DeepMetaHandles: Learning Deformation Meta-Handles of 3D Meshes with Biharmonic Coordinates — Supplementary Material

Minghua Liu[1] Minhyuk Sung[2] Radomir Mech[3] Hao Su[1]
[1]University of California San Diego [2]KAIST [3]Adobe Research

Please check out our webpage[1] for the animations of the learned meta-handles. In this supplementary material, we first discuss the network architecture details of the range prediction module, DeformNet, and the discriminator network (Sec. S.2), and also the training details (Sec. S.3). We then present the details and ablation studies of the disentanglement loss $\mathcal{L}_{disen}$ (Sec. S.4). Moreover, we evaluate the impact of the numbers of control points and the output meta-handles (Sec. S.5) and also discuss the difference between our differentiable-renderer-based 2D discriminator and 3D discriminator (Sec. S.6). We also examine the effectiveness of our deformation generative model when it is used for data augmentation (Sec. S.7). Lastly, we provide more results of the target-driven deformation and show learned meta-handles for the laptop category (Sec. S.8).

S.1. Animations of the Learned Meta-Handles

We show the animations of the learned meta-handles in our webpage. Chrome browser is preferred for the best display. On the webpage, each row shows deformations of meta-handles with the *same* index for different shapes. Note that the learned meta-handles are *consistent* across the shapes. The animations demonstrate that our learned meta-handles properly factorize the plausible deformation space of the shape while each of them corresponds to an intuitive deformation direction.

S.2. Network Architecture

In this subsection, we describe the architectures of the range prediction module, DeformNet, and the discriminator network, which are introduced in Sec. 3.2 in the paper.

Range Module. As shown in Fig. S2, after predicting the meta-handles, the range module predicts a coefficient range $[L_i, R_i]$ for each meta-handle. It takes the rest positions of the control points, 64-dimensional control point features (predicted by MetaHandleNet), and the predicted meta-handles as input. The module incorporates the information by building a 3D tensor, where each pair of meta-handle and control point has a 70-dimensional feature. The module then applies an MLP to the 70-dimensional features, resulting in a 2-dimensional feature for each pair of meta-handle and control point. The module then utilizes a max-pooling to aggregate the information across all the control points, resulting in a $m \times 2$ matrix. We then reverse the sign of the first column (due to the max-pooling) to output the final coefficient ranges.

DeformNet. After MetaHandleNet predicts a set of meta-handles with the corresponding coefficient ranges for the source shape, DeformNet finds a coefficient vector within the deformation space so that the deformed source shape matches the target shape. Fig. S1 shows the architecture of DeformNet. It first utilizes PointNet [5] to process both the source and target point clouds to obtain the global features for the source and target shapes. The global features are repeated for the control points and are then combined with the 64-dimensional control point features (predicted by MetaHandleNet) and the rest positions of the control points, resulting in a 2,115-dimensional feature for each control point. The features are fed into an MLP to output a 128-dimensional feature for each control point. We then create a 3D tensor to incorporate the control point features, predicted meta-handles, and the rest positions of the control points. In this 3D tensor, each pair of meta-handle and control point has a 134-dimensional feature. We then apply an MLP to the features to output a 128-dimensional feature for each pair of meta-handle and control point. A max-pooling is then applied to aggregate the features across all the control points, resulting in a 128-dimensional feature for each meta-handle. The features are then combined with the predicted coefficient ranges and are fed into another MLP (with Sigmoid as the final activation function) to output a ratio within $[0, 1]$ for each meta-handle. With both the ratios and the coefficient ranges, we output a coefficient vector within the ranges to represent the deformation.

Discriminator. We utilize a relatively simple 2D network as the discriminator network to match the capability of the deformation generation part. Specifically, the discriminator network takes a 128×128 image as input and uses three

[1] http://cseweb.ucsd.edu/~mil070/deep_meta_handles_supp_animations

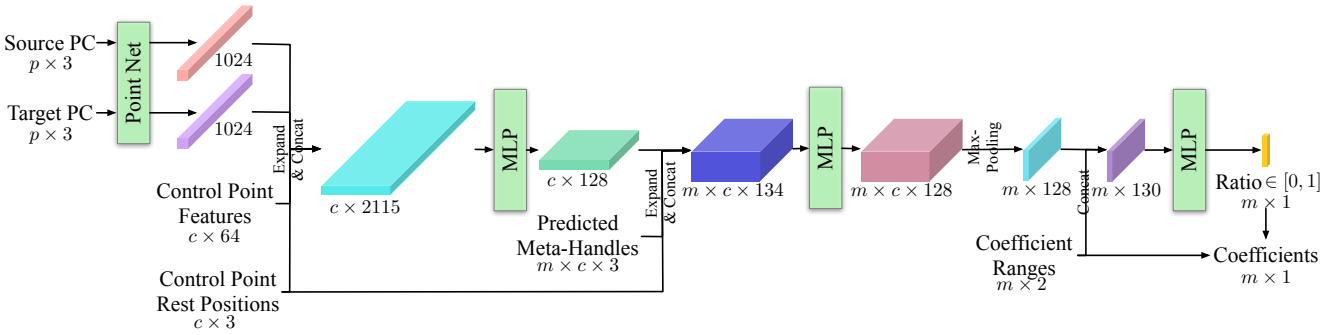

Figure S1: Architecture of DeformNet.

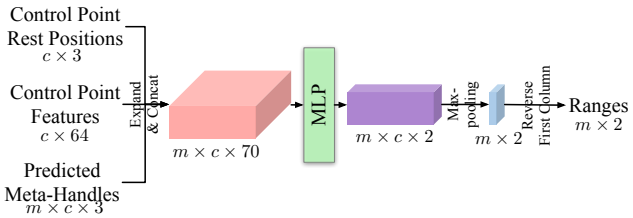

Figure S2: Architecture of the range prediction module.

convolutional layers to process. Each convolutional layer is followed by batch normalization and LeakyReLU. A fully connected layer, along with the sigmoid function, is then utilized to output the final probability.

S.3. Training Details

As described in Sec. 4.1, we use public code to convert the mesh to tetrahedral mesh and then calculate the biharmonic coordinates. We found that the mesh conversion and the coordinate computation are robust even to the shapes with thin parts and complicated topology. Note that we have an example with a complex wire structure in the 9th column of Fig. 6. In the network training, however, we also find that pruning some shapes that have large biharmonic coordinates is helpful for faster convergence. We removed 10% of such shapes in the ShapeNet dataset.

We trained our models on 3 Nvidia RTX 2080 Ti GPUs for 2.8×10^4 iterations (i.e., 1.1×10^6 pairs) with a batch size of 39. Adam is used as the optimizer with a learning rate of 1e-4. All loss terms have an individual weight, and we empirically select the weights. For chair category, the weights are set to 1, 1, 0.1, 3, 6e-3, 1e-3, 1e-3, 1e-3, and 0.3 for $\mathcal{L}_{fit}$, $\mathcal{L}_{symm}$, $\mathcal{L}_{nor}$, $\mathcal{L}_{Lap}$, $\mathcal{L}_{adv}$, $\mathcal{L}_{sp}$, $\mathcal{L}_{cov}$, $\mathcal{L}_{ortho}$, and $\mathcal{L}_{SVD}$ respectively.

S.4. Details of the Disentanglement Loss $\mathcal{L}_{disen}$

Here, we introduce the four terms of the disentanglement loss $\mathcal{L}_{disen}$ (Eq. 5 in the paper).

$\mathcal{L}_{sp}$ is a sparsity loss that encourages the meta-handles $\mathbf{M}_i$ and the coefficient vector $\mathbf{a}$ to be sparse by penalizing their $l1$-norm:

$$\mathcal{L}_{sp} = \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{M}_i\|_1 + \|\mathbf{a}\|_1, \tag{S.1}$$

where m is the number of the meta-handles.

$\mathcal{L}_{cov}$ is a covariance penalty loss introduced by Aumentado-Armstrong *et al.* [1] that encourages meta-handles to be independent with each other. This loss calculates the covariance matrix of the coefficients $\mathbf{a}$ for each batch and penalizes the $l1$-norm of the matrix:

$$\mathcal{L}_{cov} = \|\operatorname{cov}(\mathbf{a}, \mathbf{a})\|_1. \tag{S.2}$$

$\mathcal{L}_{ortho}$ is an orthogonality loss that encourages the meta-handles to cover different coordinates of control point offsets. It is calculated as:

$$\mathcal{L}_{ortho} = \sqrt{\sum_{i \neq j} \|\mathbf{M}_i \circ \mathbf{M}_j\|_{1,1}^2}, \tag{S.3}$$

where '$\circ$' denotes element-wise multiplication. Intuitively, if two meta-handles have no overlap over the offset coordinates, we regard them to be "orthogonal" and they have zero contribution to $\mathcal{L}_{ortho}$.

Lastly, $\mathcal{L}_{SVD}$ is an SVD loss that encourages the control points to translate along with similar directions within each meta-handle. Specifically, for each meta-handle $\mathbf{M}_i$, we regard the control-point offsets of $\mathbf{M}_i$ as c points in the 3D space. Given the points, we find the best-fit plane and then calculate the sum of the distances from the points to the plane, which is equal to $\sigma_3(\mathbf{M}_i^T \mathbf{M}_i)$ and σ_3 indicates the smallest singular value of the matrix. $\mathcal{L}_{SVD}$ is defined as minimizing the distances:

$$\mathcal{L}_{SVD} = \frac{1}{m} \sum_{i=1}^{m} \sigma_3(\mathbf{M}_i^T \mathbf{M}_i). \tag{S.4}$$

Table S1 shows the quantitative comparison for the disentanglement loss $\mathcal{L}_{disen}$. We find that after removing

Table S1: Quantitative comparison between without and with the disentanglement loss $\mathcal{L}_{disen}$. The network is trained on the chair category.

	w/o $\mathcal{L}_{disen}$	w/ $\mathcal{L}_{disen}$
$\mathcal{L}_{sp} \downarrow$	13.355	7.7468
$\mathcal{L}_{cov} \downarrow$	3.5890	1.3229
$\mathcal{L}_{ortho} \downarrow$	10.843	4.2842
$\mathcal{L}_{SVD} \downarrow$	0.1063	0.0002

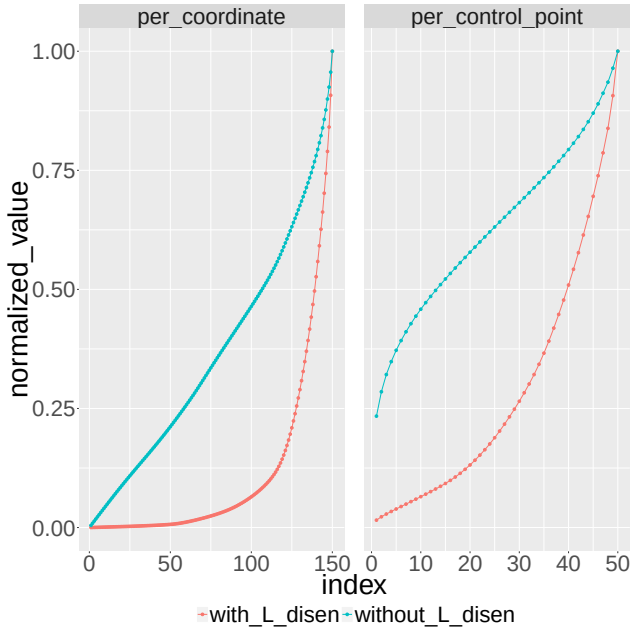

Figure S3: The impact of $\mathcal{L}_{disen}$ on the sparsity of meta-handles.

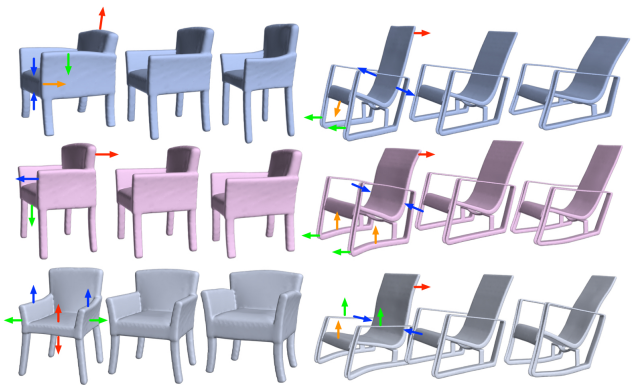

Figure S4: Results without $\mathcal{L}_{disen}$: the figure shows six meta-handles of two shapes, with arrows highlight the deformations. Each meta-handle corresponds to multiple deformations, and there are overlapping between the meta-handles.

$\mathcal{L}_{disen}$, all the four terms increase a lot, which indicates that $\mathcal{L}_{disen}$ is essential for the proper factorization of the deformation space.

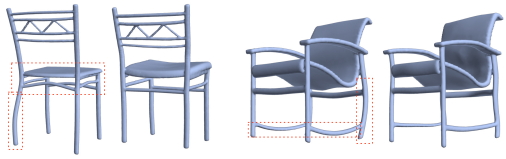

Figure S5: In each pair, the left one shows the deformation with a 3D point cloud discriminator, while the right one shows the deformation with our 2D discriminator.

Fig. S3 further illustrates the impact of $\mathcal{L}_{disen}$ on the sparsity of meta-handles. For the per-coordinate case, we show the distribution of offsets of 50×3 coordinates. For the per-control-point case, we first calculate the $l2$-norm of the offsets for each control point and then show the distribution of the $l2$-norms. For both cases, the values are normalized within each meta-handle and averaged across all the meta-handles and shapes. As shown in the figure, when $\mathcal{L}_{disen}$ is applied, each meta-handle tends to be sparse, and only a small part of coordinates (control points) are impacted. When $\mathcal{L}_{disen}$ is not applied, however, the meta-handles are no longer sparse and tend to deform most of the control points.

Fig. S4 shows more results when $\mathcal{L}_{disen}$ is ablated. The results demonstrate the importance of $\mathcal{L}_{disen}$ for the proper factorization of the deformation space.

S.5. Impact of the Numbers of Control Points and Meta-Handles

We evaluate the impact of the number of control points and meta-handles on the deformation. Please note that here the number of meta-handles indicates the upper bound of the size since the network can use part of them by setting the ranges to zero. Table S2 shows the quantitative results of the chair category. As shown in the table, when we decrease the number of control-point handles from 50 to 25, both the Chamfer distance and cotangent Laplacian increase a little bit, since the degree of freedom of the deformation drops. However, when we increase the number of control-point handles from 50 to 100 and 200, both the Chamfer distance and cotangent Laplacian increase a lot. There are two possible reasons: (a) it is more difficult for the network to deform shapes with such a large number of control points; (b) due to the GPU memory bottleneck, we have to reduce the batch size during training when there is a large number of control points (we reduced the batch size from 39 to 12 when increasing the number of control points from 50 to 200). As for the meta-handles, when we double the number of meta-handles, the Chamfer distance is similar, which indicates that 15 meta-handles are already enough to produce flexible deformations for the chair category. However, the cotangent Laplacian becomes worse, which sug-

gests that it is more difficult for the network to handle lots of meta-handles, and they may introduce some unnecessary distortions.

Table S2: Impact of the numbers of control points and meta-handles. The last row is the model used in the rest of the experiments. 'CD' indicates Chamfer distance, and 'CotLap' indicates cotangent Laplacian.

# Control Points	# Meta-Handles	CD ↓	CotLap ↓
25	15	0.0644	0.5955
100	15	0.0715	0.9101
200	15	0.0758	0.8777
50	30	0.0621	0.8948
50	15	0.0628	0.5751

S.6. Comparison with PointNet-based 3D Discriminator

While we leverage a differentiable renderer and a 2D network for the discriminator, one can consider directly feeding the 3D deformed shape to a 3D processing network. To compare our discriminator with the case of directly processing 3D, we implemented another discriminator using PointNet [5] and fed the points sampled over the deformed 3D mesh as input. Figure S5 shows some comparisons. Although the PointNet-based 3D discriminator can also prevent large distortions, we found that our 2D discriminator produces more visually pleasing deformations in practice. This might happen since the 2D discriminator can capture more subtle visual differences in the 2D space comparing with taking only account with the 3D geometry.

S.7. Application: Data Augmentation

Table S3: Data augmentation for subcategory classification.

	Test Accuracy
No Augmenation	88.3%
w/o $\mathcal{L}_{adv}$	89.8%
Target-Driven	90.4%
Ours	91.6%

Our method learns a plausible deformation space for the input shape and can thus be used for data augmentation. We evaluate our approach as a tool of data augmentation with a multi-label shape classification task. Specifically, we use ten subcategories of ShapeNet [2] chair models, and each model can belong to multiple subcategories (e.g., armchairs and swivel chairs). We sample 50 ShapeNet chair models as training data and 500 chair models as test data. To balance the data, while sampling, we ensure that each subcategory appears at least five times in the training data and at least 50 times in the test data. We employ PointNet [5] as the classification network and train it with binary cross-entropy loss for each subcategory. We test four different settings: a) training on 50 shapes without data augmentation; b) training on 50×20 augmented shapes, where we utilize our method to randomly generate 20 variants within the deformation space of each shape; c) same with b) but without the adversarial loss $\mathcal{L}_{adv}$ when training our network; and d) training on 50×20 augmented shapes, where we randomly sample 20 targets for each shape and use our target-driven deformation to generate the variants. For all the generated deformations, we keep their original subcategory labels for training. The results are shown in Table S3. The results verify that our method can improve the classification accuracy and also that the adversarial loss $\mathcal{L}_{adv}$ is essential to generate plausible deformations. The target-driven deformation is less effective in sampling all the plausible variants.

S.8. More Results of Target-Driven Deformation and Learned Meta-Handles on Laptops

Figure S6 shows more results of the target-driven deformation, as also shown in Figure 6 in the paper. Figure S7 also shows the learned meta-handles on the laptop category. The results show that our method can learn the articulated motion of the two parts.

References

[1] T. Aumentado-Armstrong, S. Tsogkas, A. Jepson, and S. Dickinson. Geometric disentanglement for generative latent shape models. In *ICCV*, 2019. 2

[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, 2015. 4

[3] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Deep self-supervised cycle-consistent deformation for few-shot shape segmentation. In *Eurographics Symposium on Geometry Processing*, 2019. 5

[4] Haibin Huang, Evangelos Kalogerakis, Siddhartha Chaudhuri, Duygu Ceylan, Vladimir G. Kim, and Ersin Yumer. Learning local shape descriptors from part correspondences with multiview convolutional networks. *ACM Transactions on Graphics*, 2017. 5

[5] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017. 1, 4

[6] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3dn: 3d deformation network. In *CVPR*, 2019. 5

[7] Wang Yifan, Noam Aigerman, Vladimir Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *CVPR*, 2020. 5

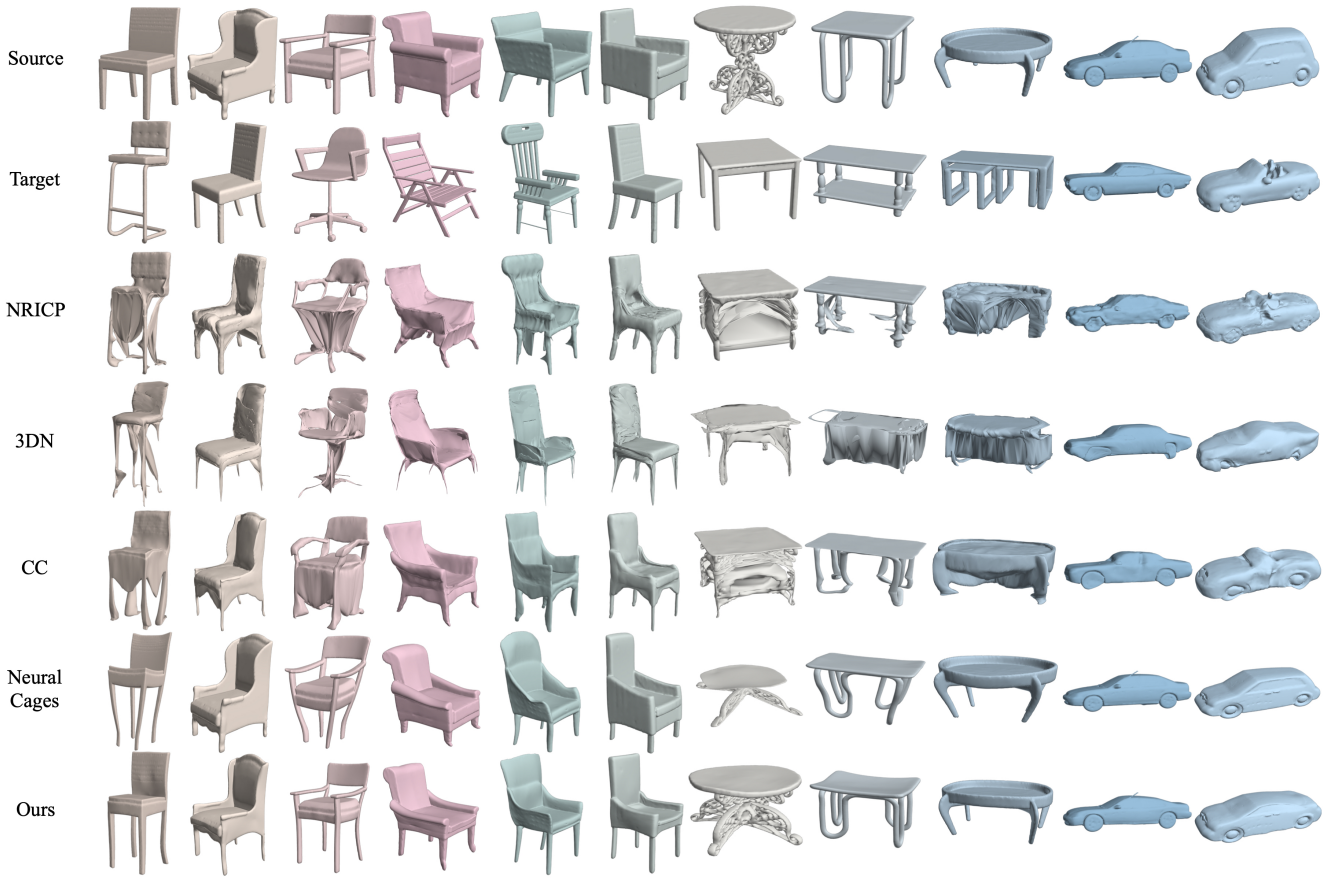

Figure S6: Qualitative comparison of our method with other deformation methods [4, 6, 3, 7]. (More examples of Figure 6 in the paper.)

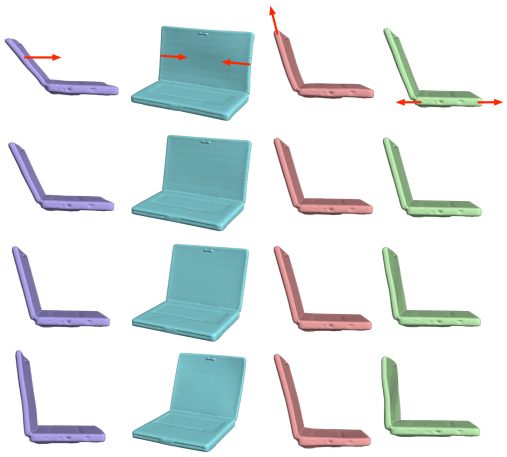

Figure S7: The learned meta-handles for the laptop category. Each column shows the deformations along the direction of a meta-handle.