Supplementary for "Learning Deep Implicit Functions for 3D Shapes with Dynamic Code Clouds"

Tianyang Li[1], Xin Wen[1,2], Yu-Shen Liu[1]*, Hua Su[3], Zhizhong Han[4]
[1] School of Software, BNRist, Tsinghua University, Beijing, China
[2] JD Logistics, Beijing, China
[3] Kuaishou Technology, Beijing, China
[4] Department of Computer Science, Wayne State University, Detroit, USA
lity20@mails.tsinghua.edu.cn wenxin16@jd.com liuyushen@tsinghua.edu.cn
shlw@kuaishou.com h312h@wayne.edu

1. Implementation Details

1.1. Network Architecture

For fair comparisons, we leverage the same decoder as NGLOD [5] in our network in the single shape fitting experiment. In the 3D dataset reconstruction experiment, we leverage a decoder with the same structure as IM-Net [2], which is also the same practice as MDIF [3]. Note that we only use one decoder while MDIF [3] uses a decoder for each level, thus we have fewer parameters in decoder than MDIF [3].

1.2. Hyperparameters

We implement our method using PyTorch and train the network by Adam optimizer. And we apply different hyperparameter settings in different experiments.

In the single shape fitting experiment, we set the initial value of learning rate to be 0.001 and decay the learning rate by 0.5 every 35 epochs. We train the network for 150 epochs with the batch size equal to 512. A set of 500K points is resampled at every epoch as training data, in which 100K points are sampled uniformly in the space, 200K points from the object surface and the others are sampled near the surfaces.

In the 3D dataset reconstruction experiments, we set the initial value of learning rate to be 0.001 and 0.003 for parameters in decoder and latent codes, respectively. And we decay the learning rates by 0.5 every 200 steps. We train the network for 500 epochs. Each batch of data contains 16 shapes and 4096 query points for each shape. During inference on test split of the dataset, we fix decoder parameters and optimize latent codes for 500 steps. We use an initial learning rate of 0.01 during inference and also decay the learning rate by 0.5 every 200 steps. In this experiment, we sample a set of 200K points for each shape, and use the same point set for all epochs. The point set contains 100K points sampled uniformly in the space and the other points sampled near the object surface.

*The corresponding author is Yu-Shen Liu. This work was supported by National Key R&D Program of China (2018YFB0505400, 2020YFF0304100), the National Natural Science Foundation of China (62072268), and in part by Tsinghua-Kuaishou Institute of Future Media Data.

2. Computational Cost

Our method has comparable computational cost with existing work such as NGLOD [5]. In our implementation, the batch size B (i.e. the number of query point) is set to 512 for Thingi32 [6] and 4096 for ShapeNet [1], as given in Sec. 1.2. Operations in our method (like interpolation and CP loss) do not involve complex calculations and very large tensors. And our method converges fast, as code locations are simple and efficient to be optimized.

In the single shape fitting experiment, taking NGLOD [5] as the comparison target, both NGLOD [5] and our method need $\sim$ 100 epochs to converge when optimizing the latent codes (and locations). We use 150 epochs for both NGLOD [5] and our method in this experiment to ensure full convergence. On a single GPU (GeForce RTX 2080 Ti), NGLOD [5] takes $\sim$ 20 seconds to run one epoch, while our method takes only $\sim$ 16 seconds. Previous methods like NGLOD [5] also need to compute Z explicitly (but automatically done by API like from PyTorch), it seems our interpolation runs faster than trilinear interpolation applied in NGLOD [5] under the same B. On the test split of ShapeNet [1], our method optimizes codes and locations for 500 iterations and takes $\sim$ 6 seconds for each shape during inference. But we find that $1 \sim 2$ seconds and ~ 150 iterations are enough to get fine results.

3. Additional Results and Discussion

3.1. More Visualization of Optimization Process

We show more cases of optimization process on Thingi32 [6] in Fig. 2 and Fig. 3.

3.2. Results on Large-Scale Scenes

We randomly select one room from SceneNet [1], and take the entire room as input to briefly show the ability of our method to scale to scenes. We compare our method with LIG [4]. And we achieve $CD = 0.96$, outperforming $CD = 1.90$ by LIG [4]. As shown in Fig. 1, our method has a better reconstruction quality. We will further explore our method for large-scale scenes and conduct more experiments in our future work.

3.3. Results for Simple Shapes in ShapeNet

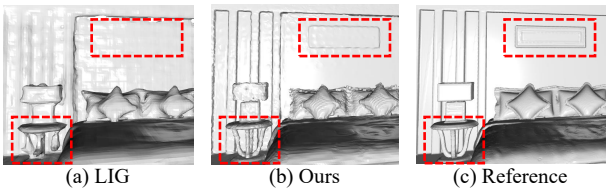
As shown in Table 2 of our main paper, MDIF [3] has better results than our proposed DCC-DIF in categories with simple shapes, such as cabinet and car. MDIF [3] has advantages in reconstructing simple shapes, which benefits a lot from its hierarchical architecture (especially the global latent code in top level). In contrast, our moving codes and CP loss demonstrate great advantages for fitting complex shapes. But our method reconstructs simple shapes not as well as MDIF, which is mainly due to lack of global perception from global code. It is hopeful to improve this disadvantage by developing hierarchical DCC-DIF.

3.4. Understanding CP Loss from Another View

As $\mathcal{A}$ is a 'similarity' whereas $\mathcal{D}$ is a 'distance', multiplying $\mathcal{A}$ and $\mathcal{D}$ in Eq 7 of our main paper may seem confusing. It might be helpful to understand why we multiply $\mathcal{D}$ for Code Position (CP) loss in the back-propagation process.

As mentioned in our paper, we cut off gradient back propagation to $\mathcal{A}$ in Eq 7. Minimizing the CP loss leads to reduction of distances at different degrees. As the distances are computed from (x, y, z) coordinates of query points and latent codes, the reduction of distances further leads to updating of (x, y, z) coordinates of latent codes. The attraction $\mathcal{A}$ determines which distance is more important to reduce. The distances corresponding to higher attraction will reduce more. Intuitively, the (x, y, z) coordinates of latent codes will get closer to query points with higher errors/attractions, where complex geometry details probably exist. Large attractions lead to reduction of corresponding distances and maybe increase of distances which correspond to small attractions, but it does not matter.

[1] https://robotvault.bitbucket.io



(a) LIG (b) Ours (c) Reference

Figure 1. Visualization on SceneNet. We show results of LIG and our method in (a) and (b), respectively. And (c) is the reference.

3.5. About Faraway Points

In Fig 1(c) of our main paper, we can find several points are placed far away from the object surface. As we compute the interpolated feature with weights based on distances, the faraway points almost do not contribute to the reconstruction of the boundary. We used to remove these points, but observe no significant changes in results. Since our operations are implemented by matrix calculation, these faraway points have little effect on computational efficiency. Thus we still keep these faraway points to facilitate matrix calculation.

References

[1] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, L. Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *ArXiv*, abs/1512.03012, 2015. 1

[2] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5932–5941, 2019. 1

[3] Zhang Chen, Yinda Zhang, Kyle Genova, Sean Fanello, Sofien Bouaziz, Christian Häne, Ruofei Du, Cem Keskin, Thomas Funkhouser, and Danhang Tang. Multiresolution Deep Implicit Functions for 3D Shape Representation. In *IEEE/CVF International Conference on Computer Vision*, pages 13087–13096, October 2021. 1, 2

[4] Chiyu Max Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas A. Funkhouser. Local Implicit Grid Representations for 3D Scenes. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6000–6009, 2020. 2

[5] Towaki Takikawa, Joey Litalien, K. Yin, Karsten Kreis, Charles T. Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 1

[6] Qingnan Zhou and Alec Jacobson. Thingi10K: A Dataset of 10, 000 3D-Printing Models. *ArXiv*, abs/1605.04797, 2016. 1, 2

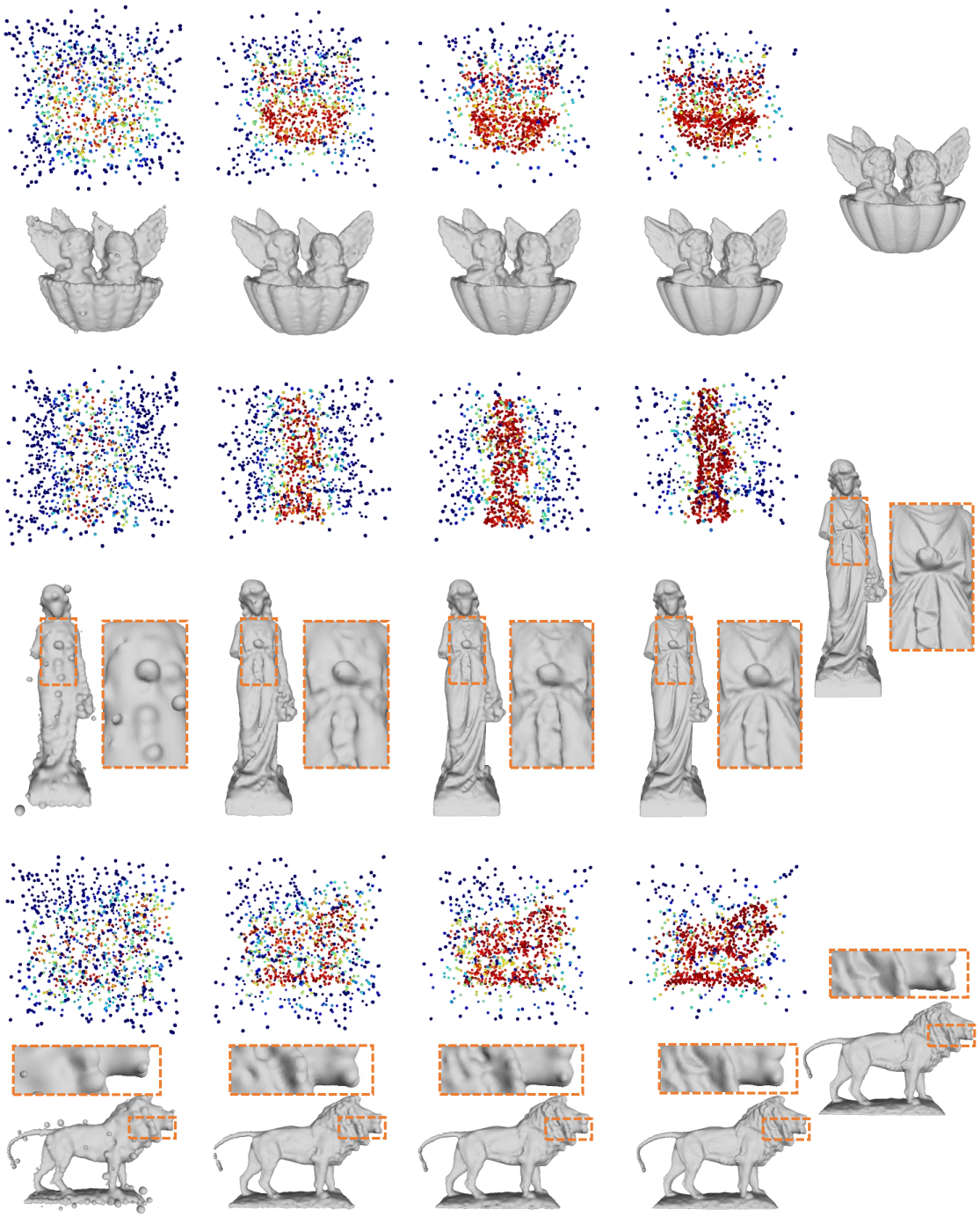


Figure 2. Illustration of moving code positions during optimization. We typically show four states during optimization. Initial and final states are shown in (a) and (d), respectively, while (b) and (c) show two intermediate states. The (e) shows the references. For each 3D shape, we show our reconstruction results (below) and code positions (above), where the warmer color indicates the local codes are closer to the surface.

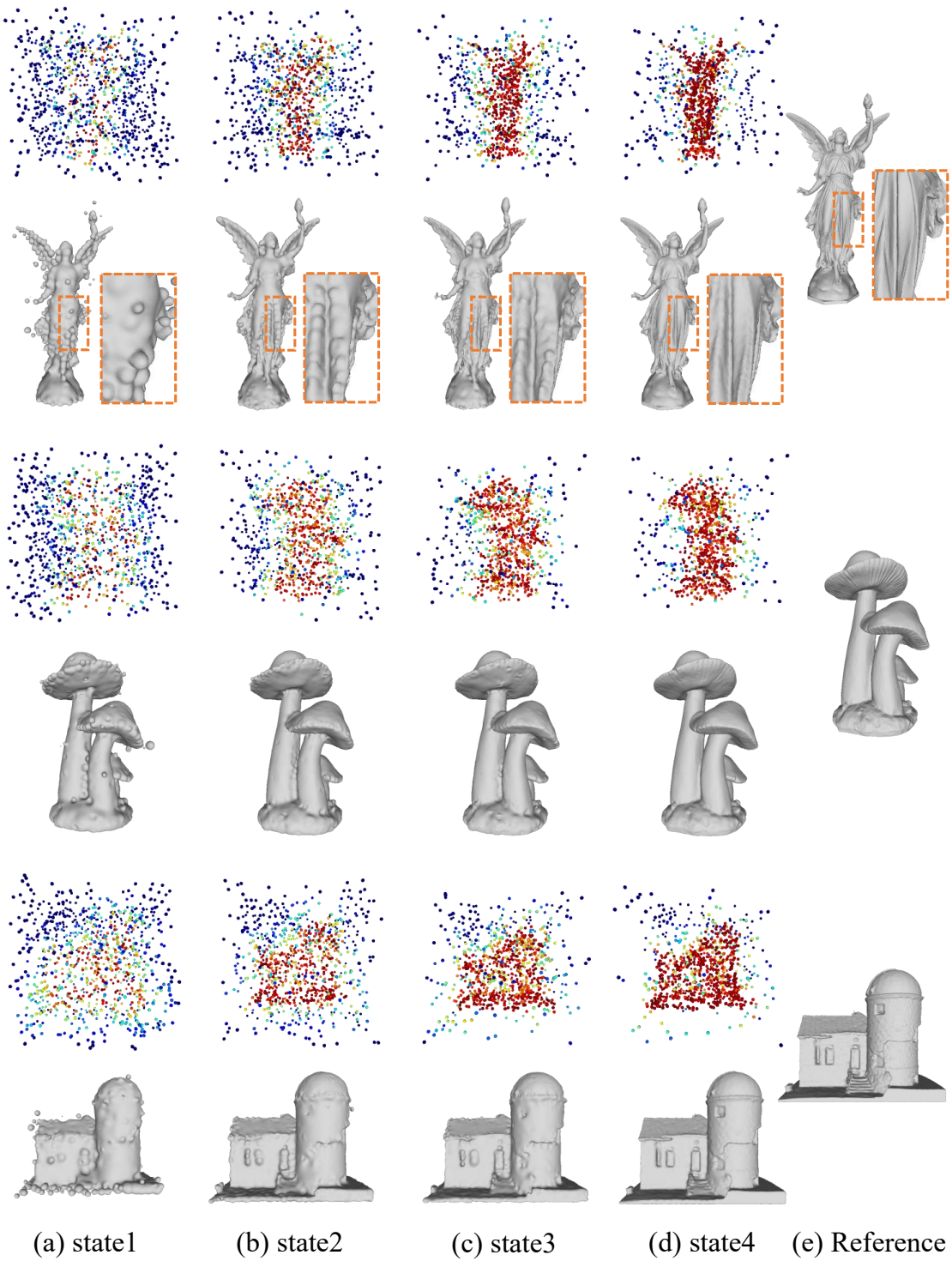


Figure 3. Illustration of moving code positions during optimization. We typically show four states during optimization. Initial and final states are shown in (a) and (d), respectively, while (b) and (c) show two intermediate states. The (e) shows the references. For each 3D shape, we show our reconstruction results (below) and code positions (above), where the warmer color indicates the local codes are closer to the surface.