Supplementary Document for Deep Implicit Templates for 3D Shape Representation

A. Overview

This supplementary document provides more experimental details. In Sec.B, we present the details about how we implement our network, how we conduct the experiments and the complexity of our method. In Sec.C we provide additional qualitative results on chairs, further showing the representation power of our method in the cases of extremely challenging structural variations. Additional experiments are also presented in Sec.C. More details about the extensions (Sec.6 in the main paper) are provided in Sec.D. Finally, we discuss the limitations and potential future work in Sec.E. Please refer to the supplementary video for more results and visualizations.

B. Experimental Details

B.1. Datasets and Preprocessing

Our experiments are conducted using the ShapeNet Core V2 dataset[2]. For simplicity, we mainly focus on five classes of shapes, i.e., sofas, cars, airplanes, chairs and tables. We use the training and testing splits from DeepSDF[12]. We also follow the practice of DeepSDF to preprocess the shapes, split training/testing data, and extract SDF samples for network training.

For the experiment on keypoint detection, we utilize keypoint annotations from KeypointNet[14]. We collect the annotations for the airplanes and cars that are also in our training/testing sets.

B.2. Architecture Details

Warping Function $\mathcal{W}$. In all experiments, $\mathcal{W}$ is implemented as a vanilla LSTM network. Specifically, $\mathcal{W}$ is composed of an LSTM cell and a linear layer. The LSTM cell has a hidden state size of 512, and the linear layer is used to convert the 512-dimensional output state to 6-dimensional transformation parameters.

Implicit Template $\mathcal{T}$. Similar to DeepSDF, the implicit template $\mathcal{T}$ is parameterized as a multi-layer perceptron (MLP). The numbers of its neurons are $(3, 256, 256, 256, 256, 256, 1)$. We use weight normalization, a dropout probability of 0.05 and ReLU activation in our network except the last layer that uses a hyperbolic tangent activation.

Parameter Name	Value
S (LSTM steps, Eqn.5)	8
δ_h (Eqn.9)	0.25
ϵ (Eqn.10)	0.5
λ_{pw} (Eqn.11)	0.005
λ_{pp} (Eqn.11)	1×10^{-4}
$1/\sigma^2$ (Eqn.11)	1×10^{-4}
N (Number of Samples per Shape, Eqn.7)	5000
SDF Truncation Band Width	0.1
Batch Size	24
Number of Epochs	2000
Learning Rate for Networks	5×10^{-4}
Learning Rate for Latent Codes	1×10^{-3}
Adam β_1	0.9
Adam β_2	0.999
Surface Isolevel	0.
Marching Cube Resolution	256

Table A: Hyperparameters for network training and evaluation.

B.3. Training Details

We use PyTorch to implement our networks. The hyperparameters needed for network implementation and training are reported in Tab.A. Note that during network training, the learning rate of Adam is decayed by a factor of 0.5 every 500 epochs. For all baseline methods in comparison, we use their open-source code but replace their training/testing splits with ours for fair comparison.

To clarify, in Table.2 (main paper) we train one single network for each object category (airplanes, sofas, cars, chairs) to calculate the results of our methods, while in Figure.4 (main paper) we train different networks for different subclasses in order to better demonstrate the capability of our method in terms of handling topological changes and shape variations.

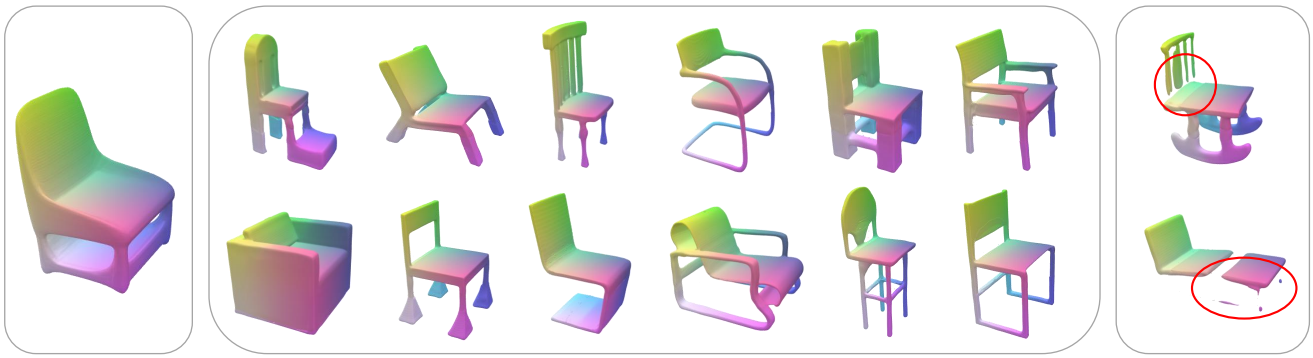

Figure A: Our results for the full chair category. When training the networks on the full shape repository of chairs, our method can still learn a plausible implicit template and represent various shape structures accurately.

B.4. Testing Details

Reconstruction. To test the reconstruction accuracy of our method and other DeepSDF variants (e.g., DeepSDF[12], C-DeepSDF[3], DualSDF[7]), we extract SDF samples for the shapes in test set and estimate the latent code to fit the shapes while keeping the network parameters fixed. The latent code is obtained through:

$$\hat{\boldsymbol{c}} = \arg\min_{\boldsymbol{c}} \mathcal{L}_{data} + \lambda \mathcal{L}_{code},$$

where $\lambda = 1 \times 10^{-4}$, $\mathcal{L}_{data}$ is an ℓ_1 loss measuring the reconstruction error, and $\lambda\mathcal{L}_{code}$ shares the same definition as the main paper. The optimization is performed using Adam optimizer with a learning rate of 5×10^{-4} and 800 iterations.

To evaluate the reconstruction performance of other methods that use encoder-decoder architectures, we directly feed the point samples on the test shapes into the network to obtain the reconstruction results.

Correspondences. We use keypoint detection results as a surrogate to evaluate the accuracy of correspondences for the baselines and our method. Specifically, after training the network, we first estimate the latent code for each test shape using the aforementioned approach, then search its closest training shape according to the L2 similarity of latent codes, and finally use the dense correspondences to transfer the keypoint annotations from the training shape to the test shape. We use the same training/testing splits as the reconstruction evaluation and the whole training set is used to push forward keypoints. Note that the correspondences in our method are established using the canonical positions of points. For SIF, the correspondences are obtained using the matching method described in [5]. For PointFlow, we regard the positions of the points at the distribution prior (i.e., the 3D Gaussian) as their "canonical positions". For AtlasNet, we project surface points onto the their corresponding square patches and use their coordinates on the atlas to establish correspondences. When calculating the PCK scores in Tab.3, we simply ignore those keypoints that only appear in either the detection results or the ground-truth annotations.

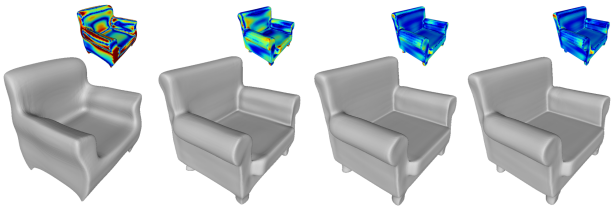

Figure B: Results and reconstruction error at different LSTM steps.

B.5. Complexity

The parameter number of our network is 1.852M, only slightly greater than that of DeepSDF (1.843M). This suggests that our representation is as compact as DeepSDF. Given a latent code, the inference time of our method is 14.770 seconds when using the hierarchical extraction method proposed in [10] and running on a single GPU of NVIDIA TITAN X (Pascal). Compared to DeepSDF which takes about 7 seconds per model, our network doubles the inference time of DeepSDF due to the introduction of multi-step transformations.

B.6. Visualization of Transformation Steps

In order to help readers better understand how the warping function works, we visualize one example of transformations outputted by each step of the warping LSTM, as presented in Fig. B.

Method	CD Mean of Sofas (10^{-3})
Ours using 4 small MLPs	0.157
Ours w/o scaling output	0.096
Ours (full)	0.093

Table B: Reconstruction accuracy of the baseline method and our full network.

C. More Experiments

C.1. More Results

In Fig.A we demonstrate our results on chairs. Note that unlike Fig.4 (main paper), we train the deep implicit templates for the full chair category. The results further proves that our method can still generalize on the categories that exhibit high structural variations, although it may suffer from some reconstruction artifacts for extremely challenging cases.

Please refer to the supplementary video for more results and visualizations.

C.2. More Experiments

Ablation on Network Design. In Tab. B, we conduct an additional ablation study to evaluate our design of the warping function.

To evaluate our choice of an MLP-based implementation, we conduct an experiment where we construct the warping network using 4 two-layer MLPs and enable curriculum supervision. Emperically, we find that although this implementation is faster to train, its reconstruction results are not as accurate as the results of our LSTM-based implementation; see the second and the last row of Tab. B. Therefore, this MLP-based implementation can be an alternative if fast training is desired over accuracy.

We also evaluate the impact of the scaling output in Eqn. (4), and find that the model using scaling achieves slightly better accuracy than the model without scaling (the last two rows of Tab. B). Note that the scaling output is inspired by [5, 8] and is introduced to boost the expression power. It adds almost no complexity to the model and can be removed if necessary.

Experiments on the Number of Meshes Used to Push Forward Keypoints. We also evaluate the correspondence accuracy when using different numbers of training meshes to push forward keypoint labels. The numerical results for airplanes are plotted in Fig.C. We find that our method is able to achieve comparable correspondence accuracy even only 20% of training meshes are used to push forward keypoints; we think this is because of the shape redundancy in ShapeNet dataset.

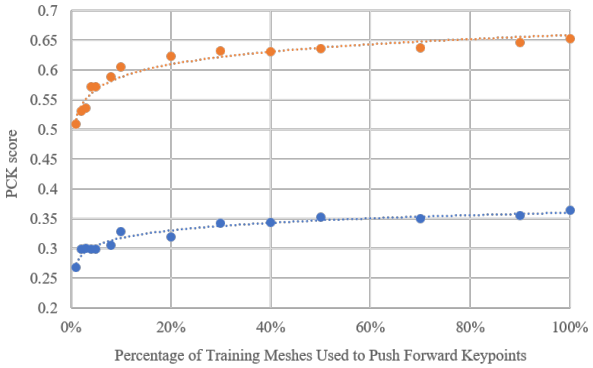

Figure C: Correspondence accuracy using different numbers of training meshes. We plot the PCK scores with threshold of 0.01 (blue) and 0.02 (orange) for airplanes.

D. Extension Details

User-defined templates. To show that our method supports manually defined templates, we collect 1600 clothed human models from Deep Fashion 3D dataset[15] and train a network to model humans in different clothes. Note that all human models are all in rest pose and wear different types of garments, and we use the SMPL model[9] in its mean shape (i.e., $\boldsymbol{\beta} = \mathbf{0}$) as the template. After 1500 epochs of training, the network is able to deform the "implicit" SMPL model to represent various clothed humans, as shown in Fig.9 (main paper).

Correspondence annotations. To extend our method to the scenarios where correspondence annotations are available, we collect three mesh sequences from D-FAUSE[1] dataset, which records different human motions and provides ground-truth registration. We preprocess the sequences following the practice of O-Flow[11]. Then we split each sequence into two subsequences with equal lengths, and use the first half for network training while the second half for testing. Note that unlike O-Flow, we train one individual network for each sequence. We use the correspondence error metric proposed in [11] to calculate the numeric results and evaluate the effect of the additional correspondence loss in Fig.11 (main paper).

E. Limitations & Future Work

Limitation. Our method has several limitations. Firstly, unlike original SIF[5], DSIF[4] or AtlasNet[6], our representation is class-specific in order to establish accurate dense correspondences across a class of shapes. Therefore, our method cannot generalize well to unseen classes. In Fig.D, we demonstrate a failure case where the network for sofas is forced to reconstruct a airplane. Secondly, our method is based on the assumption that all shapes in the same category share a common structure. Consequently,

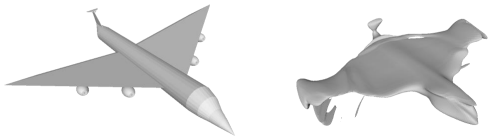

Figure D: A failure case: The model trained on sofas fails to reconstruct a airplane.

our method is more suitable for object classes that are amenable to being modeled as the deformation of a template (e.g., cars, airplanes), but degenerates slightly for object classes that exhibit too many structural variations, as shown in the failure cases in Fig.A. Thirdly, initial experiments show that our method does not work well on large articulated movements such as human body motions.

Future work. To overcome the aforementioned limitations, we can learn a set of fine-grained implicit templates for object parts, and then combine them adaptively to describe different structures. Furthermore, our method currently uses the auto-decoder architecture proposed in DeepSDF[12]; we can introduce pixel-aligned feature as in PIFu[13] to recover more geometric details from an input image, which we leave for future research.

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.

[2] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015.

[3] Yueqi Duan, Haidong Zhu, He Wang, Li Yi, Ram Nevatia, and Leonidas J. Guibas. Curriculum deepsdf, 2020.

[4] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas A. Funkhouser. Deep structured implicit functions. *CoRR*, abs/1912.06126, 2019.

[5] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas A. Funkhouser. Learning shape templates with structured implicit functions. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV*, pages 7153–7163. IEEE, 2019.

[6] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. *CoRR*, abs/1802.05384, 2018.

[7] Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge Belongie. Dualsdf: Semantic shape manipulation using a two-level representation. *arXiv*, 2020.

[8] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas A. Funkhouser. Learning part-based templates from large collections of 3d shapes. *ACM Trans. Graph.*, 32(4):70:1–70:12, 2013.

[9] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: a skinned multi-person linear model. *ACM Trans. Graph.*, 34(6):248:1–248:16, 2015.

[10] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] Michael Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV*, pages 5378–5388. IEEE, 2019.

[12] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[13] Shunsuke Saito, , Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[14] Yang You, Yujing Lou, Chengkun Li, Zhoujun Cheng, Liangwei Li, Lizhuang Ma, Cewu Lu, and Weiming Wang. Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations. *CoRR*, abs/2002.12687, 2020.

[15] Heming Zhu, Yu Cao, Hang Jin, Weikai Chen, Dong Du, Zhangye Wang, Shuguang Cui, and Xiaoguang Han. Deep fashion3d: A dataset and benchmark for 3d garment reconstruction from single images, 2020.