Supplementary Material of Marching-Primitives: Shape Abstraction from Signed Distance Function

Weixiao Liu[1,2] Yuwei Wu[1] Sipu Ruan[1] Gregory S. Chirikjian[1]
[1]National University of Singapore [2]Johns Hopkins University
{mpewxl, yw.wu, ruansp, mpegre}@nus.edu.sg

Abstract

In this supplementary material, we provide the details about the derivations, discussions and experiment settings. In Sec. 1, we provide the detailed derivation of the approximation of the SDF of a superquadric. In Sec. 2, the primitive initialization strategy in the connectivity marching step is detailed. In Sec. 3, we provide the derivation of the probabilistic primitive marching step. Furthermore, Sec. 4 elaborates the fail-safe primitive removal criterion. The overview of the Marching-Primitive algorithm is summarized into pseudo-code in Sec. 5. Finally, in Sec. 6, we detail the experiment implementation and show more qualitative examples.

1. Approximation of superquadric SDF

Recall the signed radial distance of a point $\mathbf{x}_i$ to a general-posed superquadric in Eq. (2) of the paper

$$d_{\boldsymbol{\theta}}(\mathbf{x}_i) = \left(1 - f^{-\frac{\epsilon_1}{2}}(g^{-1} \circ \mathbf{x}_i)\right) \|g^{-1} \circ \mathbf{x}_i\|_2 \quad (1)$$

The derivation is as follows (with an illustration shown in Fig.1). The radial distance of a point $\mathbf{x}_i$ to the surface of a superquadric is defined as

$$\|\mathbf{x}_i - \mathbf{q}_i\|_2 \quad (2)$$

$\mathbf{q}_i$ is where the vector from the center of the superquadric frame to $\mathbf{x}_i$ intersects the surface. Therefore, when viewed from the superquadric frame g, the vectors $g^{-1} \circ \mathbf{q}_i$ and $g^{-1} \circ \mathbf{x}_i$ are colinear, *i.e.*

$$g^{-1} \circ \mathbf{q}_i = \alpha(g^{-1} \circ \mathbf{x}_i), \text{ where } \alpha \in \mathbb{R} \quad (3)$$

Note that $g^{-1} \circ \mathbf{q}_i$ lies on the surface of superquadric. Thus, substituting it into the implicit equation of the superquadric (Eq. (1) in the paper), we obtain

$$\alpha = f^{-\frac{\epsilon_1}{2}}(g^{-1} \circ \mathbf{x}_i) \quad (4)$$

Therefore,

$$\begin{aligned} \|\mathbf{x}_i - \mathbf{q}_i\|_2 &= \|g^{-1} \circ \mathbf{x}_i - g^{-1} \circ \mathbf{q}_i\|_2 \\ &= \|(1-\alpha)g^{-1} \circ \mathbf{x}_i\|_2 \end{aligned} \quad (5)$$

Considering the inside/outside of $\mathbf{x}_i$ relative to the superquadric surface, the signed radial distance is thus Eq.(1).

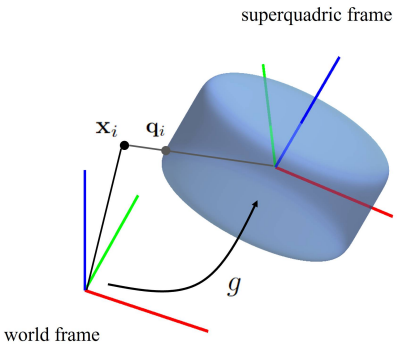

Figure 1. Signed distance approximated by signed radial distance.

2. Primitive Initialization

In Sec. 3.1 of the paper, the geometric primitive is initialized as an ellipsoid for each volume of interest (VOIs) detected in the target SDF. This is achieved by first finding out the smallest bounding-box encompassing the connected voxels that form each of the VOI. For example, the lengths of a bounding-box are l_x, l_x and l_z, and the centroid locates at $\mathbf{x}_c$. Ideally, the primitive is initialized as an ellipsoid centered at $\mathbf{x}_c$ with scales proportional to the lengths correspondingly. Recall that a superquadric $\boldsymbol{\theta}$ is parameterized by

$$\boldsymbol{\theta} = \{\epsilon_1, \epsilon_2, a_x, a_y, a_z, \mathbf{R}, \mathbf{t}\} \quad (6)$$

Then, the initialized ellipsoid is a special case of the superquadrics

$$\boldsymbol{\theta}_{init} = \{1, 1, \gamma l_x, \gamma l_y, \gamma l_z, \mathbf{I}, \mathbf{x}_c\} \quad (7)$$

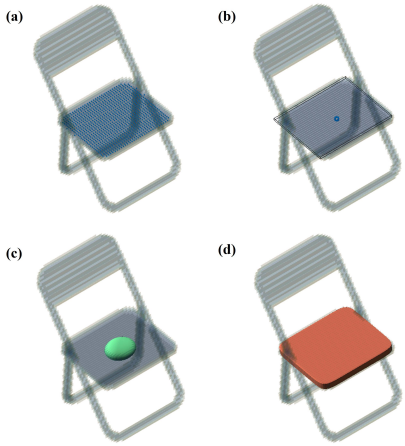

Figure 2. Visualizations of concepts. (a) Blue dots indicate the detected VOI. (b) The smallest bounding-box encompassing the VOI. (c) Initialization of the primitive as an ellipsoid. (d) The final marched superquadric capturing the local geometry.

where γ is the initial scale ratio, $\mathbf{I}$ is the identity rotation matrix. However, if the VOI is nonconvex, the centroid might lie in the exterior space. In this situation, it has the risk of activating the auto-degeneration mechanism. Therefore, instead of $\mathbf{t} = \mathbf{x}_c$, we constrain the initial location $\mathbf{t}$ within the interior of the target shape:

$$\mathbf{t} = \underset{\mathbf{x}_i \in \mathbf{V}, d(\mathbf{x}_i) \leq 0}{\arg\min} \|\mathbf{x}_i - \mathbf{x}_c\|_2 \tag{8}$$

where $\mathbf{V}$ is the current set of voxel points, $d(\mathbf{x}_i)$ is the target signed distance evaluated at voxel point $\mathbf{x}_i$. $\boldsymbol{\theta}_{init}$ works as the initial input to the subsequent probabilistic primitive marching step. The concepts are visualized in Fig.2 for better understanding.

3. Derivation of the Probabilistic Marching

In this section, we provide the detailed derivation of the probabilistic primitive marching in Sec. 3.4 of the paper. Based on the probabilistic model $p\big(d(\mathbf{x}_i)|\boldsymbol{\theta}_k, z_{ik}\big)$ (Eq.8 in the paper), the likelihood of the superquadric parameter $\boldsymbol{\theta}_k$ and variance σ^2 given the target SDF is

$$\begin{aligned} L(\boldsymbol{\theta}_k, \sigma^2) &= \prod_{\mathbf{x}_i \in \mathbf{V}} p(d(\mathbf{x}_i), z_{ik}|\boldsymbol{\theta}_k) \\ &= \prod_{\mathbf{x}_i \in \mathbf{V}} p\big(d(\mathbf{x}_i)|\boldsymbol{\theta}_k, z_{ik}\big) p(z_{ik}|\boldsymbol{\theta}_k) \\ &= \prod_{\mathbf{x}_i \in \mathbf{V}} p_0(\mathbf{x}_i)^{1-z_{ik}} \mathcal{N}\big(d_i|d_{\boldsymbol{\theta}_k}(\mathbf{x}_i), \sigma^2\big)^{z_{ik}} p(z_{ik}) \end{aligned} \tag{9}$$

where

$$p_0(\mathbf{x}_i) = \frac{\mathbb{1}_{d_i \in [-t, 0)}}{t} \quad d_i \doteq d(\mathbf{x}_i) \tag{10}$$

and $p(z_{ik})$ is the prior probability of the correspondence between the ith voxel point and the kth primitive, which is independent of $\boldsymbol{\theta}_k$, *i.e.* $p(z_{ik}|\boldsymbol{\theta}_k) = p(z_{ik})$. As discussed in the paper, we assume that z_{ik} is subjected to a Bernoulli prior distribution $B(p_0)$, *i.e.* $p(z_{ik} = 1) = p_0$. The definitions of other variables can be found in the paper. Our goal is to find the optimal $\boldsymbol{\theta}_k$ and σ^2 that maximize the likelihood function. This is equivalent to minimizing the negative log-likelihood

$$\begin{aligned} l(\boldsymbol{\theta}_k, \sigma^2) &= -\log L(\boldsymbol{\theta}_k, \sigma^2) \\ &= -\sum_{\mathbf{x}_i \in \mathbf{V}} \log \Bigg[p(z_{ik}) p_0(\mathbf{x}_i)^{1-z_{ik}} c(\sigma^2)^{z_{ik}} \\ &\quad \exp\left(-\frac{1}{2} \left(\frac{d(\mathbf{x}_i) - d_{\boldsymbol{\theta}_k}(\mathbf{x}_i)}{\sigma} \right)^2 \right)^{z_{ik}} \Bigg] \\ &= \sum_{\mathbf{x}_i \in \mathbf{V}} \Bigg[\frac{z_{ik}}{2} \left(\frac{d(\mathbf{x}_i) - d_{\boldsymbol{\theta}_k}(\mathbf{x}_i)}{\sigma} \right)^2 - \log p(z_{ik}) \\ &\quad - z_{ik} \log c(\sigma^2) - (1 - z_{ik}) \log p_0(\mathbf{x}_i) \Bigg] \end{aligned} \tag{11}$$

where $c(\sigma^2) = (2\pi\sigma^2)^{-\frac{1}{2}}$ is the normalizing coefficient of the Gaussian distribution. By ignoring the terms independent of $\boldsymbol{\theta}_k$ and σ^2, it is equivalent to minimize

$$l'(\boldsymbol{\theta}_k, \sigma^2) = \sum_{\mathbf{x}_i \in \mathbf{V}} z_{ik} \left[\frac{\big(d(\mathbf{x}_i) - d_{\boldsymbol{\theta}_k}(\mathbf{x}_i)\big)^2}{2\sigma^2} - \log c(\sigma^2) \right] \tag{12}$$

Unlike $d(\mathbf{x}_i)$ which is observed from the target signed distance, the correspondence z_{ik} is a latent variable that cannot be observed. Therefore, it is intractable to solve Eq.12 directly. Our algorithm solves the problem in a two-step expectation-maximization fashion. That is, z_{ik} is replaced with

$$P_{ik} \doteq E\big(z_{ik}|\boldsymbol{\theta}_k, d(\mathbf{x}_i)\big) = p\big(z_{ik} = 1|\boldsymbol{\theta}_k, d(\mathbf{x}_i)\big) \tag{13}$$

Eq.13 is the conditional expectation of z_{ik} given the current estimation of $\boldsymbol{\theta}_k$ and the target SDF, whose value can be calculated by Eq.9 in the paper. Subsequently, we derive that the minimization of Eq.12 is equivalent to Eq.10 in the paper, where we use an adaptive activation subset $\mathbf{V}_a$ instead of the whole voxel space $\mathbf{V}$ to boost performance. After we obtain the updated primitive estimation $\boldsymbol{\theta}_k$, the variance σ^2 of the Gaussian distribution can be updated in closed form by solving

$$\begin{aligned} &\frac{\partial l'}{\partial \sigma^2} = 0 \\ \Leftrightarrow &\sum_{\mathbf{x}_i \in \mathbf{V}_a} P_{ik} \left[\frac{\big(d(\mathbf{x}_i) - d_{\boldsymbol{\theta}_k}(\mathbf{x}_i)\big)^2 - \sigma^2}{2\sigma^4} \right] = 0 \\ \Leftrightarrow \quad &\sigma^2 = \frac{\sum_{\mathbf{x}_i \in \mathbf{V}_a} P_{ik} \big(d(\mathbf{x}_i) - d_{\boldsymbol{\theta}_k}(\mathbf{x}_i)\big)^2}{\sum_{\mathbf{x}_i \in \mathbf{V}_a} P_{ik}} \end{aligned} \tag{14}$$

4. Primitive Removal Criterion

In this section, we detail the fail-safe primitive removal criterion introduced in Sec. 3.5 in the paper. Our method counts the number of positive (exterior), negative (interior), and inactive voxels encompassed by the recovered primitive, which we denote as N_+, N_- and N_0, respectively. The inactive voxels are those already fitted by recovered primitives, which is defined by Eq.7 in the paper. Our algorithm removes the primitive from the representation if

$$N_- < 1 \quad or \quad \frac{N_+}{N_+ + N_- + N_0} \geq 0.5 \tag{15}$$

The first criterion removes the auto-degenerated primitive that shrinks to a point. The second one is a fail-safe checking criterion, which removes the primitive that significantly contradicts the target SDF.

5. Overview of the Algorithm

In this section, we briefly summarize the Marching-Primitives algorithm into a pseudo-code (Algorithm.1) to give an overview of the structure. Note that V_- in the fourth row indicates the sets of voxels with negative signed distance, *i.e.* interior of the shape. The Marching-Primitives can be roughly separated into 2 parts. Firstly, it marches on the signed distance domain (row 5-15) to find VOIs by analysing the connectivity. Then for each VOI, the algorithm continues to march on the voxelized space domain (row 16-24) to grow a primitive capturing the local geometry of the VOI. The algorithm terminates when all the interior volumes are well captured by the primitive representation $\mathbf{\Theta}$.

Algorithm 1 Marching-Primitives

1: **Input:** voxel set $\mathbf{V}$, with target SDF $d(\cdot)$
2: **Output:** primitive set $\mathbf{\Theta}$
3: $\mathbf{\Theta} \leftarrow \{\}$
4: **while** $\mathbf{V}_- \neq \emptyset$ **do**
5: generate marching sequence T^c ▷ Eq.4 in paper
6: **for** t_m^c in T^c **do**
7: **if** $t_m^c >$ termination threshold **then**
8: **return** $\mathbf{\Theta}$
9: **else**
10: calculate VOIs $\bar{S}_m$ ▷ Eq.6 in paper
11: **if** $\bar{S}_m \neq \emptyset$ **then**
12: **break for**
13: **end if**
14: **end if**
15: **end for**
16: **for** $\mathcal{S}_k$ in $\bar{S}_m$ **do**
17: initialize primitive $\boldsymbol{\theta}_k^{init}$ ▷ Eq.2 in supplement
18: **while** not converged **do**
19: march correspondence P_{ik} ▷ Eq.9 in paper
20: update primitive $\boldsymbol{\theta}_k$ ▷ Eq.10 in paper
21: **end while**
22: $\boldsymbol{\theta}_k \rightarrow \mathbf{\Theta}$ **if** $\boldsymbol{\theta}_k$ valid ▷ Eq.10 in supplement
23: $\mathbf{V} = \mathbf{V} - \{\mathbf{x}_i, d(\mathbf{x}_i) \leq 0 \wedge d_{\boldsymbol{\theta}_k}(\mathbf{x}_i) \leq 0\}$
24: **end for**
25: **end while**
26: **return** $\mathbf{\Theta}$

6. Implementation and Additional Results

6.1. Metrics

In this section, we provide details on the two metrics used to evaluate the experiments.

Chamfer L_1-distance: The common Chamfer L_1-distance is defined as follows:

$$\begin{aligned} D_{\text{chamfer}}(\mathbf{X}, \mathbf{Y}) = & \frac{1}{M} \sum_{\mathbf{y}_j \in \mathbf{Y}} \min_{\mathbf{x}_i \in \mathbf{X}} \|\mathbf{y}_j - \mathbf{x}_i\|_1 + \\ & \frac{1}{N} \sum_{\mathbf{x}_i \in \mathbf{X}} \min_{\mathbf{y}_j \in \mathbf{Y}} \|\mathbf{x}_i - \mathbf{y}_j\|_1 \end{aligned} \tag{16}$$

where $\mathbf{X} = \{\mathbf{x_i}\}$ denotes the points sampled from the predicted model, $\mathbf{Y} = \{\mathbf{y_j}\}$ denotes the points sampled from the original model, and N and M is the number of points of the sets $\mathbf{X}$ and $\mathbf{Y}$, respectively. For D-FAUST dataset, it provides a dense point cloud for each human model, which we take as $\mathbf{Y}$. ShapeNet, on the other hand, does not provide point cloud representation for the object. So, we need to sample points densely on each face of the original mesh. To obtain $\mathbf{X}$, we apply the equal-distance sampling strategy [1] on each superquadric surface $\boldsymbol{\theta}_k \in \mathbf{\Theta}$ of the predicted model to get a point set $\mathbf{\Gamma}_k$. However, some points from $\mathbf{\Gamma}_k$ might lie inside of another superquadric $\boldsymbol{\theta}_l, l \neq k$, *i.e.*, those points are on the inside of the 3D model. Therefore, we need to remove those interior points by forming a subset $\tilde{\mathbf{\Gamma}}_k \subset \mathbf{\Gamma}_k$,

$$\tilde{\mathbf{\Gamma}}_k = \{\boldsymbol{\gamma}_i^k \mid \boldsymbol{\gamma}_i^k \in \mathbf{\Gamma}_k, f(\boldsymbol{\gamma}_i^k, \boldsymbol{\theta}_l) \geq 0, \forall \boldsymbol{\theta}_l \in \mathbf{\Theta}\} \tag{17}$$

where $f(.)$ denotes the inside-outside function of the superquadric. By taking the union of all the point sets $\{\tilde{\mathbf{\Gamma}}_1, \tilde{\mathbf{\Gamma}}_2, ..., \tilde{\mathbf{\Gamma}}_K\}$, we obtain a point cloud representation for the predicted model, which we treat as $\mathbf{X}$. For both ShapeNet and D-FAUST, we further downsample $\mathbf{X}$ and $\mathbf{Y}$ to 50K-60K points for calculating the Chamfer distance. The first term of Eq.16 computes how far on average the closest point of the predicted model is to the original mesh, and the second term calculates how far on average the closest point of the original mesh is to the predicted model. Thus, a lower value of Chamfer distance implies a better abstraction accuracy in terms of surface fitness.

Intersection over Union (IoU): The definition of IoU is

shown as follows:

$$\text{IoU} = \frac{V\left(S_{\text{pred}} \cap S_{\text{original}}\right)}{V\left(S_{\text{pred}} \cup S_{\text{original}}\right)}, \tag{18}$$

where S_{pred} is the predicted primitive-based model obtained by our algorithm, S_{original} is the original mesh model, and $V(.)$ computes the volume. It is difficult, if not impossible, to obtain the volume of the intersection or union of two models. Therefore, we approximate the volume with the Monte Carlo method. Firstly, we sample a set of points $\mathbf{\Phi}$ uniformly with a predefined density inside the bounding box of $S_{\text{pred}} \cup S_{\text{original}}$. We use 100^3 points for ShapeNet and 64^3 points for DFAUST. The number is far more than the previous papers, expecting a more accurate evaluation. Then, for each point $\mathbf{x} \in \mathbf{\Phi}$, we check if it is inside of the original mesh and the predicted model, respectively. We approximate $V\left(S_{\text{pred}} \cap S_{\text{original}}\right)$ to be the number of points that are on the inside of both S_{original} and S_{pred}, and approximate $V\left(S_{\text{pred}} \cup S_{\text{original}}\right)$ with the number of points that are on the inside of either S_{original} or S_{pred}. If two models match perfectly, the IoU will be 1 and if two models disjoint from each other, the IoU is 0.

6.2. Implementation Details

In this section, we elaborate on the parameters implemented in the experiment. All the experiments use the settings provided as follows, if not specified in the experiment section in the paper. The truncation threshold for the target and source SDF is 1.3 times the input grid interval. In the connectivity marching step, the common ratio of the geometric sequence α is $4/5$; The minimum size of the valid connected volume $N_c = 5$; The primitive initial scale ratio $\gamma = 0.1$; The terminating marching threshold is 0.01 times the negative truncation threshold. For the probabilistic model, the parameter of the Bernoulli prior distribution is set as $p_0 = 0.01$; The variance σ^2 is initialized as the truncation threshold. During the primitive update step, we set the activation distance a as 3.5 times the truncation threshold. The source code of our algorithm is implemented in MATLAB. The experiments are conducted on a computer running Intel Core i9-9900K CPU. The baseline method SQ [2] is trained and tested on an NVIDIA RTX3090 GPU.

All the methods consume different types of input. We use the official codes and configurations of [2, 3]. For SQs, occupancy grids of resolution 32^3 are generated from meshes by the provided code. We had to and tried to modify their network to consume occupancy grids of 128^3. Relatively incremental improvement is observed (*e.g.*, chair IoU $0.30 \rightarrow 0.34$). Therefore, we used the original network and followed the official configuration for consistency with the previous literature. We use 1000 and 200 points from objects and each superquadric for the loss function, respectively. For NB, we densely sample points from mesh and uniformly downsample to around 3500 (ShapeNet) and 5500 (DFAUST) points and set the number of initial components $K = 30$ following the settings in [3].

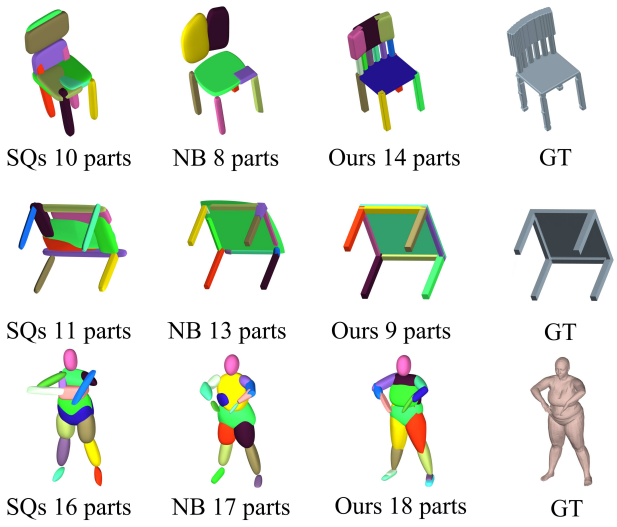

Figure 3. Shape abstraction results with the number of parts. Recovered primitives are colored in different colors.

6.3. Number of Parts

The number of parts used is not and cannot be predefined in all the methods. SQs [2] has a hyper-parameter to limit the maximum number set to 20. The training is unsupervised, and the network learns to predict the number of components. NB [3] infers the number via the Chinese Restaurant Process, splitting/merging when probabilistically necessary with no limits. Our method grows parts as needed. Since there is no ground truth and shapes vary greatly even in the same category, it is hard to quantify the correct number, which makes statistics less meaningful. Our result is satisfying qualitatively as shown in Fig.3. Our method can successfully separate different parts if they possess different geometric semantics (*e.g.* telling apart cuboids, cylinders, and balls). Therefore, it is semantically interpretable. In many cases (*e.g.*, Fig.3), the segmentation coincides with the human-defined semantics, though not trained to.

6.4. Time Performance

The proposed method (MPS) has an average runtime of 6.7s on ShapeNet and 2.5s on DFAUST per item. Time varies on the complexity of objects and grid resolutions. For an intuitive example, the chair, table, and human in Fig.3 take 3.9s, 3.6s, and 2.8s, respectively. The complex Reading Room takes 146s for resolution 400^3 and 30s for 200^3.

6.5. Additional Results

Due to the limited length of the paper, in this Supplementation Material, we prepare more qualitative comparisons

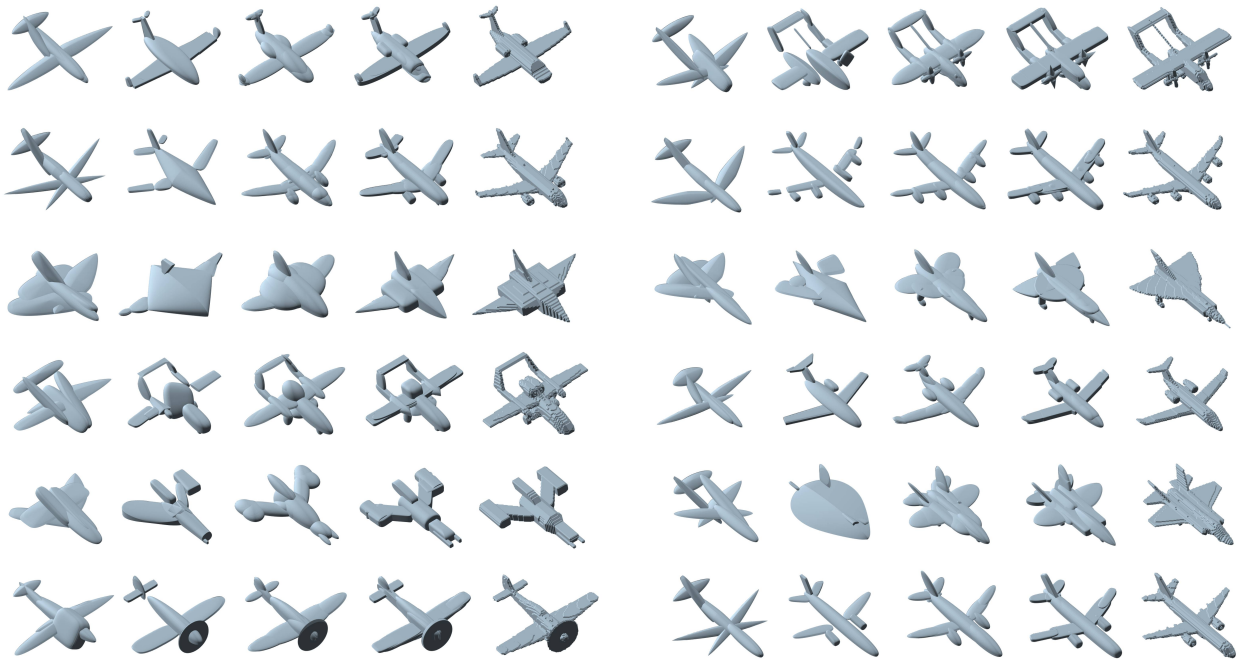

Figure 4. Shape Abstraction results on airplanes. From left to right: SQs, Non-parametric Bayesian (NB), Marching-Primitives with ellipsoids (MPE), Marching-Primitives with superquadrics (MPS), and the ground truth (pre-processed watertight mesh).

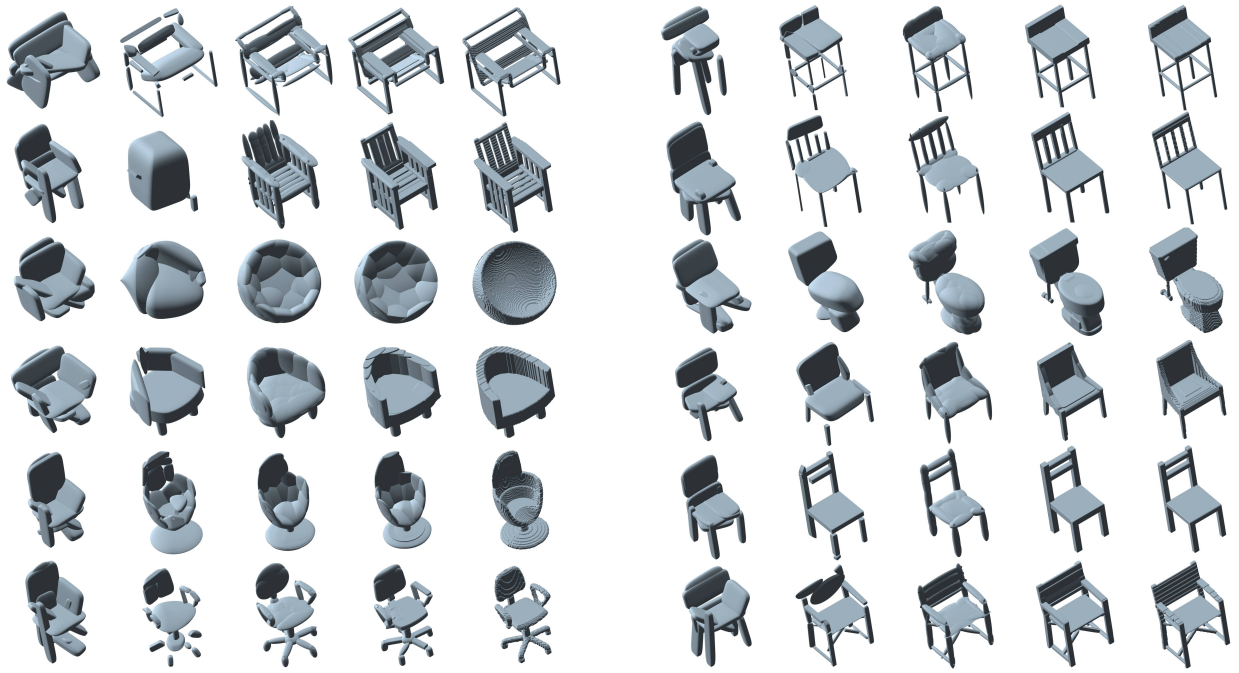

Figure 5. Shape Abstraction results on chairs. From left to right: SQs, Non-parametric Bayesian (NB), Marching-Primitives with ellipsoids (MPE), Marching-Primitives with superquadrics (MPS), and the ground truth (pre-processed watertight mesh).

on the ShapeNet dataset. From the additional results, we further demonstrate that our method is able to achieve high accuracy shape abstraction. Our method not only well captures the geometry of different objects in a same category, but also is generalizable among various categories without the need of fine-tuning.

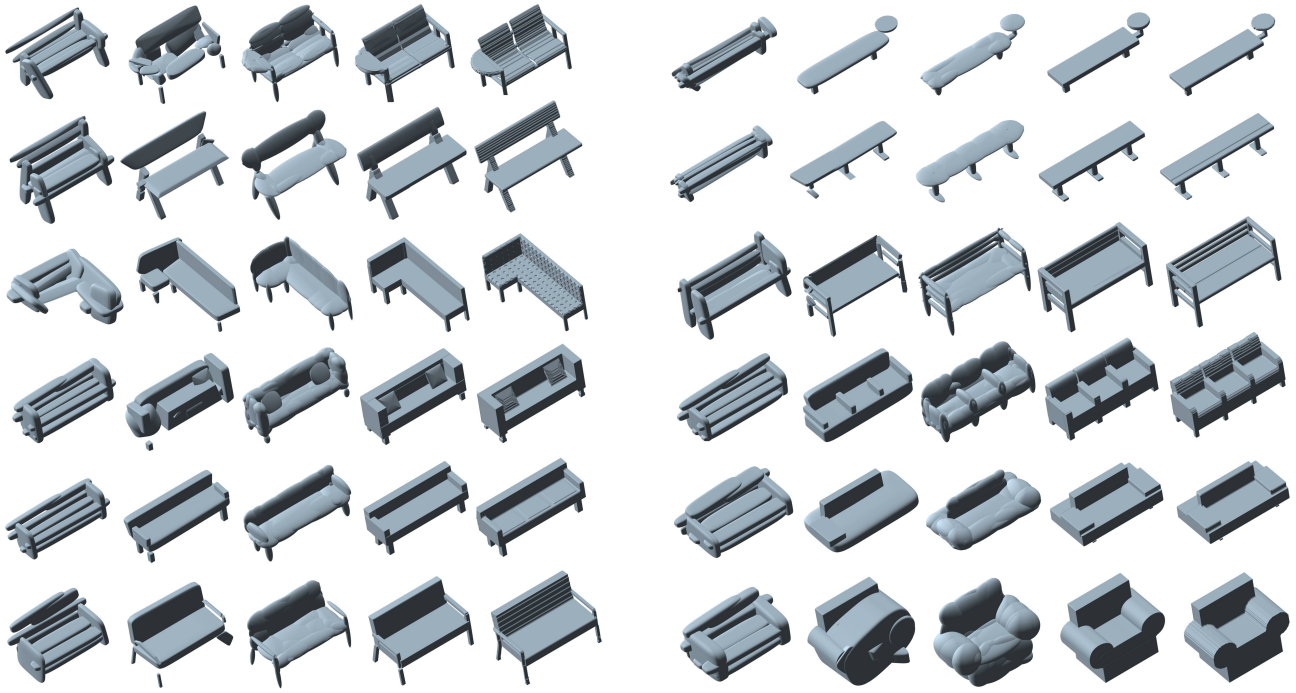

Figure 6. Shape Abstraction results on benches and sofas. From left to right: SQs, Non-parametric Bayesian (NB), Marching-Primitives with ellipsoids (MPE), Marching-Primitives with superquadrics (MPS), and the ground truth (pre-processed watertight mesh).

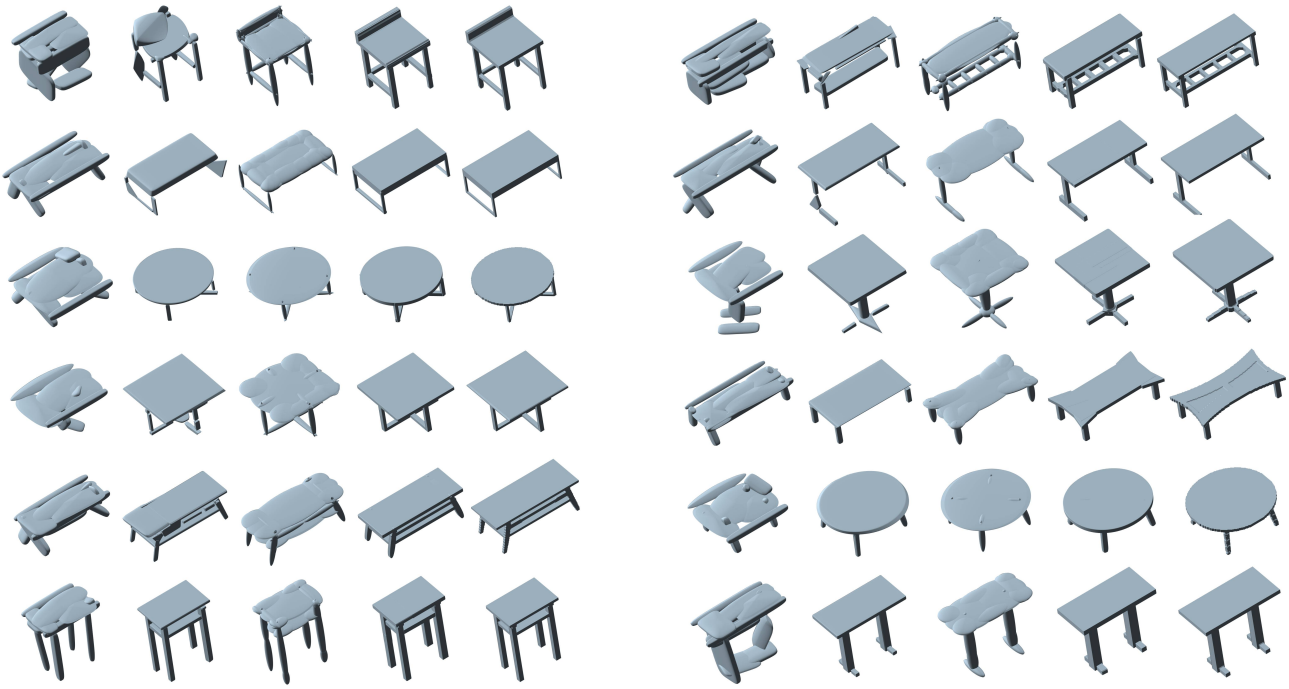

Figure 7. Shape Abstraction results on tables. From left to right: SQs, Non-parametric Bayesian (NB), Marching-Primitives with ellipsoids (MPE), Marching-Primitives with superquadrics (MPS), and the ground truth (pre-processed watertight mesh).

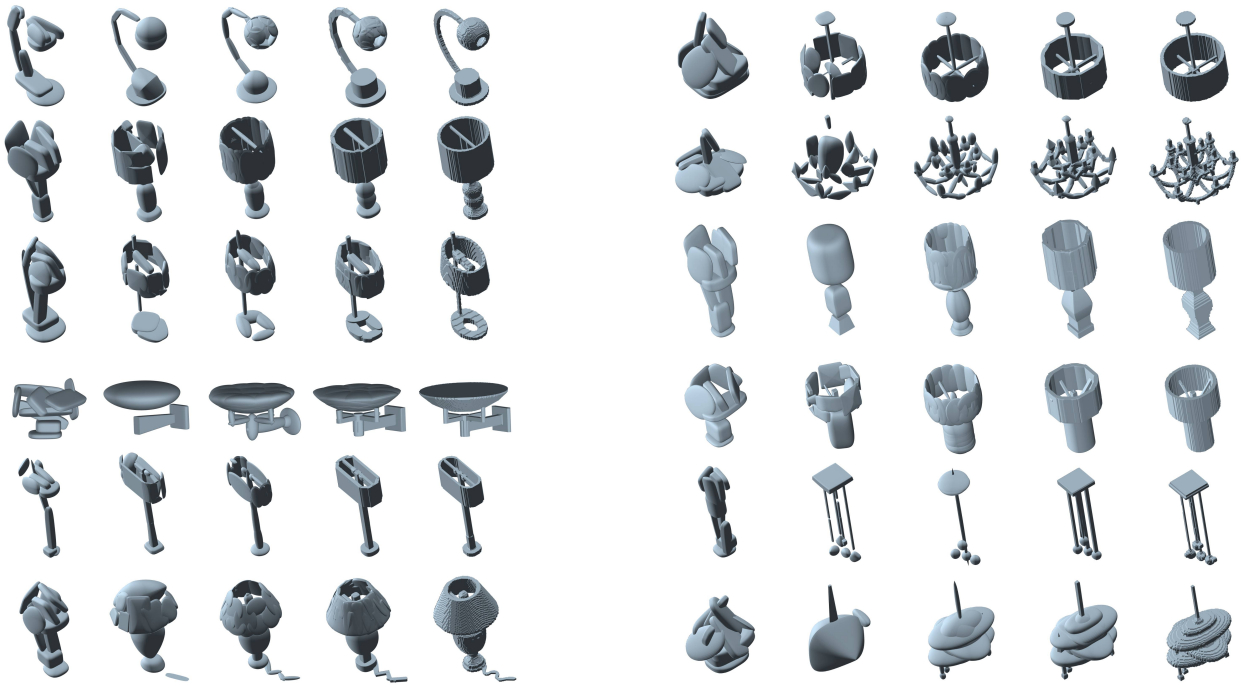

Figure 8. Shape Abstraction results on lamps. From left to right: SQs, Non-parametric Bayesian (NB), Marching-Primitives with ellipsoids (MPE), Marching-Primitives with superquadrics (MPS), and the ground truth (pre-processed watertight mesh).

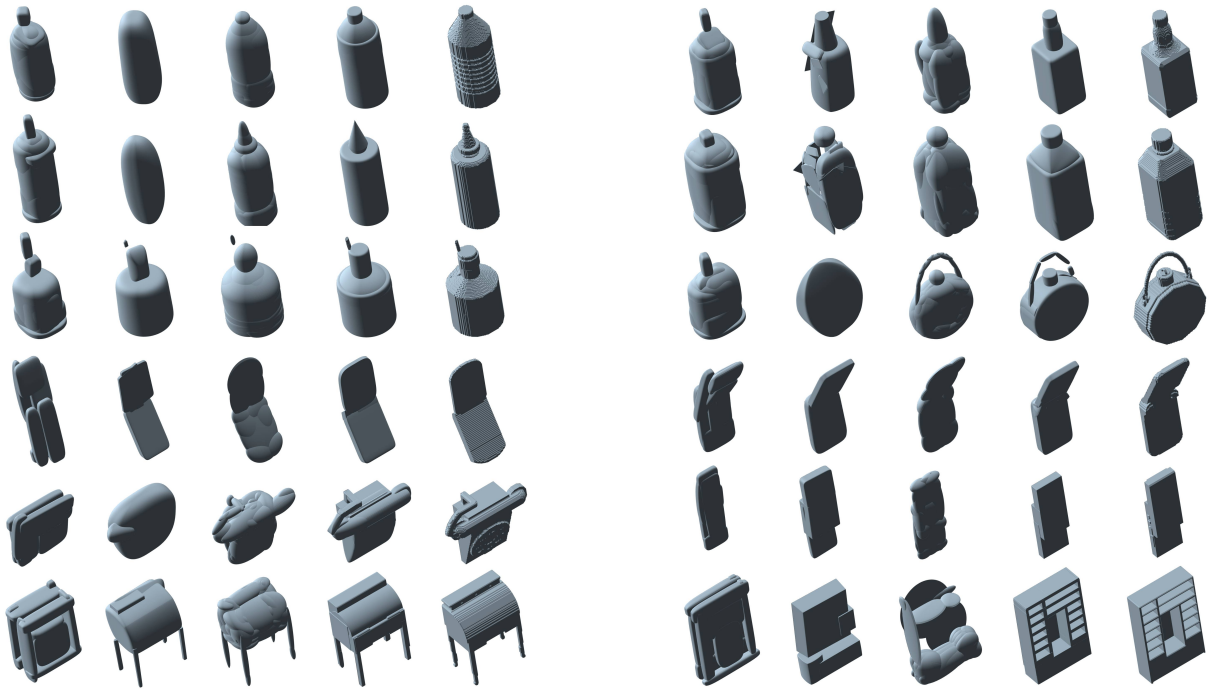

Figure 9. Shape Abstraction results on bottles, phones and cabinets. From left to right: SQs, Non-parametric Bayesian (NB), Marching-Primitives with ellipsoids (MPE), Marching-Primitives with superquadrics (MPS), and the ground truth.

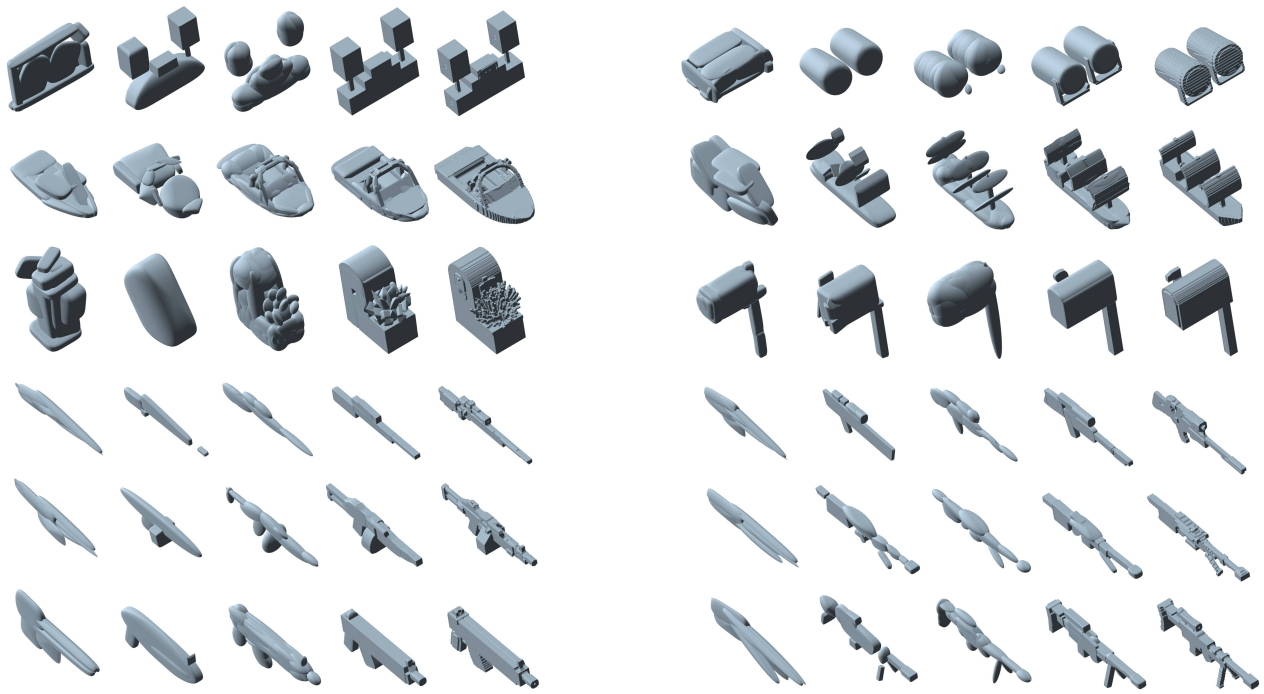

Figure 10. Shape Abstraction results on speakers, water-crafts, mailboxes and rifles. From left to right: SQs, Non-parametric Bayesian (NB), Marching-Primitives with ellipsoids (MPE), Marching-Primitives with superquadrics (MPS), and the ground truth.

References

[1] W. Liu, Y. Wu, S. Ruan, and G. S. Chirikjian. Robust and accurate superquadric recovery: A probabilistic approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2676–2685, June 2022. 3

[2] D. Paschalidou, A. O. Ulusoy, and A. Geiger. Superquadrics revisited: Learning 3D shape parsing beyond cuboids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 4

[3] Y. Wu, W. Liu, S. Ruan, and G. S. Chirikjian. Primitive-based shape abstraction via nonparametric bayesian inference. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 479–495. Springer Nature Switzerland, 2022. 4