Supplementary material

Neural Template: Topology-aware Reconstruction and Disentangled Generation of 3D Meshes

CVPR 2022

Overview

This supplementary material consists of the following sections:

- In Section A, we provide more visual comparison results on mesh reconstruction from 3D voxels.
- In Section B, we provide more visual comparison results on mesh reconstruction from single-view images and the detailed quantitative results of Table 2 (main paper) on each category.
- In Section C, we present more visual results on shape generation with controllability.
- In Section D, we present a TSNE visualization on the learned topology space.
- In Section E, we present failure cases that illustrate the limitations.
- In Section F, we present the implementation details.

A. More Visual Comparisons on Mesh Reconstruction from 3D Voxels

Figures 1-2 below show more visual results. Comparing the meshes produced by our DT-Net (d) and others (b-c), we can see that other methods tend to produce less details and miss some of the object parts. In contrast, our method produces more complete objects that are visually the closest to the targets, and our reconstructed objects exhibit more tiny local structures and manifest various object topologies.

Figure 1. Visual comparison for mesh reconstruction from 3D voxels (1/2).

Figure 2. Visual comparison for mesh reconstruction from 3D voxels (2/2).

B. More Visual Comparisons on Mesh Reconstruction from Single-View Images

Figure 3 shows more visual results reconstructed from single-view images. Comparing the meshes produced by our DT-Net (g) and others (b-f), we can see that other methods either are hard to adapt objects of various topologies or tend to produce over-smooth or noisy surfaces with less details. In contrast, our method (g) can produce high-quality meshes, which not only exhibit various topologies but also describe the surface more faithfully with smooth and sharp features simultaneously. The detailed quantitative results of each category are shown in Table 1 on next page.

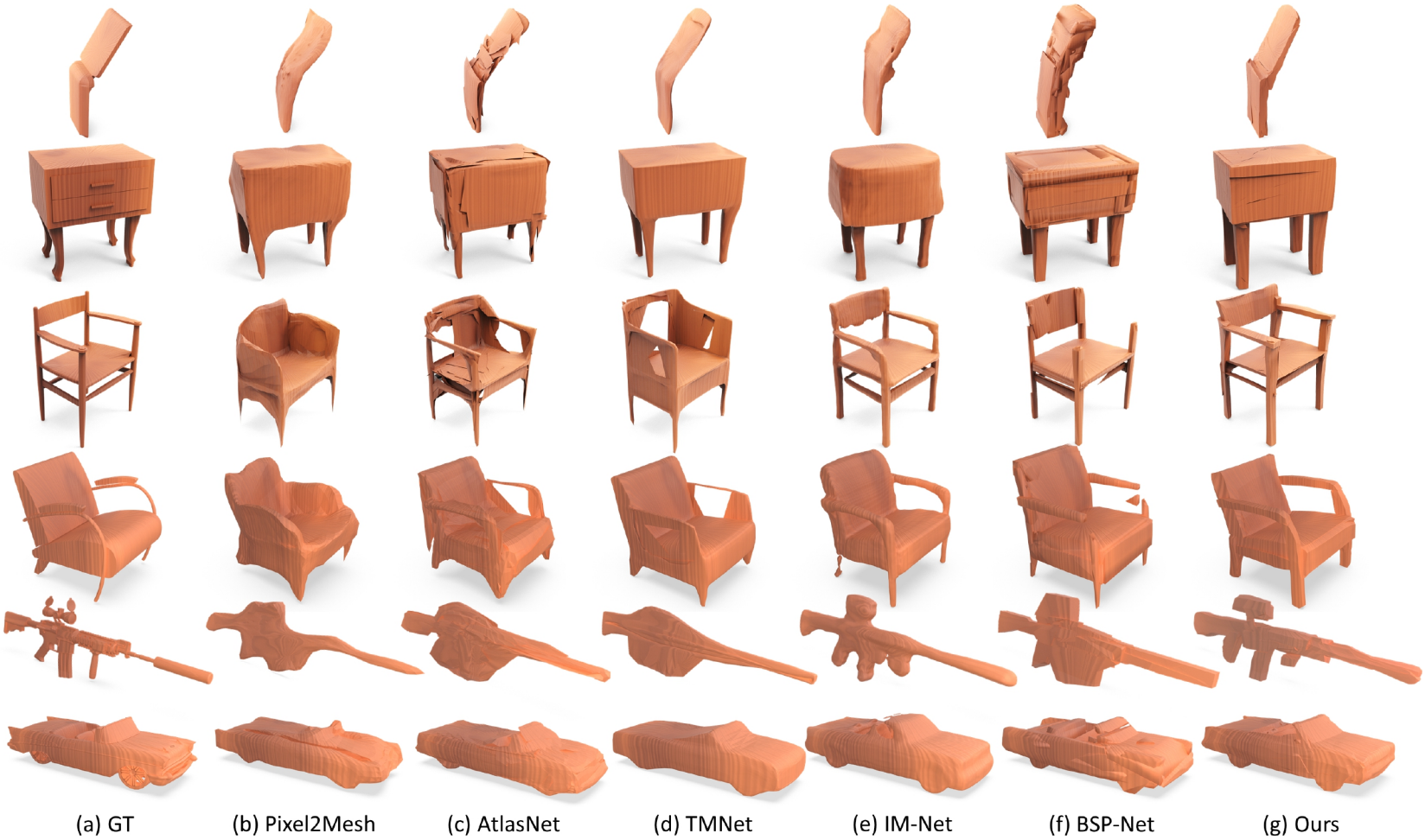

Figure 3. Visual comparison on 3D mesh reconstructions from single-view 2D images.

Table 1. Detailed quantitative results on mesh reconstruction from 2D images for various object categories. Overall, our method is better on LFD and comparable with others on distance metrics P2F and CD.

Metric	Method		Categories													
			Mean	Plane	Bench	Cabinet	Car	Chair	Display	Lamp	Speaker	Riffle	Couch	Table	Phone	Vessel
LFD(↓)	Explicit	Pixel2Mesh	4056.2	5661.1	5056.1	2323.2	2895.0	3985.7	3643.2	**6584.6**	2446.8	**6552.2**	2972.8	3665.9	3442.1	**5351.3**
		AtlasNet25	3880.9	**5127.8**	4689.4	1887.1	**2599.3**	3691.3	3401.5	7117.7	2346.9	7352.8	2951.4	3399.0	2681.6	5475.2
		TMNet	**3765.5**	5571.2	**4640.8**	**1865.5**	2715.4	**3305.6**	**3335.9**	6873.0	**2308.4**	6848.9	**2742.6**	**3117.8**	**2554.6**	5530.6
	Implicit	IM-NET(256^3)	3559.2	4846.1	4252.0	1794.8	2391.2	3229.1	3416.8	6934.1	2360.2	6193.6	2732.3	3039.9	2676.5	5259.1
		BSP-NET	3426.5	4708.4	**3991.7**	**1624.6**	**2361.4**	**3065.9**	3091.2	7003.9	**2147.6**	6144.7	**2572.8**	2857.5	2601.3	**5057.0**
		Ours	**3388.3**	**4604.1**	4012.8	1678.4	2373.0	3067.9	**3080.8**	**6644.2**	2237.3	**6002.7**	2630.8	**2787.0**	**2502.8**	5087.5
P2F(↓)	Explicit	Pixel2Mesh	1.903	1.341	1.709	1.972	1.731	2.146	2.033	2.638	2.497	1.374	2.079	1.942	1.371	2.027
		AtlasNet25	1.289	**0.801**	**1.072**	1.235	1.001	1.444	**1.559**	2.466	**1.826**	**0.925**	1.505	**1.288**	**1.062**	1.343
		TMNet	**1.285**	0.821	1.157	**1.192**	**0.929**	**1.415**	1.580	**2.256**	1.828	1.021	**1.423**	1.389	1.101	**1.327**
	Implicit	IM-NET(256^3)	1.422	0.969	1.390	1.368	0.895	1.558	1.898	2.583	2.114	1.196	1.492	1.543	1.199	1.523
		BSP-NET	1.354	0.964	1.265	**1.118**	0.888	1.521	**1.759**	2.726	**1.786**	1.229	**1.415**	1.337	1.235	1.615
		Ours	**1.294**	**0.818**	**1.146**	1.174	**0.878**	**1.474**	1.821	**2.357**	1.939	**1.055**	1.491	**1.319**	**1.027**	**1.395**
CD(↓)	Explicit	Pixel2Mesh	1.855	1.314	1.793	1.569	1.418	2.112	2.088	3.774	2.691	1.278	1.700	1.964	0.948	1.946
		AtlasNet25	**1.041**	**0.397**	**0.856**	0.955	0.624	**1.059**	1.432	3.028	**2.320**	**0.520**	**1.091**	**1.155**	**0.609**	**0.806**
		TMNet	1.149	0.534	0.998	**0.933**	**0.608**	1.099	**1.431**	**2.944**	2.564	0.761	1.265	1.339	1.073	1.028
	Implicit	IM-NET(256^3)	1.497	0.770	1.358	1.353	0.799	1.405	2.211	3.713	2.589	0.946	1.642	1.841	1.241	**1.379**
		BSP-NET	1.478	0.713	1.367	1.190	0.866	1.385	1.927	**4.182**	**2.586**	0.878	**1.629**	1.646	1.534	1.420
		Ours	**1.396**	**0.618**	**1.257**	**1.119**	**0.648**	**1.273**	**1.715**	4.398	2.734	**0.856**	1.789	**1.539**	**0.825**	1.512

C. More results on Shape Generation with Controllability

Our DT-Net learns a disentangled representation of topology and shape, thus enabling novel forms of 3D object manipulations. Figures 4-7 show more generation results with high-level controllability.

Figure 4. More results of remixing the shape and topology of multiple objects. In each table, objects on top provide the shape codes, whereas objects on the leftmost side provide the topology codes. The remaining objects are produced by remixing the corresponding shape and topology codes.

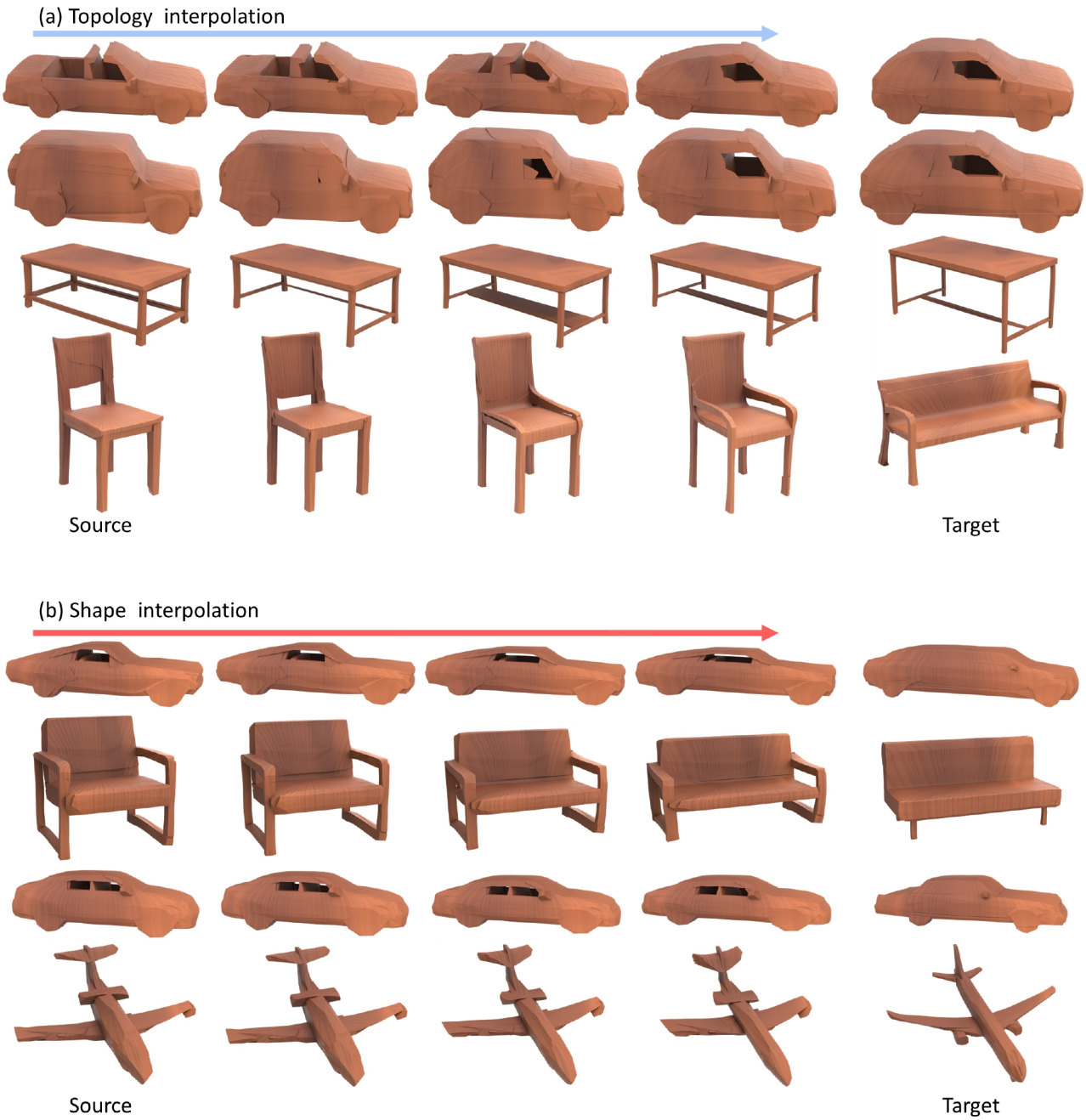

Figure 5. More results of object interpolation from source (left) to target (right), separately on topology (top) and on shape (bottom). Note the smooth transitions achieved by DT-Net (from left to right).

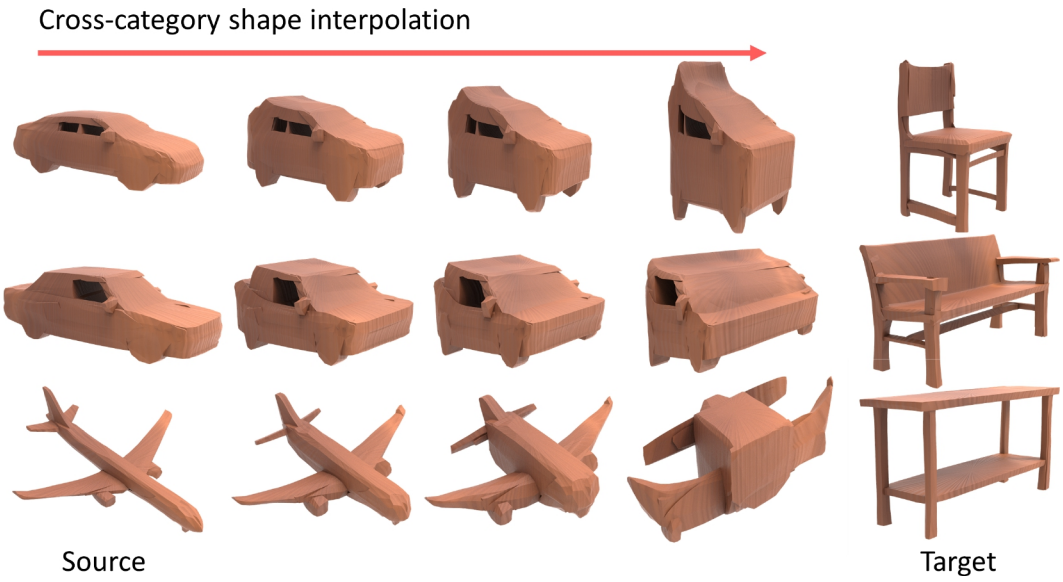

Figure 6. Interpolation between objects of different categories.

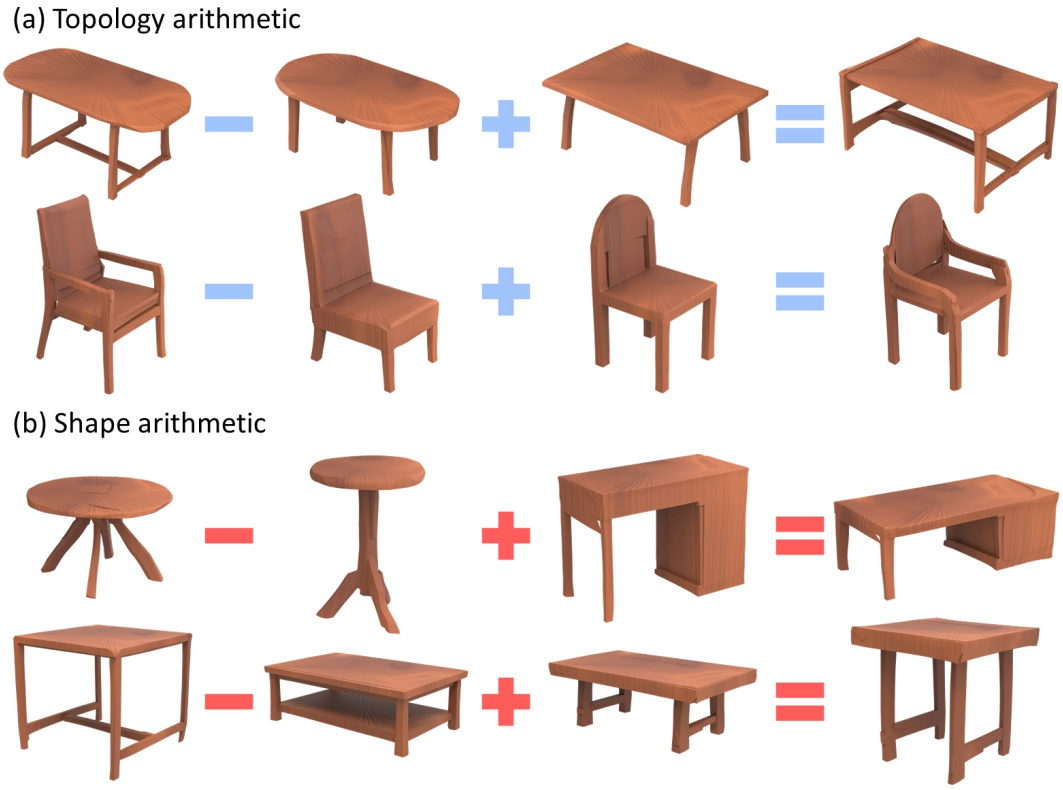

Figure 7. More results on arithmetic operations between different objects.

D. TSNE Visualization

To reveal the smoothness and meaningfulness of the learned topology space, we produce a TSNE embedding visualization for $\mathbf{Z}_T$ on chairs. As shown in Figure 8 below, we can see that DT-Net can learn a smooth embedding space for objects of varying topological structures, in which objects of similar topologies are closely clustered.

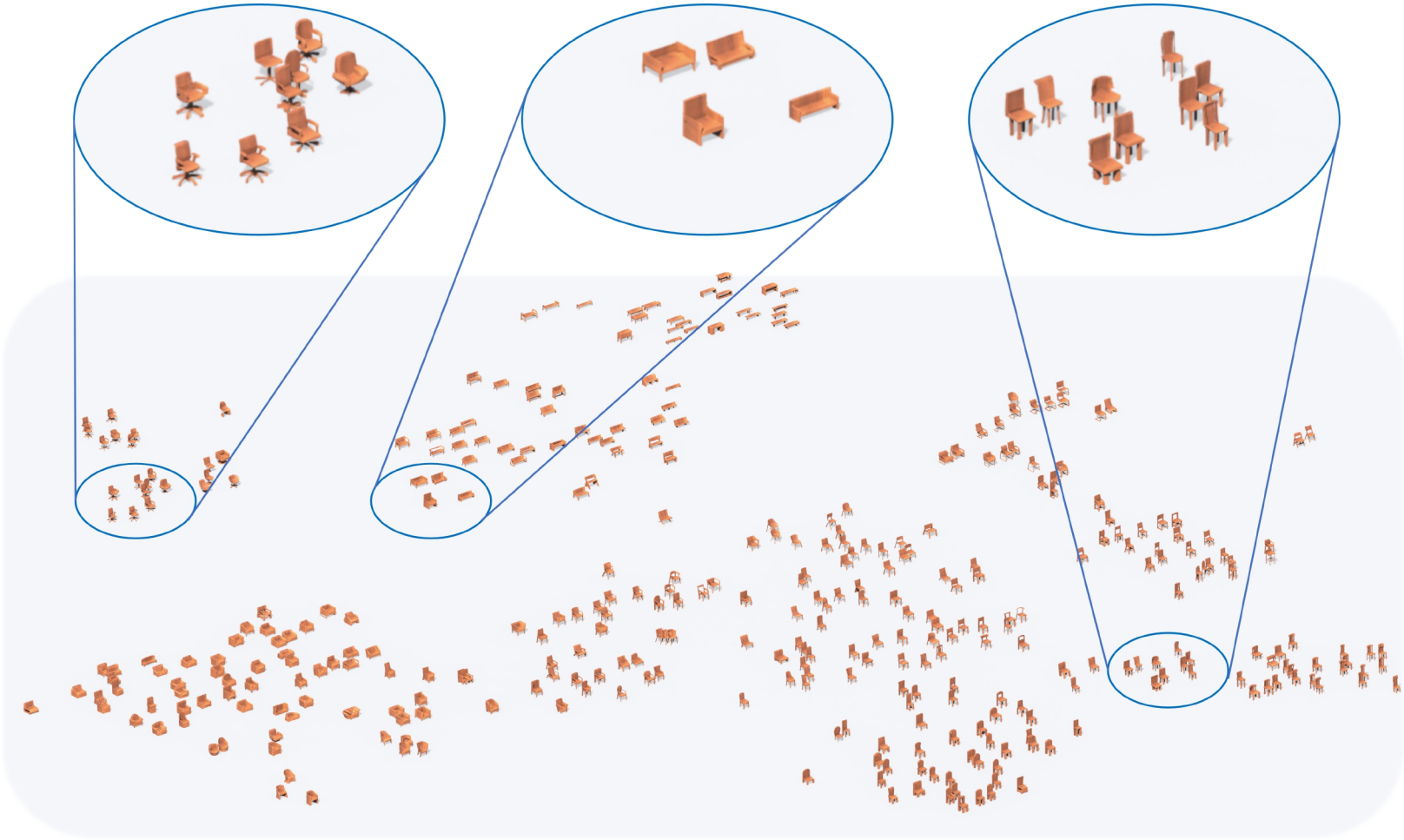

Figure 8. TSNE visualization of the topology codes for chairs via a TSNE embedding. We put each reconstructed object at its code location.

E. Limitations and Failure Cases

Figure 9 shows two failure cases that illustrate the limitations of our current approach. First, it is still very challenging for our method (and also for the existing ones) to reconstruct 3D objects of very complex topologies and fine structures. Also, our current approach does not consider parts information in the shape generation. Thus, in the future, we aim to further formulate the topology-aware neural template in a hierarchical manner and deform it in a part-wise manner for fine-grained 3D reconstructions.

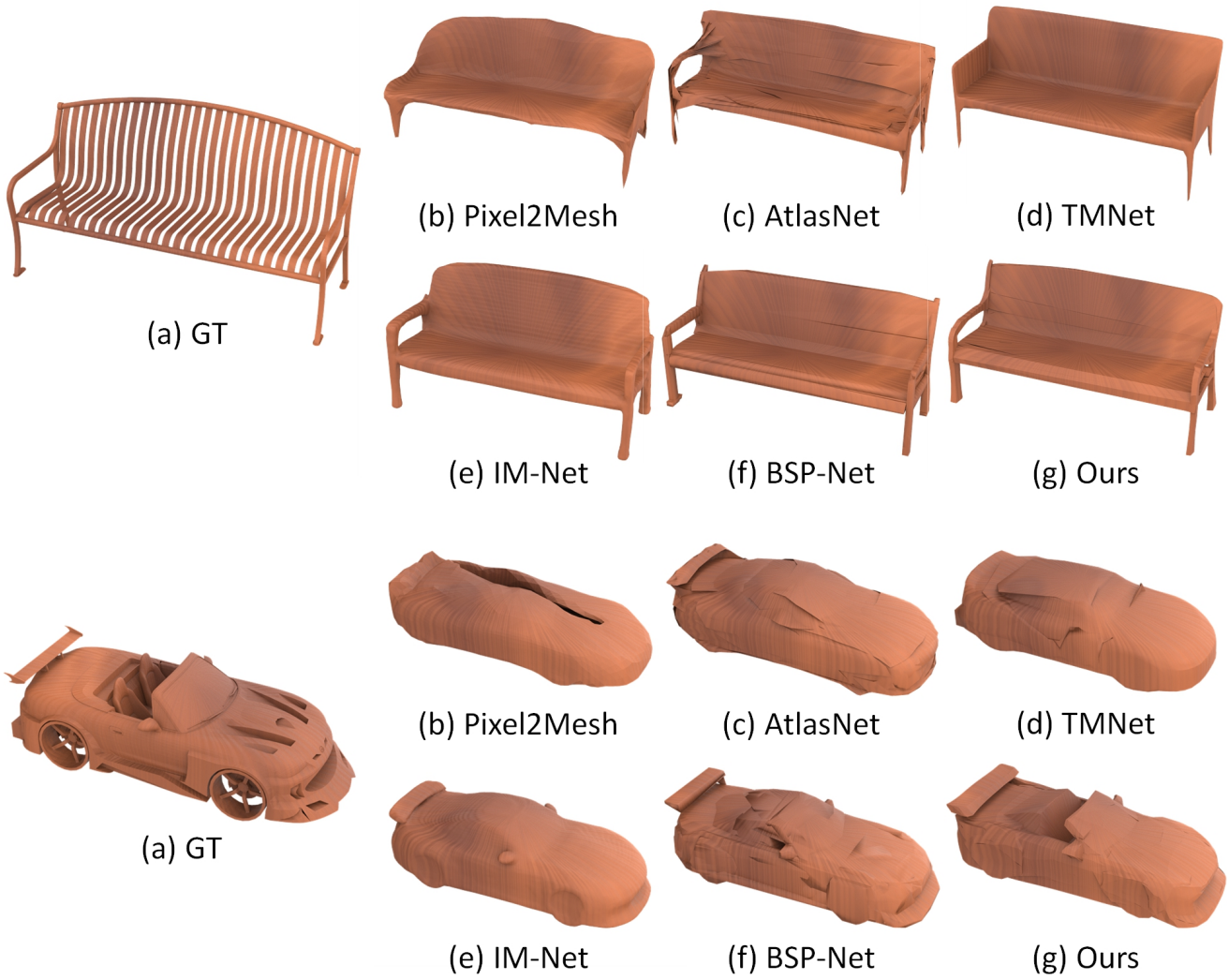

Figure 9. The failure cases.

F. Implementation Details

In this section, we first present the implementation details for the topology formation module (Sec. F.1) and the shape deformation module (Sec. F.2). Then, we present more details on the training (Sec. F.3) and testing (Sec. F.4) of our DT-Net. Lastly, we provide the adoption details for alternative implementations for our framework (Sec. F.5).

F.1. Implementation of Topology Formation Module

The topology formation network first predicts the parameters from the topology code $\mathbf{Z}_T$ to construct various hyperplanes $\mathcal{H} \in R^{N_h * 4}$ (*i.e.*, $ax + by + cz + d = 0$). The network is adopted from [2] and is implemented via a series of MLPs ([256, 512, 1024, $N_h * 4$]) and each of the first three MLP layers is followed by a LeakyReLU activation function.

The planes are then grouped to form a set of convexes $\mathcal{C}$ by a learnable binary matrix $\mathbf{B} \in R^{N_h * N_c}$ (a selective mask) and these convexes are composed to obtain our implicit neural template $\mathcal{T}_I$. After that, we take an occupancy function $O(\mathcal{T}_I, \cdot)$ to calculate whether any point p (x, y, z, 1) is inside or outside. Here, we define $O^{\text{con}}(\mathcal{T}_I, \cdot)$ and $O^{\text{dis}}(\mathcal{T}_I, \cdot)$ as the occupancy functions in the discrete and continuous phase, respectively; please refer to Sec. 3.4 of the main paper.

In particular, $O^{\text{dis}}(\mathcal{T}_I, \cdot)$ first evaluates the distance from p to each hyperplane, and mask out the hyperplanes that p locates in the negative side through the ReLU activation. Then the occupancy for each convex can be computed through the matrix multiplication with the binary mask $\mathbf{B}$, and the minimum operation would then take the union of the convexes. The definition of $O^{\text{dis}}(\mathcal{T}_I, \cdot)$ is given below:

$$O^{\text{dis}}(\mathcal{T}_I, p) = \min_{i \in N_c}(\mathbf{B}^T_{[i,:]} relu(\mathcal{H}p)) \quad \begin{cases} > 0 & \text{outside} \\ = 0 & \text{inside} \end{cases}, \tag{1}$$

where $relu(\cdot)$ is the ReLU activation function. To ease the training difficulty [2], we also introduce a continuous approximation $O^{\text{con}}(\mathcal{T}_I, \cdot)$ and a sum matrix $\mathbf{W} \in R^{N_c}$ as

$$O^{\text{con}}(\mathcal{T}_I, p) = clip[\sum_{i \in N_c}(clip[1 - \mathbf{B}^T_{[i,:]} relu(\mathcal{H}p)]_{0,1})]_{0,1} \quad = \begin{cases} 1 & \text{inside} \\ [0, 1] & \text{outside} \end{cases} \tag{2}$$

$$\approx clip[\sum_{i \in N_c}(\mathbf{W}_i * clip[1 - \mathbf{B}^T_{[i,:]} relu(\mathcal{H}p)]_{0,1})]_{0,1} \quad \text{if } \mathbf{W}_i \approx 1 \ \ \forall i \ , \tag{3}$$

where $clip[\cdot]$ clips the value in range [0,1]. During the initial training, we introduce $\mathbf{W} = 0$ to smooth the gradient flow in the early stage, and $\mathbf{W}$ would be promoted to equal to 1 within a few epochs.

F.2. Implementation of Shape Deformation Module

Inspired by [1], we adopt the invertible deformation function g as a solution of a first-order Ordinary Differential Equation (ODE), where the velocity function is parametrized by a deep neural network $\hat{g}(\mathbf{Z}_S, \cdot)$ conditioned on the shape code $\mathbf{Z}_S$. The solution of the ODE can be expressed in the form $p_T = g(\mathbf{Z}_S, p_0) = p_0 + \int_0^T \hat{g}(\mathbf{Z}_S, p_t)dt$. By the existence and uniqueness theorem of Ordinary Differential Equation proven in [3], there exists unique solution for the ODE under the mild condition that $\hat{g}(\mathbf{Z}_S, \cdot)$ satisfies the Lipschitz continuous condition. This ensures that the deformation function is a diffeomorphism and preserves the topology of the neural template.

For memory saving, we define the deformation vector field $\hat{g}(\mathbf{Z}_S, \cdot)$ conditioned on a latent vector similar to [10, 4]. We first transform both the shape code $\mathbf{Z}_S$ and the input coordinates into two fixed dimension vectors by a single-layer perceptron, respectively, and apply an element-wise product to them. The multiplied hidden vectors are then transformed to a deformation direction in R^3 with three layers of MLPs (*i.e.*, [256, 256, 256]) and each layer is followed by an ReLU activation function.

F.3. Training Details

We leverage two auto-encoders for image and voxel feature extraction [2] followed by MLPs for extracting the topology code and shape code (each of 128 dimensions). When training the mesh reconstruction process from voxels, we adopt a progressive training strategy with occupancy pairs sampled in increasing resolutions (16^3, 32^3, 64^3), aiming for a speedup in convergence [2]. For each resolution, we train the network for 300 epochs in the continuous phase. In the discrete phase, we train another 300 epochs in the resolution of 64^3 directly.

With the sampled occupancy pairs $\hat{\mathcal{M}}_I = \{p'_i, o_i\}_{i=1}^{N_p}$ and the inversely-mapped point in the topology space as $p_i = g^{-1}(\mathbf{Z}_S, p'_i)$, the loss function for continuous phase consists of three main terms:

$$\mathcal{L}^{\text{con}} = \mathcal{L}^{\text{con}}_{\text{align}} + \mathcal{L}^{\text{con}}_{\mathbf{B}} + \mathcal{L}^{\text{con}}_{\mathbf{W}} \ , \tag{4}$$

where $\mathcal{L}^{\text{con}}_{\text{align}}$ promotes an accurate prediction of the continuous occupancy $O^{\text{con}}(\mathcal{T}_I, \cdot)$ in a least-squares manner:

$$\mathcal{L}^{\text{con}}_{\text{align}} \ = \ \frac{1}{N_p} \sum_{i=1}^{N_p} \left(O^{\text{con}}(\mathcal{T}_I, p_i) - o_i\right)^2 \ ;$$

$\mathcal{L}^{\text{con}}_{\mathbf{B}}$ promotes and keeps the binary selection matrix in range [0, 1]:

$$\mathcal{L}^{\text{con}}_{\mathbf{B}} \ = \ \sum_{i=1}^{N_h} \sum_{j=1}^{N_c} max(0, \mathbf{B}_{i,j} - 1) - min(0, \mathbf{B}_{i,j}) \ ;$$

and $\mathcal{L}^{\text{con}}_{\mathbf{W}}$ encourages the sum matrix $\mathbf{W}$ to converge to a direct sum where $\mathbf{W} = 1$, this term vanishes empirically within the first few epochs:

$$\mathcal{L}^{\text{con}}_{\mathbf{W}} \ = \ \sum_{i=1}^{N_c} |\mathbf{W}_i - 1| \ .$$

The loss term for the discrete phase contains two major parts:

$$\mathcal{L}^{\text{dis}} = \mathcal{L}^{\text{dis}}_{\text{align}} + \mathcal{L}^{\text{dis}}_{\mathbf{B}} \ , \tag{5}$$

where $\mathcal{L}^{\text{dis}}_{\text{align}}$ ensures the correctness of the discrete occupancy $O^{\text{dis}}(\mathcal{T}_I, \cdot)$:

$$\mathcal{L}^{\text{dis}}_{\text{align}} \ = \ \frac{1}{N_p} \sum_{i=1}^{N_p} \left[o_i * \max(O^{\text{dis}}(\mathcal{T}_I, p_i), 0) + (1 - o_i) * (1 - \min(O^{\text{dis}}(\mathcal{T}_I, p_i), 1)) \right] \ ;$$

and $\mathcal{L}_{\mathbf{B}}^{\text{dis}}$ enforces the binary selection matrix to be either close to 0 or 1, with hyper-parameter λ=0.01:

$$\mathcal{L}_{\mathbf{B}}^{\text{dis}} = \sum_{i=1}^{N_h}\sum_{j=1}^{N_c}[\mathbf{B}_{i,j} < \lambda](|\mathbf{B}_{i,j}|) + [\mathbf{B}_{i,j} \geq \lambda](|1 - \mathbf{B}_{i,j}|) \ .$$

For reconstructions from single-view images, we reuse the pre-trained model from voxels and further train the image encoder (ResNet [5]) for 1,000 epochs to predict the latent codes. The batch size of all training phases is set as 32 and the learning rate is 0.0005 with the Adam optimizer [6], while the number of hyperplanes $N_h = 4096$ and convexes $N_c = 32$. The training of the auto-encoding task for the continuous and discrete phases in total takes around 7 days on a cluster of eight RTX3090 GPUs, and the training of the image encoder takes around 12 hours.

F.4. Testing Details

After training the DT-Net, we can obtain the topology mesh $\mathcal{T}_E$ as a union of the learned convexes. To obtain the final mesh $\mathcal{M}_E$, a straightforward way is to directly apply the deformation g to each vertex of $\mathcal{T}_E$; however, this might lose the geometric details as the shape refinement module learns a continuous deformation on all spatial locations simultaneously. For simplicity, we address this by uniformly subdividing the topology template before applying the deformation. Specifically, we subdivide each face of $\mathcal{T}_E$ up to five times to obtain more vertices. By then, we can obtain $\mathcal{M}_E$ with continuous and smooth features from the deformation procedure.

F.5. Alternative implementations for Topology Formation Module and Shape Deformation Module

We have explored the usage of alternative primitives to compose our neural topology-aware templates, including cuboids [9] and superquadrics [8]. For the cuboid template, we first use multi-layer perceptions to predict a set of scale vectors $s \in R^{N*3}$, a set of quaternions $q \in R^{N*4}$ for rotation, and a set of translation vectors $v \in R^{N*4}$, where N is the number of cuboids ($N = 32$). For the superquadric template, we also predict a set of scale vectors $s \in R^{M*3}$, a set of rotation vectors represented by quaternions $q \in R^{M*4}$, a set of translation vectors $v \in R^{M*4}$, and a set of norm vectors $e \in R^{M*3}$ for the smoothness of superquadrics, where M is the number of superquadrics ($M = 64$).

For calculating the occupancy, we first transform a query point $p(x, y, z)$ into the canonical space of the i^{th} cuboid or superquadric, denoted as d^i:

$$d^i = R_i^T(p - v_i) \tag{6}$$

where R_i is the rotation matrix produced from q_i (*i.e.*, the i^{th} quaternion).

The occupancy calculation in terms of the cuboid representation can be expressed as

$$O^{\text{cuboid}}(\mathcal{T}_I, p) = \sigma(-\frac{1}{\delta}\text{LogSumExp}(-\delta \max_{j\in 1,2,3}(|d_j^i| - s_i))) \quad \begin{cases} \geq 0.5 & \text{outside} \\ < 0.5 & \text{inside} \ . \end{cases} \tag{7}$$

The occupancy calculation in term of the superquadric representation can be expressed as

$$O^{\text{super}}(\mathcal{T}_I, p) = \sigma(-\frac{1}{\delta}\text{LogSumExp}(-\delta(\sum_{j\in 1,2,3}(\frac{d_j^i}{s_i})^{\frac{2}{e_j}} - 1))) \quad \begin{cases} \geq 0.5 & \text{outside} \\ < 0.5 & \text{inside} \ . \end{cases} \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid activation; LogSumExp computes a soft minimum of the occupancy values among all the primitives; and δ is a parameter for controlling the hardness of the minimum. We empirically set δ as 100.0.

With these occupancy functions, the least-squares loss is used to promote the accurate prediction for the ground-truth occupancy.

When using invertible neural network (INN) [7] to implement our shape deformation module, we directly adopt the default settings; please find more details in the original paper [7].

Figure 10 shows another example to use other primitives, including superquadrics (a) and cuboids (b), vs. our convexes (d) for composing the topology-aware neural templates and INN [7] (c) to implement the shape deformation module; see the qualitative results in the table below.

Metric	(a) Ours with Superquadrics	(b) Ours with Cuboids	(c) Ours with INN	(d) Ours with Convexes
LFD ↓	5369.2	2857.7	2681.2	**2368.7**
P2F ↓	1.544	1.459	1.105	**0.847**
CD ↓	4.281	0.963	0.808	**0.581**

Figure 10. Visualization of another example for alternative implementations, and the corresponding evaluation results for all chairs.

References

[1] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. In *NeurIPS*, pages 6571–6583, 2018. 11

[2] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-Net: Generating compact meshes via binary space partitioning. In *CVPR*, pages 45–54, 2020. 11, 12

[3] Earl A. Coddington and Norman Levinson. *Theory of Ordinary Differential Equations*. 1955. 11

[4] Kunal Gupta and Manmohan Chandraker. Neural Mesh Flow: 3D manifold mesh generation via diffeomorphic flows. In *NeurIPS*, pages 1747–1758, 2020. 12

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 13

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 13

[7] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural Parts: Learning expressive 3D shape abstractions with invertible neural networks. In *CVPR*, pages 3204–3215, 2021. 14

[8] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics Revisited: Learning 3D shape parsing beyond cuboids. In *CVPR*, pages 10344–10353, 2019. 13

[9] Chunyu Sun, Qianfang Zou, Xin Tong, and Yang Liu. Learning adaptive hierarchical cuboid abstractions of 3D shape collections. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 38(6):241:1–241:13, 2019. 13

[10] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *ICCV*, pages 4541–4550, 2019. 12

The End