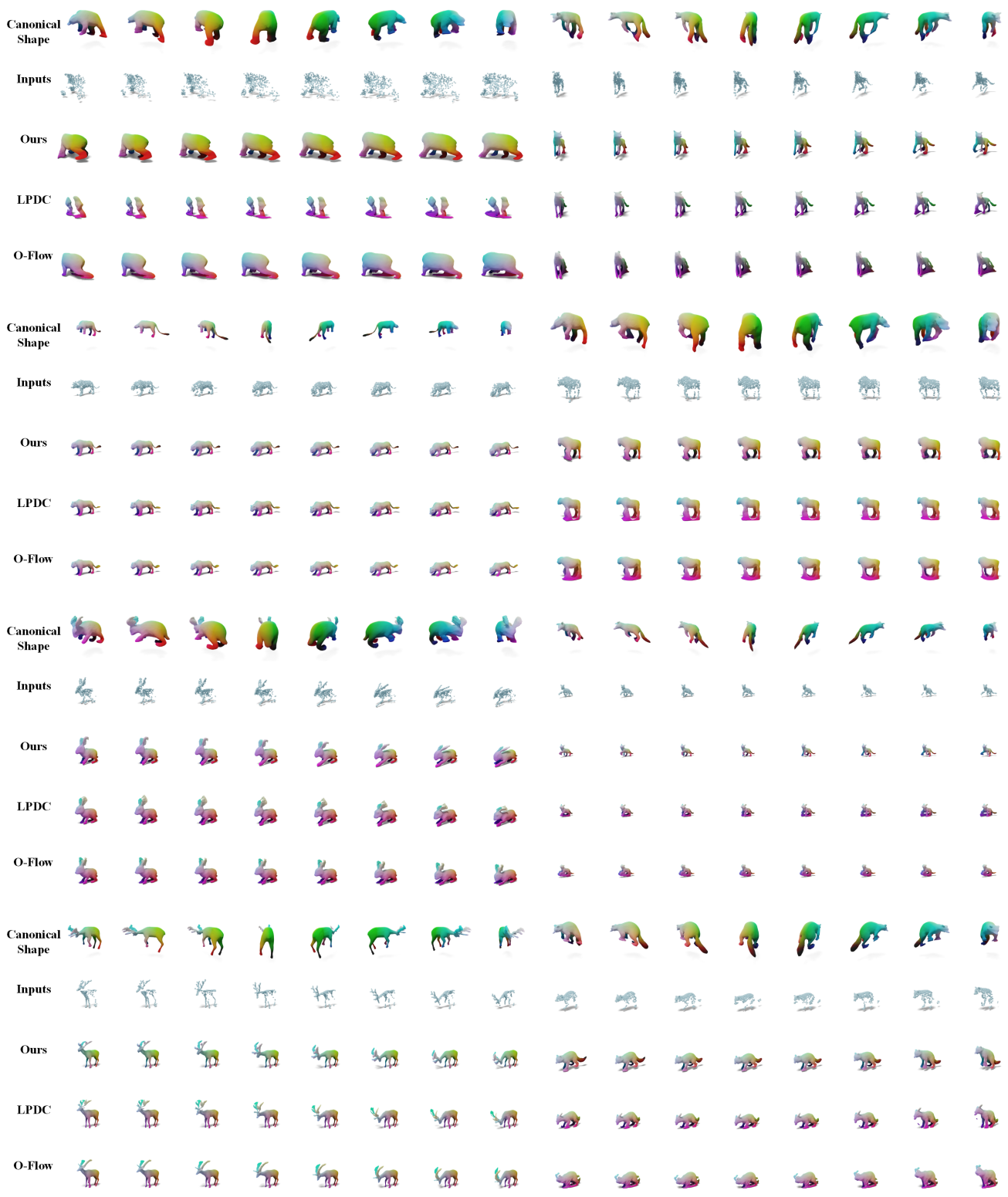

Figure S7. Modelling animals with depth inputs: each column corresponds to a time frame of the reconstruction or a visualization viewpoint of the learned canonical shape.

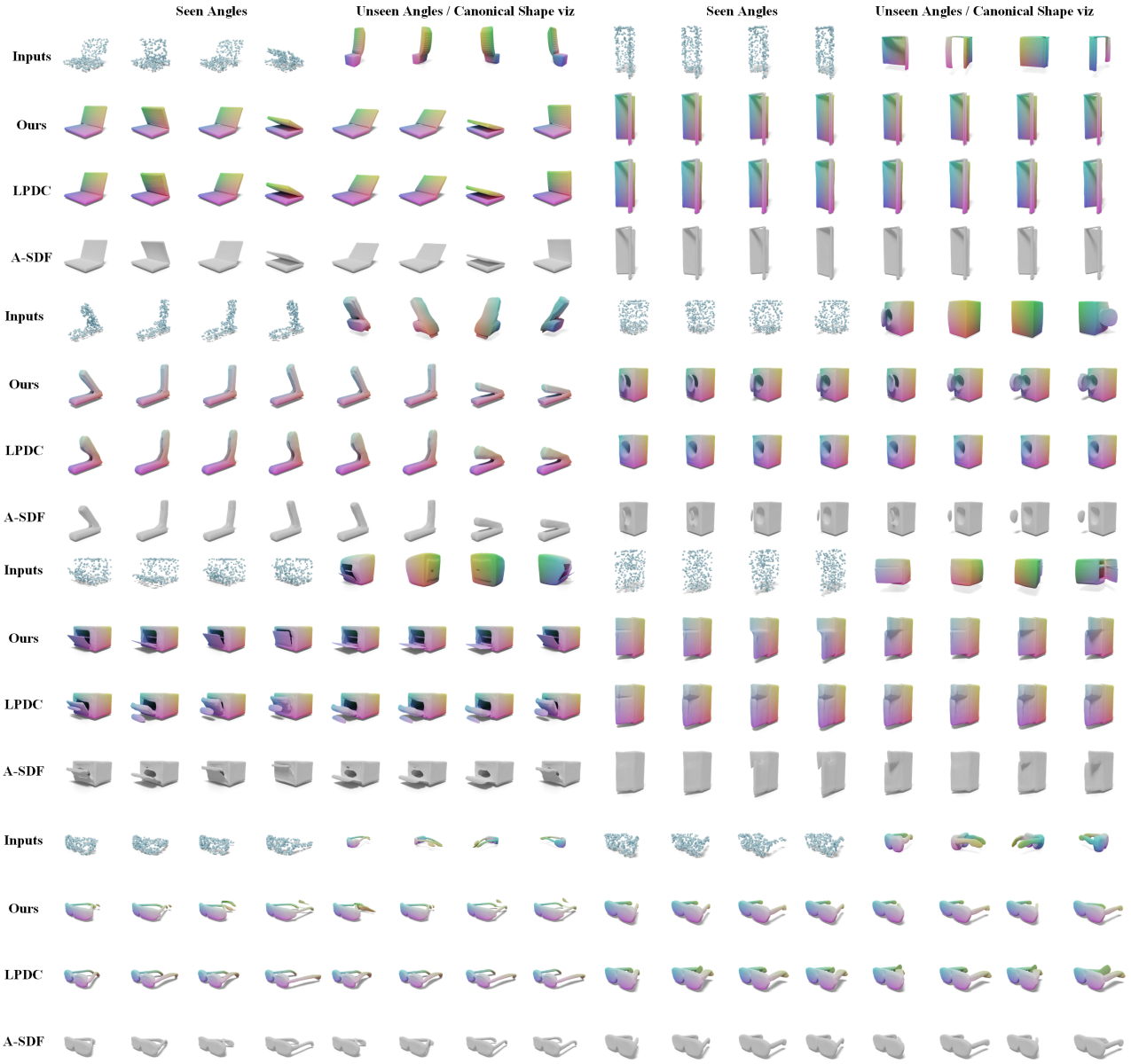

Figure S8. Modelling articulated objects with sparse point cloud inputs: the left-top four point clouds are the inputs and the right-top are four visualization viewpoints of the learned canonical shape. The left four columns correspond to the reconstruction of the seen articulation angles and the right four correspond to the unseen angles. A-SDF [6] is not colored since it can not produce the correspondence.

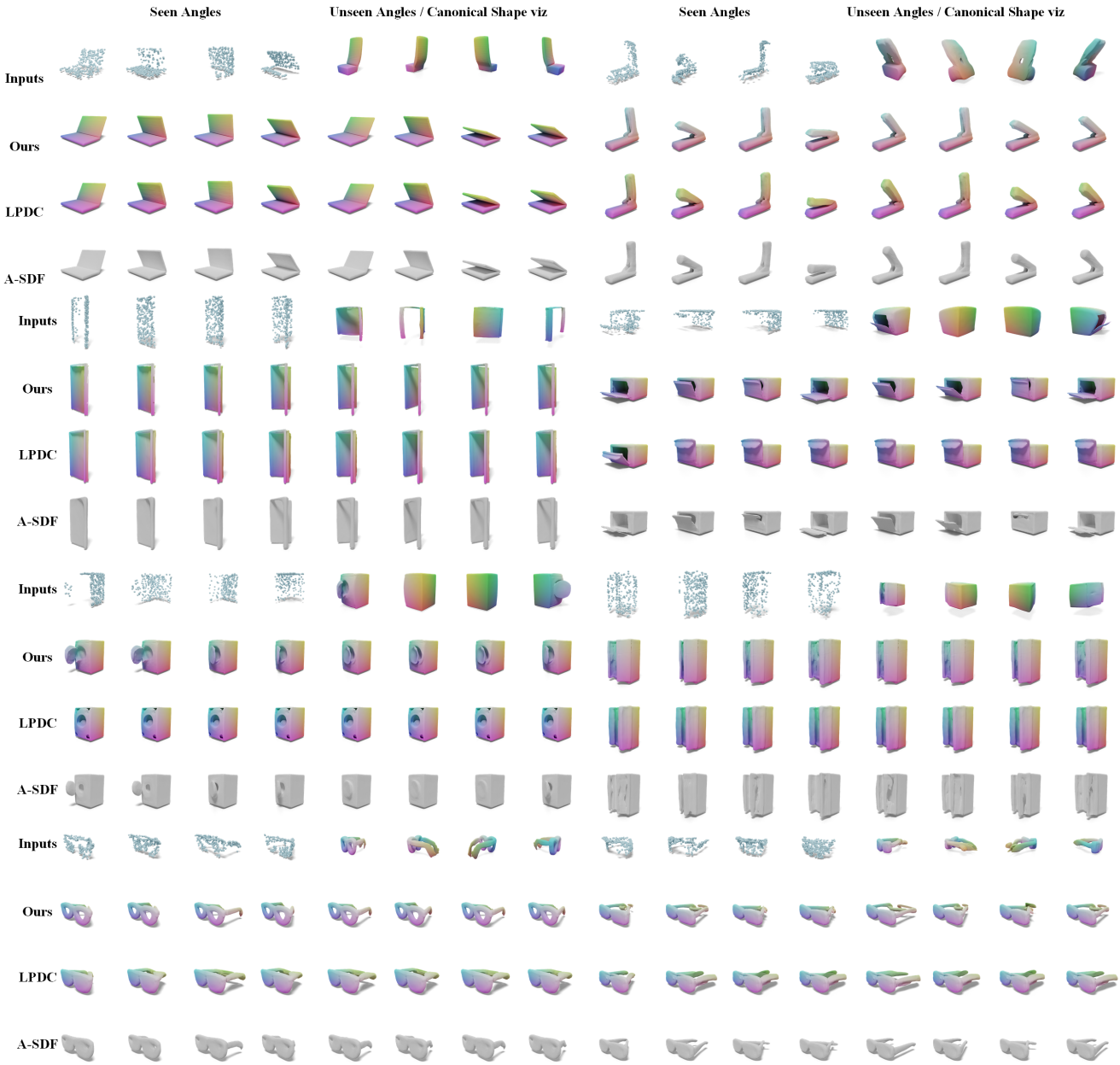

Figure S9. Modelling articulated objects with depth inputs: the left-top four partial point clouds are the inputs and the right-top are four visualization viewpoints of the learned canonical shape. The left four columns correspond to the reconstruction of the seen articulation angles and the right four correspond to the unseen angles.