Marching-Primitives: Shape Abstraction from Signed Distance Function

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

Abstract

Representing complex objects with basic geometric primitives has long been a topic in computer vision. Primitive-based representations have the merits of compactness and computational efficiency in higher-level tasks such as physics simulation, collision checking, and robotic manipulation. Unlike previous works which extract polygonal meshes from a signed distance function (SDF), in this paper, we present a novel method, named Marching-Primitives, to obtain a primitive-based abstraction directly from an SDF. Our method grows geometric primitives (such as superquadrics) iteratively by analyzing the connectivity of voxels while marching at different levels of signed distance. For each valid connected volume of interest, we march on the scope of voxels from which a primitive is able to be extracted in a probabilistic sense and simultaneously solve for the parameters of the primitive to capture the underlying local geometry. We evaluate the performance of our method on both synthetic and real-world datasets. The results show that the proposed method outperforms the state-of-the-art in terms of accuracy, and is directly generalizable among different categories and scales. The code is open-sourced at https://github.com/ChirikjianLab/Marching-Primitives.git.

1. Introduction

Recent years have witnessed great progress in the areas of 3D shape representation and environmental perception. Low-level representations such as surface meshes, point clouds, and occupancy grids are widely used as inputs to high-level computer vision algorithms and artificial intelligence tasks. They have the advantage of being able to represent and visualize objects with high accuracy and rich local geometric features. However, the low-level representations are ineffective in delivering a general and intuitive sense of structural geometry as well as part-level scene understanding. Studies [3, 20] show that human vision, unlike computer vision, tends to perceive and understand scenes as combinations of simple primitive shapes. Human beings perform well and robustly in complex tasks, providing a basic geometric description of the scene is available [32]. Therefore, researchers turn to exploring the possibility of interpreting complex objects and scenes with basic geometric primitives. Taking advantage of the primitive-based representation, many higher-level tasks, such as segmentation [14, 16, 21, 30], scene understanding [29, 31, 41, 47], grasping [33, 44, 45] and motion planning [35, 36], are able to be solved efficiently.

However, it still remains challenging to extract primitive-based abstractions from low-level representations. Starting from the 1990s, Solina et al. [1, 17, 39] aim to extract a single superquadric representation from a simple object by minimizing the least-square error between the primitive and the measured points. Later in [7, 22], their method is extended to represent more complex objects with multiple primitives. More recently, the authors of [24, 47] reformulate the task as a probabilistic inference problem with enhanced accuracy and robustness to noise and outliers. At the same time, with the surge of data-driven techniques, researchers attempt to train neural networks to infer cuboids [27, 38, 41, 48, 50] and superquadrics [29, 31] representations in an end-to-end fashion. However, both the computational and learning-based approaches have their own limitations. The computational methods are vulnerable to the inherent ambiguity of the point-to-surface relationship. For exam-
ple, the algorithms tend to fill empty spaces of a non-convex object with primitives by mistake, due to the inside/outside ambiguity of a surface depicted by a set of points [24, 48]. The main drawback of the learning approaches lies in the lack of generalizability beyond the object category on which the model is trained [24, 31, 47, 48]. Also, the shape abstraction accuracy is inferior to the computational methods.

The signed distance function (SDF) has been a successful 3D volumetric representation in varieties of computer vision and graphics tasks. It is the basic framework for many classic 3D reconstruction algorithms such as TSDF volume reconstruction [10, 15], KinectFusion [19], and DynamicFusion [26]. Recently, the SDF representation is adapted to the deep learning frameworks, and exhibits boosted potentials in shape encoding [8, 18, 28, 43], surface reconstruction [23, 46], and shape completion [11, 12, 34]. Usually, triangular mesh surfaces are extracted from the SDF representation with the marching cubes algorithm [25]. Point cloud and occupancy grid representations are also obtained by keeping the vertices of the meshes and the sign of each voxel point, respectively. The SDF is among the most informative 3D representations since it encodes not only the surface geometry but also the distance and side of a point relative to the shape. Meanwhile, it is easily achievable via range images from 3D sensors [10], or learnable from other input modalities [8, 18, 28]. Since we are able to extract meshes from an SDF, it is natural to think about the possibility of extracting primitives as well. Furthermore, the primitive-based abstraction is a continuous interpretation of the complete geometric information encoded in the original discrete SDF, but requires much less storage size (Fig.1).

Motivated by the aforementioned facts and the bottleneck of the current shape abstraction algorithms, we propose a general shape abstraction method by reasoning directly on the informative SDF representation. The goal of our method is to find a combination of geometric primitives whose underlying SDF values match the target values evaluated on the evenly spaced discrete grid points (Sec. 3.1 and Sec. 3.2). To solve this problem, we propose a two-step iterative algorithm called the Marching-Primitives. Our algorithm ‘marches’ on two domains: the signed distance domain and the voxelized space domain, alternately. Firstly, the connectivity of volumes are analyzed by generating isosurfaces on a sequence of decreasing levels of negative signed distances (Sec.3.3). By doing so, volumes of interest (VOIs) where primitives are likely to be encoded can be identified sequentially. In the second step, for each of the VOIs, our algorithm marches on the neighbouring voxels to infer their probabilistic correspondences to the primitive and simultaneously optimizes the shape and pose of the primitive (Sec.3.4). After the primitive representation of a VOI is achieved, the fitted volumes are deactivated from the voxel grid. Our algorithm continues marching on the signed distance domain until it approaches zero, i.e., all the interior volumes of the SDF have been captured by the recovered primitives. We compare our algorithm with the state-of-the-art of both the computational and learning-based approaches on the ShapeNet object dataset [6] and D-FAUST human shape dataset [4] (Sec. 4.1). We also study the performance of our algorithm on different conditions (Sec. 4.2). Finally, we demonstrate the scene abstraction result of the Stanford Reading Room [49], which contains several pieces of furniture of various categories (Sec. 4.3).

2. Related Work

SDF Representation: The SDF can be stored in two different ways: discrete or continuous. A majority of computer vision and graphics algorithms are built on the SDF discretized on a 3D grid of voxel points. The signed distances are stored on each of the corresponding voxel points. The authors of [10, 19, 26] pioneer in fusing several noisy range images into a single discrete SDF. Their work is widely applied in 3D reconstruction and plays an important role in robotics tasks such as simultaneous localization and mapping. The discrete SDF is also a promising input/output representation for 3D deep learning [11, 12, 34]. Recently, it becomes popular to encode shapes as a continuous SDF with neural networks [28]. Vasu et al. [43] further improve the shape encoding quality by enforcing local regularities with geometric primitives. In [8], the authors adopt a two-stage meta-learning approach to further extend the generalization capabilities of neural SDF. With the deep neural network, it becomes possible to infer SDF representations from partially observed 3D inputs or even images. Both the discrete and the continuous SDF are implicit representations of geometric surfaces. To extract the explicit surface from the SDF, continuous SDFs need to be discretized on a voxel grid first and then conduct the marching cubes [25]. This method allows high-quality rendering of the objects, however, surface meshes are non-sparse and contain no structural level information. Our method provides an alternative approach to describe the underlying object in the SDF. Instead of meshes, we directly extract a collection of sparsely parameterized primitives from the SDF. Other than that, our primitive-based representation itself is also a concise yet continuous SDF approximation to the original discrete SDF.

Computational Shape Abstraction: The most well-studied primitive for computational shape abstraction is the superquadric, due to its extensive shape vocabulary including cuboids, ellipsoids, cylinders, octohedra, and many shapes in between (e.g., cuboids with rounded edges). It is first proposed as a versatile modeling element for complex objects in computer graphics [2, 32]. Later, Solina et al. propose a method to conduct abstraction of simple objects from range images with a single superquadric [17, 39]. Leonardis et al. [22] and Chevalier et al. [7] further extend
the previous work to recover complex objects with multiple superquadrics with a Split-and-Merge strategy. A numerical instability problem is addressed and revisited in [42]. The authors introduce an auxiliary function in the unstable region and receive a better abstraction accuracy. More recently, Liu et al. [24] formulate the problem in a probabilistic fashion and propose a geometric strategy to avoid local optimum, bringing a significant improvement in robustness to outlier and fitting accuracy. Wu et al. [47] extend and recast the work as a nonparametric Bayesian inference problem so as to improve the applicability on complex shapes. To the best of the authors’ knowledge, the existing computational methods are all based on range images or point clouds, which suffer from geometric ambiguities [48]. In contrast, our method takes advantage of the abundant geometric information encoded in the SDF and is easily compatible with other computer vision algorithms based on the SDF representation.

**Learning-based Shape Abstraction:** The learning-based method is first seen in [41]. Tulsiani et al. propose a 3D convolutional neural network (CNN) to learn shape abstractions with cuboids from the occupancy grid. Sun et al. [40] design an adaptive hierarchical cuboid representation and introduce an unsupervised approach to learn to extract the parameters for the representation. Yang et al. [48] train a variational auto-encoder network to transform point clouds into parametric cuboids. Other than cuboids, researchers also seek to extract spheres and ellipsoids representations from objects. Hao et al. [18] combine the neural SDF with the spherical representation by sharing a same latent layer. In [37], ellipsoids or cuboids are extracted to help segment the input point cloud. However, a single type of primitive has very limited expressiveness. Therefore, Paschalidou et al. [29, 31] turn to training neural networks to conduct abstractions with the superquadrics as the atomic elements. Learning-based approaches are versatile in dealing with different input sources. They are able to make shape abstractions from point clouds, voxel grids, or even RGB images which are so ill-conditioned that the computational approaches have little chance of working. However, learning-based approaches rely heavily on the training dataset and thus are less generalizable to unseen categories. Instead, our approach reasons about the primitive abstraction from a case-by-case geometric perspective, which provides an inherent advantage in generalizability and accuracy.

3. Method

3.1. Preliminary

The discrete SDF is a volumetric surface representation built on a voxel grid \( V = \{ x_i \in \mathbb{R}^3, i = 1, 2, ..., N \} \). A scalar \( d(x_i) \) is assigned to each grid point, which indicates the signed distance of \( x_i \) to the nearest surface. The point \( x_i \) lies inside the surface if \( d(x_i) < 0 \) and outside otherwise. Typically, the surface mesh of an object is extracted from the SDF by the marching cubes algorithm [25]. In this paper, we call the input SDF \( d(x_i) \) the target SDF.

For the primitive representation, we select the superquadrics [2], a family of geometric primitives defined by the implicit equation

\[
f(x) = \left( \frac{x}{a_1} \right)^{\frac{2}{\epsilon_1}} + \left( \frac{y}{a_2} \right)^{\frac{2}{\epsilon_2}} + \left( \frac{z}{a_3} \right)^{\frac{2}{\epsilon_3}} = 1 \quad (1)
\]

The superquadric family is only encoded by 5 parameters (shape parameters \( \epsilon_1, \epsilon_2 \in [0, 2] \subset \mathbb{R} \), and scale parameters \( a_x, a_y, a_z \in \mathbb{R}_{>0} \), but has an extensive shape vocabulary. Note that the shape parameters can exceed 2, resulting in nonconvex shapes. However, practically in most studies [24, 31] and also in our paper, we limit them within the convex region as defined above. The points \( x = [x, y, z] \in \mathbb{R}^3 \) satisfying Eq. (1) form the surface of the superquadric. In this paper, we also include the Euclidean transformation \( g \in SE(3) \), i.e. 3 Euler angles for rotation \( R \in SO(3) \) and 3-dimensional translation \( t \in \mathbb{R}^3 \), to parameterize a superquadric with a general pose. In total, we denote a superquadric with a vector \( \Theta \) of 11 elements. According to Eq. (1), we can approximate the signed distance of a grid point \( x_i \) to a general posed superquadric \( \Theta \) explicitly by

\[
d_\Theta(x_i) = \left( 1 - f^{-\frac{2}{\epsilon_1}}(g^{-1} \circ x_i) \right) \| g^{-1} \circ x_i \|_2 \quad (2)
\]

We are not able to use the exact SDF of superquadrics, because it has no analytical solution and is only achievable by numerical optimization [5]. Eq.(2) is the radial distance [17, 24] of \( x_i \) to the superquadric surface, which converges to the exact signed distance as \( d_\Theta(x_i) \to 0 \). Therefore, we truncate both the input target SDF \( d(\cdot) \) and the primitive SDF \( d_\Theta(\cdot) \) within the vicinity of zero to ensure the approximation accuracy.

3.2. Problem Formulation

To obtain a primitive-based abstraction from the target SDF, we seek a combination of primitives \( \Theta \triangleq \{ \Theta_k, k = 1, 2, ..., K \} \) whose underlying signed distances measured on the voxel points (we call it the source SDF in contrast to the target SDF) match the target SDF, that is

\[
\Theta = \arg \min_{\Theta} \sum_{x_i \in V} \min_{\Theta_k \in \Theta} \| d_\Theta_k(x_i) - d(x_i) \|_2^2 \quad (3)
\]

Our problem formulation is simple and intuitive, however, it is intractable to solve directly. First, the number of primitives \( K \) needed is unknown. Also, the correspondences between the voxel points \( x_i \) and the primitives \( \Theta_k \) are unknown a priori. Therefore, it makes Eq.(3) a chicken-and-egg problem: on the one hand, if we know the set of voxel
points contributing to a same primitive, we are able to solve for the parameters of the primitive by optimization; on the other hand, if we know the configurations of the primitives, we are able to tell the belongings of each voxel. Other than those factors, the computation complexity itself is prohibitively high, because we need to evaluate hundreds of thousands or even millions of voxel points. This makes it infeasible to operate directly on the complete dense voxel grid. To tackle these difficulties, we propose an iterative algorithm, the Marching-Primitives, which simultaneously solves the correspondences and primitive parameters efficiently. It is worth noting that our method is generalizable beyond the superquadrics. Any volumetric primitives with easily accessible SDF representations can be adapted to our framework.

3.3. Iterative Connectivity Marching

Instead of setting a predefined number of primitives, our method starts from an empty set of primitives and grows as needed. This is realized by analyzing the connectivity (in this paper, we use the 26-connectivity) of the voxels at different levels of signed distance. Given the target SDF, our algorithm checks the connectivity of voxels whose signed distance is less than a sequence of thresholds

\[ T^c = \{t^c_1, t^c_2, \ldots\}, \quad t^c_k = \min_{x_i \in V} d(x_i), \quad t^c_{m+1} = \alpha t^c_m \]  

where \( \alpha \in (0, 1) \subset \mathbb{R} \) is the common ratio, resulting in a geometric sequence exponentially decaying to zero. Each threshold \( t^c_m \) defines a set of disjoint isosurfaces \( S_m \), encompassing the connected voxels whose signed distances are less than the threshold.

\[ S_m = \{S_k, k = 1, 2, \ldots, |S_m|\} \]  

| \( S_m \) | denotes the number of the disjoint isosurfaces. We construct a subset \( \bar{S}_m \) from \( S_m \)

\[ \bar{S}_m = \{S_k \in S_m, |S_k| \geq N_c\} \subseteq S_m \]  

where \( |S_k| \) is the number of the connected voxels within the isosurface \( S_k \). \( \bar{S}_m \) is the set of the isosurfaces encompassing no less than \( N_c \) connected voxels. The connectivity marching starts from the innermost threshold \( t^c_1 \), where the resulting \( S_1 \) might be empty. The marching continues on the sequence (increasing the threshold) until \( \bar{S}_m \) is non-empty. Each of the connected volumes in \( \bar{S}_m \) (we call it a VOI) is an ideal starting point for growing a primitive, because: (1) those volumes are among the most interior of the target SDF, corresponding to most prominent part of the geometry; (2) disconnected and weakly connected volumes can be separated apart; (3) the SDF of a primitive can also be interpreted as layers of isosurfaces, and thus sharing similar geometric structure; and furthermore (4) the size of the volumes is large enough to provide sufficient geometric information for the primitive recovery. We illustrate the idea of isosurfaces in Fig.2(a). For each VOI, we initialize a primitive as an ellipsoid, with scales proportional to the size of its smallest bounding-box. The details for the primitive initialization are demonstrated in the Supplementary Material. The procedure of how a primitive marches from the initial guess to capture the local geometry is detailed in the next Sec. 3.4. After obtaining the primitive representations for all the VOIs in \( \bar{S}_m \), we subtract the voxels

\[ \{x_i \in V, d(x_i) \leq 0 \land d\alpha(x_i) \leq 0\} \]  

from the set of voxel grid \( V \). In other words, we deactivate the voxels which are interior of the target SDF and well fitted by a recovered primitive, while the updated SDF preserves the exterior and unfitted interior volumes. The connectivity marching repeats with the updated target SDF, until the threshold marches higher than a preset limit close to zero, indicating that no prominent interior volume is left unrepresented.

3.4. Probabilistic Primitive Marching

Firstly, we explain what we mean by primitive marching. In the marching cubes algorithm [25], marching means the progressive extraction of polygonal meshes from the neighboring 8 voxel points in the discrete SDF. In each step, the algorithm focuses within the single cube formulated by the 8 voxel points, calculates triangle vertices using linear in-
parameter of the primitive is updated by correspondences between the primitive and voxels, the primitive is able to achieve an overall better fitting quality.

The target signed distance values. In this way, the algorithm to occupy some interior volumes without complying with the interior voxels, allowing part of the primitive to be shrunk to a proper shape. This is because, under this circumstance, only variations which shrink the volume of the primitive can decrease the difference between the truncated target and source SDF. Secondly, when the primitive is initialized near or inside the target volumes, the marching process encourages the primitive to converge to one nearest local target shape by analyzing the posterior correspondences. Fig. 2(c) illustrates the examples of the above properties. After the primitive marching, our algorithm removes the degenerated primitives. As a fail-safe measure, primitives which significantly contradict the target SDF is also removed. Detailed implementations can be found in the Supplementary Material.

\[ P_{ik} = p(z_{ik} = 1|\theta_{k}^{prev}, d_{i}) = \frac{1}{1 + \exp(-\theta_{ik})} \]

\[ V_{a} = \{ x_{i} \in V | d_{i}^{prev}(x_{i}) \in [-a, a] \subset \mathbb{R} \} \]

\[ \theta_{k}^{prev} \] is the previous estimation of the primitive parameters. \( V_{a} \) is an adaptive subset of the complete voxels V, which includes the voxels close to the surface of the previous estimated primitive. \( a \) defines the distance threshold of the activated region. Only voxels in \( V_{a} \) are activated during the optimization. The reason why we use an adaptive subset instead of the complete voxel space is twofold. As discussed in Sec. 3.1, both the source and the target SDF are truncated within a vicinity of zero. Therefore, small variations around \( \theta_{k}^{prev} \) (i.e., small changes on the shape of the primitive surface) have minor if not zero effects on the source SDF values evaluated at voxels distant to the primitive surface. Moreover, the size of \( V_{a} \) is much smaller than the complete set, and thus providing a significant boost in performance.

The correspondence marching and the primitive updating alternate until convergence. The process is akin to the EM algorithm [13]. The scope of voxels from which a primitive can be extracted is ‘marched’ with the progressive variation of the primitive shape, and simultaneously the parameters of the primitive are optimized based on the scope of voxels. We apply the optimization method proposed in [24] to avoid local minima. More derivation and implementation details can be found in the Supplementary Material.

3.5. Fail-safe Auto-degeneration

The primitives are roughly initialized in the connectivity marching step and further grow to capture local geometric shape. Since the probabilistic marching is based on optimization, an important question to ask is if the recovered primitives can always converge to a proper shape, regardless of poor initialization. We demonstrate that our probabilistic marching strategy is robust to initialization. Firstly, our method possesses a feature called auto-degeneration, i.e., the primitive will autonomously degenerate towards a point when accidentally initialized far from the target volumes. This is because, under this circumstance, only variations which shrink the volume of the primitive can decrease the difference between the truncated target and source SDF. Secondly, when the primitive is initialized near or inside the target volumes, the marching process encourages the primitive to converge to one nearest local target shape by analyzing the posterior correspondences. Fig. 2(c) illustrates the examples of the above properties. After the primitive marching, our algorithm removes the degenerated primitives. As a fail-safe measure, primitives which significantly contradict the target SDF is also removed. Detailed implementations can be found in the Supplementary Material.
4. Experiments

In this section, we conduct several experiments to demonstrate the high accuracy and generalizability of our proposed method. First, we show the shape abstraction results on various datasets, ranging from daily objects to human body models. Furthermore, we study the impact of the grid resolution and the truncation threshold on our algorithm. In the end, we apply our algorithm on a large-scale complex indoor scene, which contains a mixed combination of furniture such as sofas, tables, books, lamps, etc. Implementation details can be found in the Supplementary.

4.1. Evaluation on Datasets

Baselines: We compare our method with both the state-of-the-art learning-based method [31] and the computational method [47], which infer superquadric abstractions of the input objects. For convenience, we refer to [31] as SQs, and [47] as NB. We use the official codes and follow the implementation details as stated in these papers, respectively. We do not compare with the cuboid-based algorithms [40, 48, 50], since they focus more on the semantic level abstraction and are limited in expressiveness and accuracy. A detailed performance comparison between the cuboid and superquadric abstractions can be found in [31].

Datasets: We evaluate on two commonly used datasets, the ShapeNet [6] and the DFAUST [4]. In both of the datasets, we are only provided with triangular meshes. Therefore, we need to transform the meshes into discrete SDF representations. It is straightforward to calculate the signed distance value of a point to a watertight mesh. However, the ShapeNet is a human-made synthetic dataset containing many non-watertight meshes formed by non-volumetric 2D surfaces, self-intersecting or overlapped triangles. Therefore, we pre-process the original ShapeNet models into watertight meshes and then generate the SDF representation with the fast marching algorithm, following the same procedure in [8]. In consideration of fairness and consistency, we use the pre-processed watertight meshes as the common ground truth and the inputs to SQs and NB. For the DFAUST dataset, we generate the SDF directly since the provided meshes are already watertight.

Metrics: Following [31, 47] we use the Chamfer-L1 distance and the volumetric intersection over union (IoU) as the quantitative evaluation metrics. The computation of the metrics is detailed in the Supplementary Material.

Results on ShapeNet: We experiment on 14 categories from the ShapeNet dataset. We split the dataset randomly into the training (80%) and testing (20%) sets [9], where we train one SQs model per category on the training sets and evaluate all the methods on the testing sets. For our method, we use the SDF representation discretized on a voxel grid of size $100^3$ and range $[-0.5, 0.5]^3$. Other than superquadrics, we also test our method with ellipsoids as the base primitive, which is a special case of the superquadric representation when the shape parameters $\epsilon_1, \epsilon_2$ are fixed to 1. The quantitative results are summarized in Table. 1. We also demonstrate the qualitative comparison among different shape abstraction methods in Fig. 3. Our method outperforms all the baselines, even implemented with less expressive ellipsoids. The computational method generally performs better than the learning-based method. This is because the parameters of the primitive are so sparse and geometrically interrelated that they are difficult to get mapped from a high dimensional input by a neural network. Compared with NB, our method has richer details, clearer edges, and more importantly does not occupy the exterior space. Two factors contribute to the advantages. The first one is the extra geometric information embedded in the SDF representation, which eliminates the inherent interior/exterior ambiguity of point clouds. The other one is our special de-
Table 1. Quantitative results on Shapenet. MPE and MPS are short for Marching-Primitives with ellipsoids and superquadrics, respectively.

<table>
<thead>
<tr>
<th>Category</th>
<th>SQs [31]</th>
<th>NB [47]</th>
<th>MPE (Ours)</th>
<th>MPS (Ours)</th>
<th>SQs [31]</th>
<th>NB [47]</th>
<th>MPE (Ours)</th>
<th>MPS (Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>airplane</td>
<td>0.037</td>
<td>0.023</td>
<td>0.021</td>
<td>0.019</td>
<td>0.441</td>
<td>0.671</td>
<td>0.731</td>
<td>0.768</td>
</tr>
<tr>
<td>bench</td>
<td>0.056</td>
<td>0.028</td>
<td>0.020</td>
<td>0.020</td>
<td>0.238</td>
<td>0.579</td>
<td>0.730</td>
<td>0.819</td>
</tr>
<tr>
<td>bottle</td>
<td>0.047</td>
<td>0.033</td>
<td>0.026</td>
<td>0.017</td>
<td>0.686</td>
<td>0.665</td>
<td>0.886</td>
<td>0.924</td>
</tr>
<tr>
<td>cabinet</td>
<td>0.059</td>
<td>0.036</td>
<td>0.037</td>
<td>0.028</td>
<td>0.394</td>
<td>0.666</td>
<td>0.840</td>
<td>0.948</td>
</tr>
<tr>
<td>can</td>
<td>0.066</td>
<td>0.036</td>
<td>0.036</td>
<td>0.022</td>
<td>0.706</td>
<td>0.553</td>
<td>0.908</td>
<td>0.950</td>
</tr>
<tr>
<td>chair</td>
<td>0.068</td>
<td>0.027</td>
<td>0.023</td>
<td>0.020</td>
<td>0.300</td>
<td>0.685</td>
<td>0.785</td>
<td>0.871</td>
</tr>
<tr>
<td>lamp</td>
<td>0.072</td>
<td>0.029</td>
<td>0.022</td>
<td>0.021</td>
<td>0.234</td>
<td>0.589</td>
<td>0.750</td>
<td>0.802</td>
</tr>
<tr>
<td>speaker</td>
<td>0.064</td>
<td>0.041</td>
<td>0.037</td>
<td>0.033</td>
<td>0.346</td>
<td>0.656</td>
<td>0.858</td>
<td>0.920</td>
</tr>
<tr>
<td>mailbox</td>
<td>0.095</td>
<td>0.026</td>
<td>0.026</td>
<td>0.024</td>
<td>0.333</td>
<td>0.694</td>
<td>0.802</td>
<td>0.905</td>
</tr>
<tr>
<td>rifle</td>
<td>0.038</td>
<td>0.020</td>
<td>0.019</td>
<td>0.019</td>
<td>0.446</td>
<td>0.732</td>
<td>0.744</td>
<td>0.811</td>
</tr>
<tr>
<td>sofa</td>
<td>0.054</td>
<td>0.037</td>
<td>0.029</td>
<td>0.023</td>
<td>0.497</td>
<td>0.726</td>
<td>0.857</td>
<td>0.940</td>
</tr>
<tr>
<td>table</td>
<td>0.070</td>
<td>0.024</td>
<td>0.024</td>
<td>0.022</td>
<td>0.247</td>
<td>0.745</td>
<td>0.818</td>
<td>0.932</td>
</tr>
<tr>
<td>phone</td>
<td>0.040</td>
<td>0.021</td>
<td>0.023</td>
<td>0.021</td>
<td>0.681</td>
<td>0.872</td>
<td>0.891</td>
<td>0.947</td>
</tr>
<tr>
<td>watercraft</td>
<td>0.048</td>
<td>0.032</td>
<td>0.022</td>
<td>0.022</td>
<td>0.465</td>
<td>0.618</td>
<td>0.793</td>
<td>0.836</td>
</tr>
<tr>
<td>mean</td>
<td>0.057</td>
<td>0.028</td>
<td>0.024</td>
<td>0.022</td>
<td>0.368</td>
<td>0.674</td>
<td>0.793</td>
<td>0.870</td>
</tr>
</tbody>
</table>

Results on DFAUST: We also evaluate on the DFAUST dataset of human body models. We follow the split settings in [29] to train the learning-based method and evaluate all the methods on the testing set. The SDF is discretized on a grid of size $64^3$ and side length $1$. Similar to the ShapeNet experiment, we test our method with both superquadrics and ellipsoids. The results are shown in Fig. 4. Our method can accurately capture various postures, while the baselines fail to distinguish different body parts in some cases. The abstraction quality of SQs is much better on this dataset compared with the ShapeNet, since human bodies share a common articulated structure that can be captured by the neural network. We observe that the ellipsoid abstraction also achieves satisfying accuracy. This is because the human body mostly consists of rounded shapes compared with man-made objects in the ShapeNet.

4.2. Performance Study

This section investigates how the Marching-Primitives algorithm behaves with different voxel grid sizes and distance truncation thresholds. We experiment on the Armadillo from the Stanford 3D scanning repository [10]. First, we discretize the model on the voxel grids of different resolutions. The qualitative abstraction results are visualized in Fig. 5, and the quantitative results are summarized in Fig. 6(a). When the input grid resolution is low, the recovered primitive representation is relatively abstract. With the increase of the grid resolution, our method is able to extract detailed abstraction more faithful to the target shape. This is because the higher the resolution, the more geometric information our method can utilize to guide the primitive marching process. As discussed in Sec. 3.1, we use a truncated version of the target and source SDF. Therefore, we also evaluate the performance on different truncation thresholds, starting from 0.1 to 4 times the interval of the voxel grids. The results are shown in Fig. 6(b). The abstraction accuracy increases as the threshold decreases at first. The reason lies in that the approximated superquadric
(source) SDF converges to the true value when truncated close to the surface. If we further decrease the threshold, however, we observe an acute decrease in accuracy. This is because an overly small truncation threshold corrupts the original geometric information embedded in the target SDF, as well as the smoothness of the cost function (Eq. (10)).

4.3. Scene Abstraction

Other than single objects, we also test the capability of our method in representing complex scenes with geometric primitives. We experiment on a real-world indoor scene called Reading Room [49]. The scene is captured by an Asus Xtion Pro Live camera. The RGB-D scans are fused utilizing [19] (an SDF-based 3D reconstruction algorithm), and further fine-tuned with the method proposed in [49]. However, the SDF representation of the scene is not available publicly. Therefore, we pre-process the scene mesh by removing the floor plane, filling the holes, and transforming it into the SDF representation with grid size $400^3$. The scene abstraction task is much more difficult compared with the object-level task because various items with great differences in size and shape are present in the very same space. We compare our abstraction result with the mesh extracted by the marching cubes algorithm on the same discrete SDF representation, as shown in Fig.7. Our representation is not only visually satisfying but also contains abundant geometric information the mesh lacks, since it is a continuous SDF approximation to the input discrete SDF. Moreover, our highly sparse representation is only 8.2KB in size, while the discrete SDF occupies 203MB.

5. Conclusion

We propose the Marching-Primitives, the first algorithm to extract primitive-based abstractions from the volumetric SDF representation. Our method outperforms the state-of-the-art in terms of accuracy and generalizability on both synthetic and real-world datasets. Our primitive-based representation is sparse, accurate, generalizable, and can be expressed analytically without training, which we believe will facilitate and inspire further explorations in scene understanding, 3D reconstruction, and robot motion planning. However, our algorithm has the limitation of not being able to properly extract object parts thinner than the interval of the voxel grid. This problem can be solved by either increasing the grid resolution or detecting and thickening the invalid parts. Future work includes the parallelization of the marching process and inferring semantic-level interpretations from the current primitive-based geometric features.

Acknowledgements This research is supported by NUS Startup grants A-0009059-02-00 and A-0009059-03-00, National Research Foundation, Singapore, under its Medium Sized Centre Programme - Centre for Advanced Robotics Technology Innovation (CARTIN), subaward A-0009428-08-00, and AME Programmatic Fund Project MARIO A-0008449-01-00, and JHU internal funds.
References


[27] C. Niu, J. Li, and K. Xu. Im2Struct: Recovering 3D shape structure from a single RGB image. In Proceedings of the