

# Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids

Despoina Paschalidou<sup>1,4</sup> Ali Osman Ulusoy<sup>2</sup> Andreas Geiger<sup>1,3,4</sup>

<sup>1</sup> Autonomous Vision Group, MPI for Intelligent Systems Tübingen

<sup>2</sup>Microsoft <sup>3</sup>University of Tübingen <sup>4</sup>Max Planck ETH Center for Learning Systems

{firstname.lastname}@tue.mpg.de

## Abstract

Abstracting complex 3D shapes with parsimonious part-based representations has been a long standing goal in computer vision. This paper presents a learning-based solution to this problem which goes beyond the traditional 3D cuboid representation by exploiting superquadrics as atomic elements. We demonstrate that superquadrics lead to more expressive 3D scene parses while being easier to learn than 3D cuboid representations. Moreover, we provide an analytical solution to the Chamfer loss which avoids the need for computational expensive reinforcement learning or iterative prediction. Our model learns to parse 3D objects into consistent superquadric representations without supervision. Results on various ShapeNet categories as well as the SURREAL human body dataset demonstrate the flexibility of our model in capturing fine details and complex poses that could not have been modelled using cuboids.

## 1. Introduction

Evolution has developed a remarkable visual system that allows humans to robustly perceive their 3D environment. It has long been hypothesized [2] that the human visual system processes the vast amount of raw visual input into compact parsimonious representations, where complex objects are decomposed into a small number of shape primitives that can each be represented using low-dimensional descriptions. Indeed, experiments show that humans can understand complex scenes from renderings of simple shape primitives such as cuboids or geons [3].

Likewise, machines would tremendously benefit from being able to parse 3D data into compact low-dimensional representations. Such representations would provide useful cues for recognition, detection, shape manipulation and physical reasoning such as path planning and grasping. In the early days of computer vision, researchers explored shape primitives such as 3D polyhedral shapes [8], generalized cylinders [4], geons [2] and superquadrics [25]. However, it proved very difficult to extract such representations

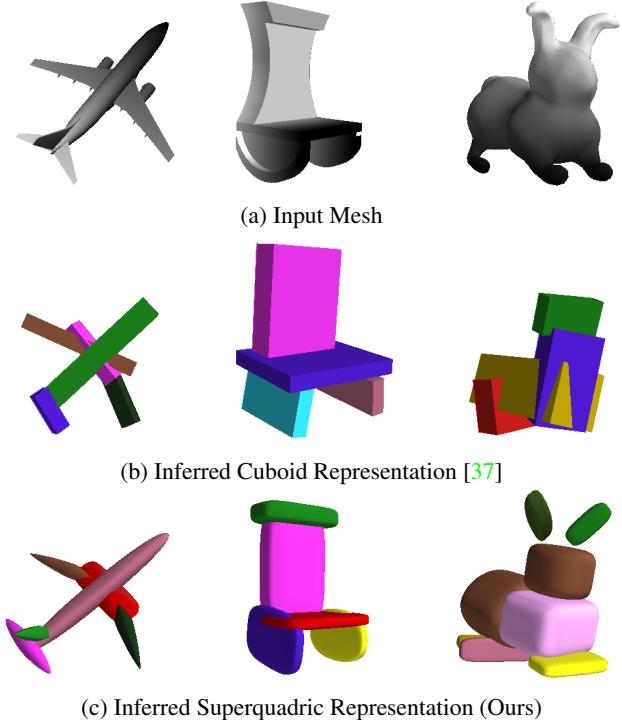


Figure 1: **3D Shape Parsing.** We consider the problem of learning to parse unstructured 3D data (e.g., meshes or point clouds) into compact part-based representations. Prior work [23, 37, 44] has considered cuboid representations (b) which capture the overall object structure, but lack expressiveness. In this work, we propose an unsupervised model for superquadrics (c), which allows us to capture details such as the body of the airplane and the ears of the rabbit.

from images due to the lack of computation power and data at the time. Thus, the research community shifted their focus away from the shape primitive paradigm.

In the last decade, major breakthroughs in shape extraction were due to deep neural networks coupled with the abundance of visual data. Recent works focus on learning 3D reconstruction using 2.5D [14, 16, 24, 43], volumetric [7, 11, 13, 18, 30, 42], mesh [12, 21] and point cloud [10, 27]

representations. However, none of the above are sufficiently parsimonious or interpretable to allow for higher-level 3D scene understanding as required by intelligent systems.

Very recently, shape primitives have been revisited in the context of deep learning. In particular, [23, 37, 44] have demonstrated that deep neural networks enable to reliably extract 3D cuboids from meshes and even RGB images.

Inspired by these works, we propose a novel deep neural network to efficiently extract parsimonious 3D representations in an unsupervised fashion, conditioned on a 3D shape or 2D image as input. In particular, this paper makes the following **contributions**:

First, we note that 3D cuboid representations used in prior works [23, 37, 44] are not sufficiently expressive to model many natural and man-made shapes as illustrated in Fig. 1. Thus, cuboid-based representation may require a large number of primitives to accurately represent common shapes. Instead, in this paper, we propose to utilize superquadrics, which have been successfully used in computer graphics [1] and classical computer vision [25, 34, 36]. Superquadrics are able to represent a diverse class of shapes such as cylinders, spheres, cuboids, ellipsoids *in a single continuous parameter space* (see Fig. 1+2). Moreover, their continuous parametrization is particularly amenable to deep learning, as their shape is smooth and varies continuously with their parameters. This allows for faster optimization, and hence faster and more stable training as evidenced by our experiments.

Second, we provide an analytical closed-form solution to the Chamfer distance function which can be evaluated in linear time wrt. the number of primitives. This allows us to compute gradients wrt. the model parameters using standard error backpropagation [31] without resorting to computational expensive reinforcement learning techniques as required by prior work [37]. We consequently mitigate the need for designing an auxiliary reward function. Instead, we formulate a simple parsimony loss to favor configurations with a small number of primitives. We demonstrate the strengths of our model by learning to parse 3D shapes from the ShapeNet [5] and the SURREAL [38]. We observe that our model converges faster than [37] and leads to more accurate reconstructions. Our code is publicly available<sup>1</sup>.

## 2. Related Work

In this section, we discuss the most relevant work on deep learning-based 3D shape modeling approaches and review the origins of superquadric representations.

### 2.1. 3D Reconstruction

The simplest representation for 3D reconstruction from one or more images are 2.5D depth maps as they can be

inferred using standard 2D convolutional neural networks [14, 18, 24, 43]. Since depth maps are view-based, these methods require additional post-processing algorithms to fuse information from multiple viewpoints in order to capture the entire object geometry. As opposed to depth maps, volumetric representations [7, 11, 13, 30, 35] naturally capture the entire 3D shape. While, hierarchical 3D data structures such as octrees accelerate 3D convolutions, the high memory requirements remain a limitation of existing volumetric methods. An alternative line of work [10, 28] focuses on learning to reconstruct 3D point sets. A natural limitation of these approaches is the lack of surface connectivity in the representation. To address this limitation, [12, 21, 29, 40] proposed to directly learn 3D meshes.

While some of the aforementioned models are able to capture fine surface details, none of them lends itself to parsimonious, semantic interpretations. In this work, we utilize superquadrics which provide a concise and yet accurate representation with significantly less parameters.

### 2.2. Constructive Solid Geometry

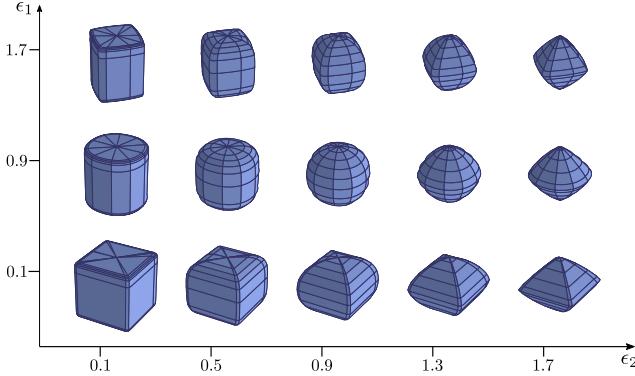
Towards the goal of concise representations, researchers exploited constructive solid geometry (CSG) [20] for shape modeling [9, 32]. Sharma et al. [32] leverage an encoder-decoder architecture to generate a sequence of simple boolean operations to act on a set of primitives that can be either squares, circles or triangles. In a similar line of work, Ellis et al. [9] learn a programmatic representation of a hand-written drawing, by first extracting simple primitives, such as lines, circles and rectangles and a set of drawing commands that is used to synthesize a L<sup>A</sup>T<sub>E</sub>X program. In contrast to [9, 32], our goal is not to obtain accurate 3D geometry by iteratively applying boolean operations on shapes. Instead, we aim to decompose the depicted object into a parsimonious interpretable representation where each part has a semantic meaning associated with it. Besides, we do not suffer from ambiguities of an iterative construction process, where different executions lead to the same result.

### 2.3. Learning-based Scene Parsing

Recently, shape primitives have been revisited in the context of deep learning [23, 37, 44]. Niu et al. [23] propose to use a recurrent neural network (RNN) to iteratively predict cuboid primitives as well as symmetry relationships from RGB images. They first train an encoder which encodes the input image and its segmentation into a 80-dimensional latent code. Starting from this root feature, they iteratively decode the structure into cuboids, splitting nodes based on adjacency and symmetry relationships. In related work, Zou et al. [44] utilize LSTMs in combination with mixture density networks to generate cuboid representations from depth maps encoded by a 32-dimensional feature vector. However, both works [23, 44] require supervision in terms of the

---

<sup>1</sup>[https://github.com/paschalidoud/superquadric\\_parsing](https://github.com/paschalidoud/superquadric_parsing)



**Figure 2: Superquadrics Shape Vocabulary.** Due to their ability to model various shapes with little parameters, superquadrics are a natural choice for geometric primitives.

primitive parameters as well as the sequence of predictions. This supervision must either be provided by manual annotation or using greedy heuristics as in [23, 44].

In contrast, our approach is unsupervised and does not suffer from ambiguities caused by different possible prediction sequences that lead to the same cuboid assembly. Furthermore, [23, 44] exploit simple cuboid representations which do not capture more complex shapes that are common in natural and man-made scenes (e.g., curved objects, spheres). In this work, we propose to use superquadrics [1] which yield a more diverse shape vocabulary and hence lead to more expressive scene abstractions as illustrated in Fig. 1.

A primary inspiration for this paper is the seminal work by Tulsiani et al. [37], who proposed a method for 3D shape abstraction using a non-iterative approach which does not require supervision. Instead, they use a convolutional network architecture for predicting the shape and pose parameters of 3D cuboids as well as their probability of existence. They demonstrate that learning shape abstraction from data allows for obtaining consistent parses across different instances in an unsupervised fashion.

In this paper, we extend the model of Tulsiani et al. [37] in the following directions. First, we utilize superquadrics, instead of cuboids, which leads to more accurate scene abstractions. Second, we demonstrate that the bi-directional Chamfer distance is tractable and doesn't require reinforcement learning [41] or specification of rewards [37]. In particular, we show that there exists an analytical closed-form solution which can be evaluated in linear time. This allows us to compute gradients wrt. the model parameters using standard error propagation [31] which facilitates learning. In addition, we add a new simple parsimony loss to favor configurations with a small number of primitives.

## 2.4. Superquadrics

Superquadrics are a parametric family of surfaces that can be used to describe cubes, cylinders, spheres, octahedra,

ellipsoids etc. [1]. In contrast to geons [2], superquadric surfaces can be described using a fairly simple parameterization. In contrast to generalized cylinders [2], superquadrics are able to represent a larger variety of shapes. See Fig. 2 for an illustration of the shape space.

In 1986, Pentland introduced superquadrics to the computer vision community [25]. Solina et al. [34] formulated the task of fitting superquadrics to a point cloud as a least-squares minimization problem. Chevalier et al. [6] followed a two-stage approach, where the point cloud is first partitioned into regions and then each region is fit with a superquadric. As a thorough survey on superquadrics is beyond the scope of this paper, we refer to [17, 33] for details.

In contrast to these classical works on superquadric fitting using non-linear least squares, we present the first approach to train a deep network to predict superquadrics directly from 2D or 3D inputs. This allows our network to distill statistical dependencies wrt. the arrangement and geometry of the primitives from data, leading to semantically meaningful parts at inference time. Towards this goal, we utilize a convolutional network that predicts superquadric poses and attributes, and develop a novel loss function that allow us to train this network efficiently from data. Our model is able to directly learn superquadric surfaces from an unordered 3D point cloud without any supervision on the primitive parameters nor a 3D segmentation as input.

## 3. Method

We now describe our model. We start by introducing the model parameters, followed by the loss functions and the superquadric parametrization we employ.

Given an input  $\mathbf{I}$  (e.g., image, volume, point cloud) and an oriented point cloud  $\mathbf{X}$  of the target object, our goal is to estimate the parameters  $\theta$  of a neural network  $\phi_\theta(\mathbf{I})$  that predicts a set of  $M$  primitives that best describe the target object. Every primitive is fully described by a set of parameters  $\lambda_m$  that define its shape, size and its position and orientation in the 3D space. For details about the parameterization of the superquadric representation, we refer the reader to Section 3.3.

Since not all objects and scenes require the same number of primitives, we enable our model to predict a variable number of primitives, hence allowing it to decide whether a primitive should be part of the assembled object or not. To achieve this, we follow [37] and associate every primitive with a binary random variable  $z_m \in \{0, 1\}$  which follows a Bernoulli distribution  $p(z_m) = \gamma_m^{z_m}(1 - \gamma_m)^{1-z_m}$  with parameter  $\gamma_m$ . The random variable  $z_m$  indicates whether the  $m^{\text{th}}$  primitive is part of the scene ( $z_m = 1$ ) or not ( $z_m = 0$ ). We refer to these variables as *existence variables* and denote the set of all existence variables as

$\mathbf{z} = \{z_1, \dots, z_M\}$ . Our goal is to learn a neural network

$$\phi_\theta : \mathbf{I} \mapsto \mathbf{P} \quad (1)$$

which maps an input  $\mathbf{I}$  to a primitive representation  $\mathbf{P}$  where  $\mathbf{P} = \{(\lambda_m, \gamma_m)\}_{m=1}^M$  comprises the primitive parameters  $\lambda_m$  and the existence probability  $\gamma_m$  for  $M$  primitives. Note that  $M$  is only an upper bound on the number of predicted primitives. The final primitive representation is obtained by sampling the existence of each primitive,  $z_m \sim \text{Bernoulli}(\gamma_m)$ .

One of the key challenges when training such models is related to the lack of direct supervision in the form of primitive annotations. However, despite the absence of supervision, one can still measure the discrepancy between the predicted object and the target object. Towards this goal, we formulate a bi-directional reconstruction objective  $\mathcal{L}_D(\mathbf{P}, \mathbf{X})$  and incorporate a Minimum Description Length (MDL) prior  $\mathcal{L}_\gamma(\mathbf{P})$ , which favors parsimony, i.e. a small number of primitives. Our overall loss function is given as:

$$\mathcal{L}(\mathbf{P}, \mathbf{X}) = \mathcal{L}_D(\mathbf{P}, \mathbf{X}) + \mathcal{L}_\gamma(\mathbf{P}) \quad (2)$$

We now describe both losses functions in detail.

### 3.1. Reconstruction Loss

The reconstruction loss measures the discrepancy between the predicted shape and the target shape. While we experimented with the truncated bi-directional loss of Tulsiani et al. [37], we empirically found that the standard Chamfer distance [10] works better in practice and results in less local minima. An empirical analysis on this is provided in our supplementary material. Thus, we use the Chamfer distance in our experiments

$$\mathcal{L}_D(\mathbf{P}, \mathbf{X}) = \mathcal{L}_{P \rightarrow X}(\mathbf{P}, \mathbf{X}) + \mathcal{L}_{X \rightarrow P}(\mathbf{X}, \mathbf{P}) \quad (3)$$

where  $\mathcal{L}_{P \rightarrow X}$  measures the distance from the predicted primitives  $\mathbf{P}$  to the point cloud  $\mathbf{X}$  and  $\mathcal{L}_{X \rightarrow P}$  measures the distance from the point cloud  $\mathbf{X}$  to the primitives  $\mathbf{P}$ . We weight the two distance measures in (3) with 1.2 and 0.8, respectively, which empirically led to good results.

**Primitive-to-Pointcloud:** We represent the target point cloud as a set of 3D points  $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ . Similarly, we approximate the continuous surface of primitive  $m$  by a set of points  $\mathbf{Y}_m = \{\mathbf{y}_k^m\}_{k=1}^K$ . Details of our sampling strategy are provided in Section 3.4. This discretization allows us to express the distance between a superquadric and the target point cloud in a convenient form. In particular, for each point on the primitive  $\mathbf{y}_k^m$ , we compute its closest point on the target point cloud  $\mathbf{x}_i$ , and average this distance across all points in  $\mathbf{Y}_m$  as follows:

$$\mathcal{L}_{P \rightarrow X}^m(\mathbf{P}, \mathbf{X}) = \frac{1}{K} \sum_{k=1}^K \Delta_k^m \quad (4)$$

where

$$\Delta_k^m = \min_{i=1, \dots, N} \|\mathcal{T}_m(\mathbf{x}_i) - \mathbf{y}_k^m\|_2 \quad (5)$$

denotes the minimal distance from the  $k$ 'th point  $\mathbf{y}_k^m$  on the  $m$ 'th primitive to the target point cloud  $\mathbf{X}$ . Here,  $\mathcal{T}_m(\mathbf{x}) = \mathbf{R}(\lambda_m) \mathbf{x} + \mathbf{t}(\lambda_m)$  is a function that transforms a 3D point  $\mathbf{x}_i$  in world coordinates into the local coordinate system of the  $m^{th}$  primitive. Note that both  $\mathbf{R}$  and  $\mathbf{t}$  depend on  $\lambda_m$  and are hence estimated by our network.

By taking the expectation wrt. the existence variables  $\mathbf{z}$  and assuming independence of the existence variables:  $p(\mathbf{z}) = \prod_m p(z_m)$ , we obtain the joint loss over all primitives as

$$\begin{aligned} \mathcal{L}_{P \rightarrow X}(\mathbf{P}, \mathbf{X}) &= \mathbb{E}_{p(\mathbf{z})} \left[ \sum_{m=1}^M \mathcal{L}_{P \rightarrow X}^m(\mathbf{P}, \mathbf{X}) \right] \\ &= \sum_{m=1}^M \gamma_m \mathcal{L}_{P \rightarrow X}^m(\mathbf{P}, \mathbf{X}) \end{aligned} \quad (6)$$

Note that this loss encourages the predicted primitives to stay close to the target point cloud.

**Pointcloud-to-Primitive:** While  $\mathcal{L}_{P \rightarrow X}$  measures the distance from the primitives to the point cloud,  $\mathcal{L}_{X \rightarrow P}$  measures the distance from the point cloud to the primitives to ensure that each observation is explained by at least one primitive. We start by defining  $\Delta_i^m$  as the minimal distance from point  $\mathbf{x}_i$  to the surface of the  $m$ 'th primitive:

$$\Delta_i^m = \min_{k=1, \dots, K} \|\mathcal{T}_m(\mathbf{x}_i) - \mathbf{y}_k^m\|_2 \quad (7)$$

Note that in contrast to (5), we minimize over the  $K$  points from the estimated primitive. Similarly to (6), we take the expectation of  $\Delta_i^m$  over  $p(\mathbf{z})$ . In contrast to (6), we sum over each point in the target point cloud  $\mathbf{X}$  and retrieve the distance to the closest primitive  $m$  that exists ( $z_m = 1$ ):

$$\mathcal{L}_{X \rightarrow P}(\mathbf{X}, \mathbf{P}) = \mathbb{E}_{p(\mathbf{z})} \left[ \sum_{\mathbf{x}_i \in \mathbf{X}} \min_{m | z_m = 1} \Delta_i^m \right] \quad (8)$$

Note that naïve computation of Eq. 8 becomes very slow for a large number of primitives  $M$  as it requires evaluating the quantity inside the expectation  $2^M$  times. In this work, we propose a novel approach to simplify this computation that results in a linear number of evaluations. Without loss of generality, let us assume that the  $\Delta_i^m$ 's are sorted in ascending order:

$$\Delta_i^1 \leq \Delta_i^2 \leq \dots \leq \Delta_i^M \quad (9)$$

Assuming this ordering, we can state the following: if the first primitive exists, the first primitive will be the one closest to point  $\mathbf{x}_i$  of the target point, if the first primitive does not exist and the second does, then the second primitive is

closest to point  $\mathbf{x}_i$  and so on and so forth. More formally, this property can be stated as follows:

$$\min_{m|z_m=1} \Delta_i^m = \begin{cases} \Delta_i^1, & \text{if } z_1 = 1 \\ \Delta_i^2, & \text{if } z_1 = 0, z_2 = 1 \\ \vdots \\ \Delta_i^M, & \text{if } z_m = 0, \dots, z_M = 1 \end{cases} \quad (10)$$

This allows us to simplify Eq. 8 as follows

$$\mathcal{L}_{X \rightarrow P}(\mathbf{X}, \mathbf{P}) = \sum_{\mathbf{x}_i \in \mathbf{X}} \sum_{m=1}^M \Delta_i^m \gamma_m \prod_{\bar{m}=1}^{m-1} (1 - \gamma_{\bar{m}}) \quad (11)$$

where  $\gamma_{\bar{m}}$  is a shorthand notation which denotes the existence probability of a primitive closer than primitive  $m$ . Note that this function requires only  $M$ , instead of  $2^M$ , evaluations of the function  $\Delta_i^m$  which is one of the main results of this paper. For a detailed derivation of (11), we refer the reader to the supplementary material.

### 3.2. Parsimony Loss

Despite the bidirectional loss formulation above, our model suffers from the trivial solution  $\mathcal{L}_D(\mathbf{P}, \mathbf{X}) = 0$  which is attained for  $\gamma_1 = \dots = \gamma_m = 0$ . Moreover, multiple primitives with identical parameters yield the same loss function as a single primitive by dispersing their existence probability. We thus introduce a regularizer loss on the existence probabilities  $\gamma$  which alleviates both problems:

$$\mathcal{L}_\gamma(\mathbf{P}) = \max \left( \alpha - \alpha \sum_{m=1}^M \gamma_m, 0 \right) + \beta \sqrt{\sum_{m=1}^M \gamma_m} \quad (12)$$

The first term of (12) makes sure that the aggregate existence probability over all primitives is at least one (i.e., we expect at least one primitive to be present) and the second term enforces a parsimonious scene parse by exploiting a loss function sub-linear in  $\sum_m \gamma_m$  which encourages sparsity.  $\alpha$  and  $\beta$  are weighting factors which are set to 1.0 and  $10^{-3}$  respectively.

### 3.3. Superquadric Parametrization

Having specified our network and the loss function, we now provide details about the superquadric representation and its parameterization  $\lambda$ . Note that, in this section, we omit the primitive index  $m$  for clarity. Superquadrics define a family of parametric surfaces that can be fully described by a set of 11 parameters [1]. The explicit superquadric equation defines the surface vector  $\mathbf{r}$  as

$$\mathbf{r}(\eta, \omega) = \begin{bmatrix} \alpha_1 \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega \\ \alpha_2 \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega \\ \alpha_3 \sin^{\epsilon_1} \eta \end{bmatrix} \quad -\pi/2 \leq \eta \leq \pi/2 \quad -\pi \leq \omega \leq \pi \quad (13)$$

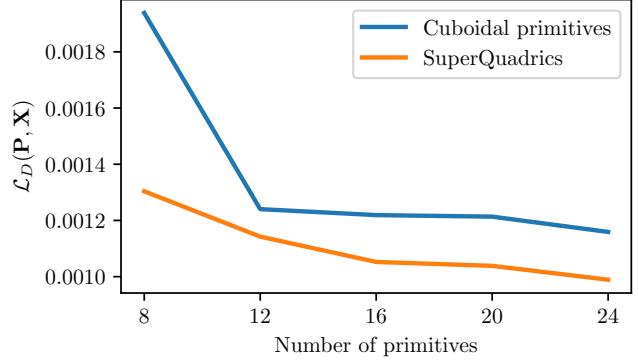


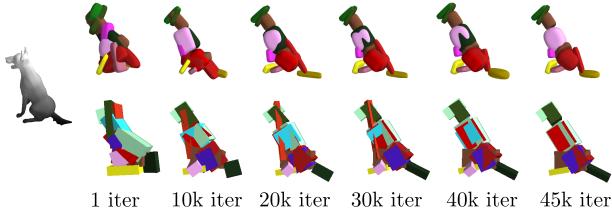
Figure 3: **Reconstruction Loss wrt. #Primitives.** We illustrate the reconstruction loss on the test set of the ShapeNet *animal* category for a different number of primitives. Superquadrics (orange) consistently outperform cuboid primitives (blue) due to their diverse shape vocabulary that allows them to better capture fine details of the input shapes.

where  $\alpha = [\alpha_1, \alpha_2, \alpha_3]$  determine the size and  $\epsilon = [\epsilon_1, \epsilon_2]$  determine the global shape of the superquadric, see supplementary material for examples. Following common practice [39], we bound the values  $\epsilon_1$  and  $\epsilon_2$  to the range  $[0.1, 1.9]$  so as to prevent non-convex shapes which are less likely to occur in practice. Eq. 13 produces a superquadric in a canonical pose. In order to allow any position and orientation, we augment the primitive parameter  $\lambda$  with an additional rigid body motion represented by a translation vector  $\mathbf{t} = [t_x, t_y, t_z]$  and a quaternion  $\mathbf{q} = [q_0, q_1, q_2, q_3]$  which determine the coordinate system transformation  $\mathcal{T}(\mathbf{x}) = \mathbf{R}(\lambda) \mathbf{x} + \mathbf{t}(\lambda)$  above.

### 3.4. Implementation

Our network architecture comprises an encoder and a set of linear layers followed by non-linearities that independently predict the pose, shape and size of the superquadric surfaces. The encoder architecture is chosen based on the input type (e.g. image, voxelized input, etc.). In our experiments, for a binary occupancy grid as input, our encoder consists of five blocks of 3D convolution layers, followed by batch normalization and Leaky ReLU non-linearities. The result is passed to five independent heads that regress translation  $\mathbf{t}$ , rotation  $\mathbf{q}$ , size  $\alpha$ , shape  $\epsilon$  and probability of existence  $\gamma$  for each primitive. Additional details about our network architecture as well as results using an image-based encoder are provided in the supplementary material.

For evaluating our loss (3), we sample points on the superquadric surface. To achieve a uniform point distribution, we sample  $\eta$  and  $\omega$  as proposed in [26]. During training, we uniformly sample 1000 points from the surface of the target object, as well as 200 points from the surface of every superquadric. Note that sampling points on the surface of the objects results in a stochastic approximator of the expected



**Figure 4: Training Evolution.** We visualize the qualitative evolution of superquadrics (top) and cuboids (bottom) during training. Superquadrics converge faster to more accurate representations, whereas cuboids cannot capture details such as the open mouth of the dog, even after convergence.

loss. The variance of this approximator is inversely proportional to the number of sampled points. We experimentally observe that our model is not sensitive to the number of sampled points. For optimization, we use ADAM [19] with learning rate 0.001 and a batch size of 32 for 40k iterations. To further increase parsimony, we then fix all parameters except  $\gamma$  for additional 5k iterations. This step removes remaining overlapping primitives as also observed in [37].

## 4. Experimental Evaluation

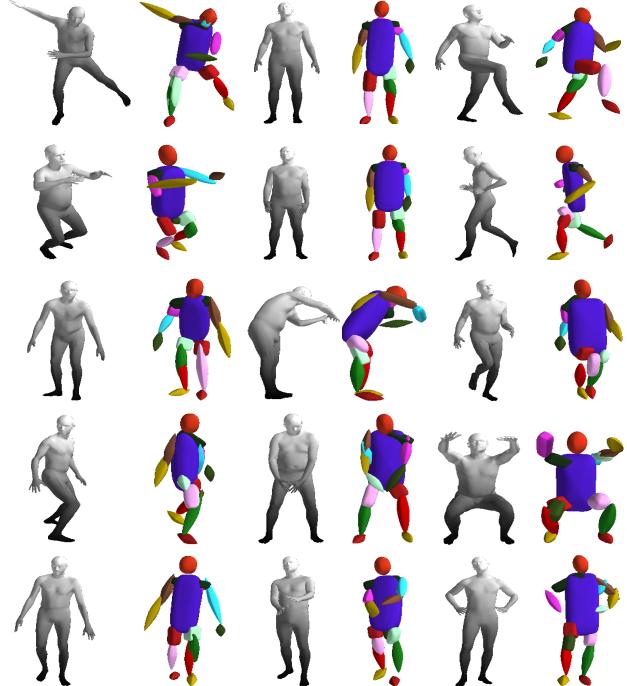
In this section, we present a set of experiments to evaluate the performance of our network in terms of parsing an input 3D shape into a set of superquadric surfaces.

**Datasets:** We provide results on two 3D datasets. First, we use the *aeroplane*, *chair* and *animals* categories from ShapeNet [5]. Following [37], we train one model per object category using a voxelized binary occupancy grid of size  $32 \times 32 \times 32$  as input. Second, we use the SURREAL dataset from Varol et al. [38] which comprises humans in various poses (e.g., standing, walking, sitting). Using the SMPL model [22], we rendered 5000 meshes, from which 4500 are used for training and 500 for testing. For additional qualitative results on both datasets, we refer the reader to our supplementary material.

**Baselines:** Most related to ours is the cuboid parsing approach of Tulsiani et al. [37]. Other approaches to cuboid-based scene parsing [23, 44] require ground-truth shape annotations and thus cannot be fairly compared to unsupervised techniques. We thus compare to Tulsiani et al. [37], using their publicly available code<sup>2</sup>.

### 4.1. Superquadrics vs. Cuboids

We first compare the modeling accuracy of superquadric surfaces wrt. cuboidal shapes which have been extensively used in related work [23, 37, 44]. Towards this goal, we fit *animal* shapes from ShapeNet by optimizing the distance



**Figure 5: Qualitative Results on SURREAL.** Our network learns semantic mappings of body parts across different body shapes and articulations. For instance, the network uses the same primitive for the left forearm across instances.

loss function in (3) while varying the maximum number of allowed primitives  $M$ . To ensure a fair comparison, we use the proposed model for both cases. Note that this is trivially possible as cuboids are a special case of superquadrics. To minimize the effects of network initialization and local minima in the optimization, we repeat the experiment three times with random initializations and visualize the average loss in Fig. 3. The results show that for any given number of primitives, superquadrics consistently achieve a lower loss, and hence higher modeling fidelity. We further visualize the qualitative evolution of the network during training in Fig. 4. This figure demonstrates that compared to cuboids, superquadrics better model the object shape, and more importantly that the network is able to converge faster.

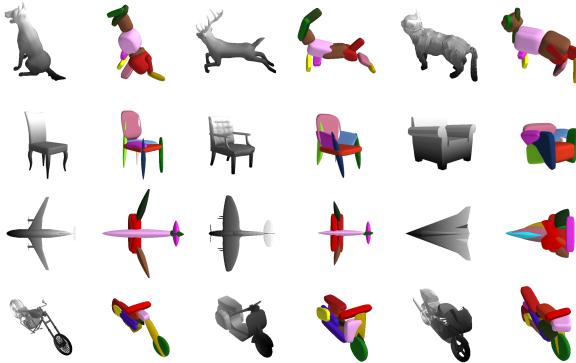
### 4.2. Results on ShapeNet

We evaluate the quality of the predicted primitives using our reconstruction loss from (3) on the ShapeNet dataset and compare to the cuboidal primitives as estimated by Tulsiani et al. [37]. We associate every primitive with a unique color, thus primitives illustrated with the same color correspond to the same object part. For both approaches we set the maximal number of primitives to  $M = 20$ . From Fig. 6, we observe that our predictions consistently capture both the structure as well as fine details (e.g., body, tails, head), whereas the corresponding cuboidal primitives from [37]

<sup>2</sup><https://github.com/shubhtuls/volumetricPrimitives>



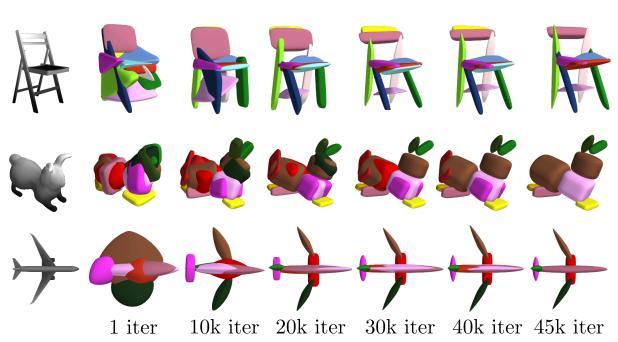
**Figure 6: Qualitative Results on ShapeNet.** We visualize predictions for the object categories *animals*, *aeroplane* and *chairs* from the ShapeNet dataset. The top row illustrates the ground-truth meshes from every object. The middle row depicts the corresponding predictions using the cuboidal primitives estimated by [37]. The bottom row shows the corresponding predictions using our learned superquadric surfaces. Similarly to [37], we observe that the predicted primitive representations are consistent across instances. For example, the primitive depicted in green describes the right wing of the aeroplane, while for the animals class, the yellow primitive describes the front legs of the animal.



**Figure 7: Attention to Details.** Superquadrics allow for modeling fine details such as the tails and ears of animals as well as the wings and the body of the airplanes and wheels of the motorbikes which are hard to capture using cuboids.

focus mainly on the structure of the predicted object.

Fig. 7 shows additional results in which our model successfully predicts animals, airplanes and also more complicated motorbike parts. For instance, we observe that our model is able to capture the open mouth of the dog using two superquadrics as shown in Fig. 7 (left-most animal in third row). In addition, we notice that our model dynamically allocates a variable number of primitives depending on the complexity of the input shape. For example, the left-most airplane in Fig. 6, is modelled with 6 primitives whereas the jetfighter (right-most) that has a more complicated shape is modelled with 9 primitives. This can also be observed for the animal category, where our model chooses a single primitive for the body of the cat (right-most animal in Fig. 6) while for all the rest it uses two. We remark that our expressive shape abstractions allow for differentiating between different types of objects such



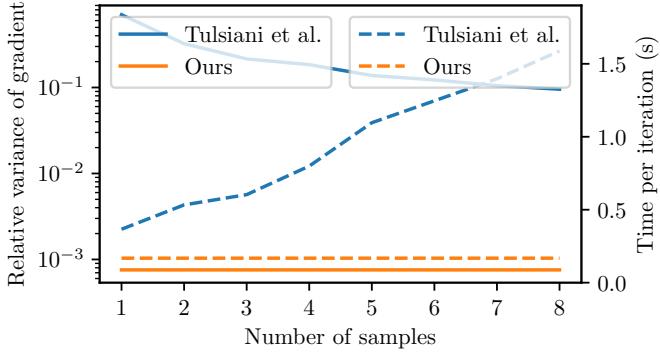
**Figure 8: Evolution of Primitives.** We illustrate the evolution of superquadrics during training. Note how our model first focuses on the overall structure of the object and starts to attend to finer details at later iterations.

as scooter/chopper/racebike or airliner/fighter by truthfully capturing the shape of the individual object parts.

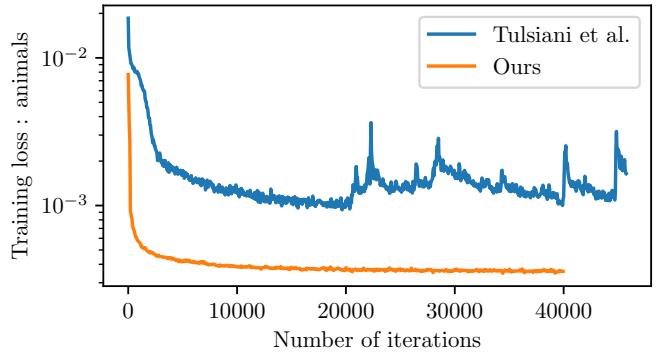
Fig. 8 visualizes the training evolution of the predicted superquadrics for three object categories. While initially, the model focuses on the overall structure of the object using mostly blob-shaped superquadrics ( $\epsilon_1$  and  $\epsilon_2$  close to 1.0), as training progresses it starts attending to details. After convergence, the predicted superquadrics closely match the shape of the corresponding (unknown) object parts.

### 4.3. Results on SURREAL

In addition to ShapeNet, we also demonstrate results on the SURREAL human body dataset in Fig. 5. The benefits of superquadrics over simpler shape abstractions are accentuated in this dataset due to the complicated shapes of the human body. Note that our model successfully captures details that require modeling beyond cuboids: For instance, our model predicts pointy octahedral shapes for the feet,



(a) Gradient Variance and Iteration Time.



(b) Evolution of Training Loss.

Figure 9: Fig. 9a depicts the variance of the gradient estimates for  $\gamma$  over 300 iterations (solid) as well as the computation time per iteration (dashed) for [37] (blue) and our method (orange). Our analytical loss function provides gradients with orders of magnitude less variance while at the same time decreasing runtime. In Fig. 9b, we compare the training loss evolution of [37] (blue) to ours (orange). The sampling based approach of [37] leads to large oscillations while ours converges smoothly.

ellipsoid shapes for the head and a flattened elongated superellipsoid for the main body without any supervision on the primitive parameters. Another interesting aspect of our model is the consistency of the predicted primitives, i.e., the same primitives (highlighted with the same color) consistently represent feet, legs, arms etc. across different poses. For more complicated poses, correspondences are sometimes mirrored. We speculate that this behavior is caused by symmetries of the human body.

#### 4.4. Analytical Loss Formulation

In this section, we compare the evolution of our training loss in Equation (3) to the evolution of the training loss proposed by Tulsiani et al. [37] in Fig. 9b. While it is important to mention that the absolute values are not comparable due to the slightly different loss formulations, we observe that our loss converges faster with less oscillations. Note that at iteration 20k, [37] starts updating the existence probabilities using reinforcement learning [41] which further increases oscillations. In contrast, our loss function decays smoothly and does not require sampling-based gradient estimation.

To further analyze the advantages of our analytical loss formulation, we calculate the variance of the gradient estimates for the existence probabilities  $\gamma$  over 300 training iterations. Fig. 9a compares the variance of the gradients of [37] to the variance of the gradients of the proposed analytical loss (solid lines). Note that the variance in our gradient is orders of magnitude lower compared to [37] as we do not require sampling [41] for approximating the gradients. Simultaneously, we obtain a lower runtime per iteration (dashed lines). While using more samples lowers the variance of gradients approximated using Monte Carlo estimation [37], runtime per iteration increases linearly with the number of samples. In contrast, our method does not require sampling and outperforms [37] in terms of runtime

	Chamfer Distance			Volumetric IoU		
	Chairs	Aeroplanes	Animals	Chairs	Aeroplanes	Animals
[37]	0.0121	0.0153	0.0110	0.1288	0.0650	0.3339
Ours	<b>0.0006</b>	<b>0.0003</b>	<b>0.0003</b>	<b>0.1408</b>	<b>0.1808</b>	<b>0.7506</b>

Table 1: **Quantitative evaluation.** We report the mean Chamfer distance (smaller is better) and the mean Volumetric IoU (larger is better) for our model compared to [37].

even for the case of gradient estimates based on a single sample. We remark that in both cases the runtime is computed for the entire iteration, considering both, the forward and the backward pass. A quantitative comparison is provided in Table 1. Note that in contrast to [37] we optimize for the Chamfer distance.

## 5. Conclusion and Future Work

We propose the first learning-based approach for parsing 3D objects into consistent superquadric representations. Our model successfully captures both the structure as well as the details of the target objects by accurately learning to predict superquadrics in an unsupervised fashion from data.

In future work, we plan to extend our model by including parameters for global deformations such as tapering and bending. We anticipate that this will significantly benefit the fitting process as the available shape vocabulary will be further increased. Finally, we also plan to extend our model to large-scale scenes. We believe that developing novel hierarchical strategies as in [44] is key for unsupervised 3D scene parsing at room-, building- or even city-level scales.

## Acknowledgments

We thank Michael Black for early discussions on superquadrics. This research was supported by the Max Planck ETH Center for Learning Systems.

## References

- [1] Alan H Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications (CGA)*, 1981. 2, 3, 5, 11, 12
- [2] Irving Biederman. Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing*, 1986. 1, 3
- [3] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological Review*, 94(2):115, 1987. 1
- [4] I Binford. Visual perception by computer. In *IEEE Conference on Systems and Control*, 1971. 1
- [5] 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. *arXiv.org*, 1512.03012, 2015. 2, 6, 11, 15
- [6] Laurent Chevalier, Fabrice Jaillet, and Atilla Baskurt. Segmentation and superquadric modeling of 3d objects. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2003. 3
- [7] Christopher Bongsoo Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 1, 2
- [8] Peter Elias and Lawrence G Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963. 1
- [9] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Joshua B. Tenenbaum. Learning to infer graphics programs from hand-drawn images. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 2
- [10] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 4
- [11] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 1, 2
- [12] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. AtlasNet: A papier-mâché approach to learning 3d surface generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2
- [13] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. *arXiv.org*, 1704.00710, 2017. 1, 2
- [14] Wilfried Hartmann, Silvano Galliani, Michal Havlena, Luc Van Gool, and Konrad Schindler. Learned multi-patch similarity. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 1, 2
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 18, 19
- [16] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1
- [17] Ales Jaklic, Ales Leonardis, and Franc Solina. *Segmentation and Recovery of Superquadrics*, volume 20 of *Computational Imaging and Vision*. Springer, 2000. 3
- [18] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. SurfaceNet: an end-to-end 3d neural network for multiview stereopsis. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 1, 2
- [19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 6, 19
- [20] David H Laidlaw, W Benjamin Trumbore, and John F Hughes. Constructive solid geometry for polyhedral objects. In *ACM Trans. on Graphics*, 1986. 2
- [21] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2
- [22] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. on Graphics*, 2015. 6
- [23] Chengjie Niu, Jun Li, and Kai Xu. Im2struct: Recovering 3d shape structure from a single RGB image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 3, 6
- [24] Despoina Paschalidou, Ali Osman Ulusoy, Carolin Schmitt, Luc van Gool, and Andreas Geiger. Raynet: Learning volumetric 3d reconstruction with ray potentials. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2
- [25] Alex Pentland. Parts: Structured descriptions of shape. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 1986. 1, 2, 3
- [26] Maurizio Pilu and Robert B. Fisher. Equal-distance sampling of superellipse models. In *Proc. of the British Machine Vision Conf. (BMVC)*, 1995. 6
- [27] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 1
- [28] G.J. Qi, X.S. Hua, Y. Rui, T. Mei, J. Tang, and H.J. Zhang. Concurrent multiple instance learning for image categorization. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007. 2
- [29] Danilo Jimenez Rezende, S. M. Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 2
- [30] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

- [31] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. [2](#), [3](#)
- [32] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. [2](#)
- [33] Franc Solina. Volumetric models in computer vision—an overview. *Journal of Computing and Information technology*, 1994. [3](#)
- [34] Franc Solina and Ruzena Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 1990. [2](#), [3](#)
- [35] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. [2](#)
- [36] Demetri Terzopoulos and Dimitris N. Metaxas. Dynamic 3d models with local and global deformations: deformable superquadrics. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 1990. [2](#)
- [37] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. [1](#), [2](#), [3](#), [4](#), [6](#), [7](#), [8](#), [11](#), [15](#), [19](#), [20](#), [21](#)
- [38] GÜl Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. [2](#), [6](#), [11](#)
- [39] Narunas Vaskevicius and Andreas Birk. Revisiting superquadric fitting: A numerically stable formulation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2017. [5](#)
- [40] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. [2](#)
- [41] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. [3](#), [8](#)
- [42] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. [1](#)
- [43] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. [1](#), [2](#)
- [44] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. [1](#), [2](#), [3](#), [6](#), [9](#)

# Supplementary Material for Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids

Despoina Paschalidou<sup>1,4</sup> Ali Osman Ulusoy<sup>2</sup> Andreas Geiger<sup>1,3,4</sup>

<sup>1</sup>Autonomous Vision Group, MPI for Intelligent Systems Tübingen

<sup>2</sup>Microsoft <sup>3</sup>University of Tübingen <sup>4</sup>Max Planck ETH Center for Learning Systems

`{firstname.lastname}@tue.mpg.de`

## Abstract

*In this supplementary document, we present a detailed derivation of the proposed analytical solution to the Chamfer loss, which avoids the need for computationally expensive reinforcement learning or iterative prediction. Moreover, we also present additional qualitative results on more complex object categories from the ShapeNet dataset [5] such as cars and motorbikes and on the SURREAL human body dataset [38]. Furthermore, we also show results on primitive prediction when using RGB images instead of 3D occupancy grids as input. Finally, we empirically demonstrate that our bi-directional Chamfer loss formulation indeed works better and results in less local minima than the original bi-directional loss formulation of Tulsiani et al. [37].*

## A. Superquadrics

In this work, we propose superquadrics as a shape primitive representation. Their simple parametrization in combination to their ability to represent a diverse class of shapes makes superquadrics a natural choice for geometric primitives. Moreover, their continuous parametrization is suitable for deep learning as their shape varies continuously with their parameters. Superquadrics are fully modelled using a set of 11 parameters [1]. The **explicit superquadric equation** defines the surface vector  $\mathbf{r}$

$$\mathbf{r}(\eta, \omega) = \begin{bmatrix} \alpha_1 \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega \\ \alpha_2 \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega \\ \alpha_3 \sin^{\epsilon_1} \eta \end{bmatrix} \quad \begin{array}{l} -\pi/2 \leq \eta \leq \pi/2 \\ -\pi \leq \omega \leq \pi \end{array} \quad (14)$$

where  $\alpha = [\alpha_1, \alpha_2, \alpha_3]$  determine the size and  $\epsilon = [\epsilon_1, \epsilon_2]$  determine the global shape of the superquadric. Fig. 10 visualizes the shape of superquadrics for different values of  $\epsilon_1$  and  $\epsilon_2$ . In addition to the shape parameters, we also associate a rigid body transformation with each superquadric. This transformation is represented by a translation vector  $\mathbf{t} = [t_x, t_y, t_z]$  and a quaternion  $\mathbf{q} = [q_0, q_1, q_2, q_3]$  that determines the coordinate system transformation  $\mathcal{T}(\mathbf{x}) = \mathbf{R}(\lambda) \mathbf{x} + \mathbf{t}(\lambda)$  from world coordinates to local primitive-centric coordinates. This transformation as well as the angles  $\eta, \omega$  and the scale parameters  $\alpha_1, \alpha_2, \alpha_3$  are illustrated in Fig. 11.

## B. Derivation of Pointcloud-to-Primitive Loss

This section provides the derivation of the pointcloud-to-primitive distance  $\mathcal{L}_{X \rightarrow P}(\mathbf{X}, \mathbf{P})$  in Eq. 11 of the main paper. For completeness, we restate our notation briefly. We represent the target point cloud as a set of 3D points  $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$  and we approximate the continuous surface of the  $m^{th}$  primitive by a set of 3D points  $\mathbf{Y}_m = \{\mathbf{y}_k^m\}_{k=1}^K$ . We further denote  $\mathcal{T}_m(\mathbf{x}) = \mathbf{R}(\lambda_m) \mathbf{x} + \mathbf{t}(\lambda_m)$  as the mapping from world coordinates to the local coordinate system of the  $m^{th}$  primitive.

The pointcloud-to-primitive distance,  $\mathcal{L}_{X \rightarrow P}$ , measures the distance from the point cloud to the primitives to ensure that each observation is explained by at least one primitive. It can be expressed as:

$$\mathcal{L}_{X \rightarrow P}(\mathbf{X}, \mathbf{P}) = \mathbb{E}_{p(\mathbf{z})} \left[ \sum_{\mathbf{x}_i \in \mathbf{X}} \min_{m|z_m=1} \Delta_i^m \right] \quad (15)$$

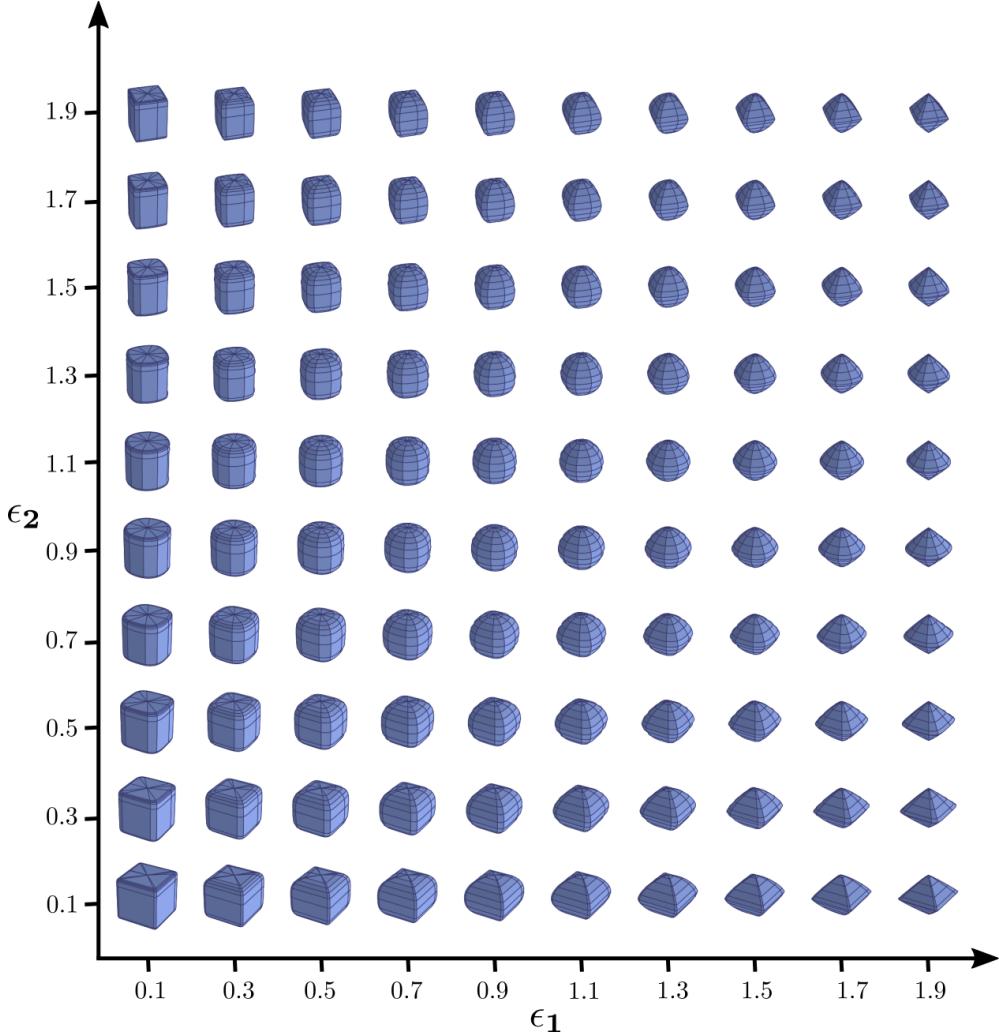


Figure 10: **Superquadric Shape Space.** Superquadrics are a parametric family of surfaces that can be used to describe cubes, cylinders, spheres, octahedral ellipsoids, etc. [1]. This figure visualizes superquadrics when varying the shape parameters  $\epsilon_1$  and  $\epsilon_2$ , while keeping the size parameters  $\alpha_1, \alpha_2$  and  $\alpha_3$  constant.

where  $\Delta_i^m$  denotes the minimal distance from point  $\mathbf{x}_i$  to the surface of the  $m$ 'th primitive:

$$\Delta_i^m = \min_{k=1,\dots,K} \|\mathcal{T}_m(\mathbf{x}_i) - \mathbf{y}_k^m\|_2 \quad (16)$$

Assuming independence of the existence variables  $p(\mathbf{z}) = \prod_m p(z_m)$ , we can replace the expectations in (15) with summations as follows:

$$\mathcal{L}_{X \rightarrow P}(\mathbf{X}, \mathbf{P}) = \sum_{z_1} \cdots \sum_{z_M} \left[ \sum_{\mathbf{x}_i \in \mathbf{X}} \min_{m | z_m = 1} \Delta_i^m \right] p(\mathbf{z}) \quad (17)$$

Naïve computation of (17) has exponential complexity, i.e. for  $M$  primitives it requires evaluating the quantity inside the expectation  $2^M$  times. Our key insight is that (17) can be evaluated in linear time if the distances  $\Delta_i^m$  are sorted. Without loss of generality, we assume that the distances are sorted in ascending order. This allows us to state the following: if the first primitive exists, the first primitive will be the one closest to point  $\mathbf{x}_i$  of the target point, if the first primitive does not exist and the second does, then the second primitive is closest to point  $\mathbf{x}_i$  and so forth. More formally, this property can be stated

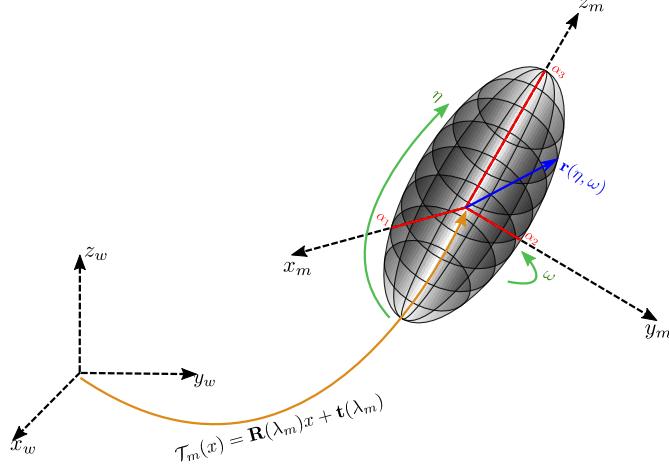


Figure 11: **Explicit Superquadric Equation.** A 3D vector  $\mathbf{r}(\eta, \omega)$  defines a closed surface in space as  $\eta$  (latitude angle) and  $\omega$  (longitude angle) change in the given intervals (14). The rigid body transformation  $T_m(x)$  maps a point from the world coordinate system to the local coordinate system of the  $m^{th}$  primitive.

as follows:

$$\min_{m|z_m=1} \Delta_i^m = \begin{cases} \Delta_i^1, & \text{if } z_1 = 1 \\ \Delta_i^2, & \text{if } z_1 = 0, z_2 = 1 \\ \vdots \\ \Delta_i^M, & \text{if } z_m = 0, \dots, z_M = 1 \end{cases} \quad (18)$$

Using (18) we can simplify (17) as follows. We start to carry out the summations over the existence variables one by one. Starting with the summations over  $z_1$ , (17) becomes:

$$\mathcal{L}_{X \rightarrow P}(\mathbf{X}, \mathbf{P}) = \sum_{\mathbf{x}_i \in \mathbf{X}} \left[ \underbrace{\gamma_1 \sum_{z_2} \cdots \sum_{z_M} \Delta_i^1 \prod_{\bar{m}=2}^M p(z_{\bar{m}})}_{(\dagger)} + (1 - \gamma_1) \sum_{z_2} \cdots \sum_{z_M} \left[ \min_{m \geq 2 | z_m=1} \Delta_i^m \right] \prod_{\bar{m}=2}^M p(z_{\bar{m}}) \right] \quad (19)$$

The expression, marked with  $(\dagger)$  corresponds to the case for  $z_1 = 1$ , namely the  $1^{st}$  primitive is part of the scene. From Eq. 18, we know that  $\min_{m|z_m=1} \Delta_i^m = \Delta_i^1$  for  $z_1 = 1$ , thus the expression marked with  $(\dagger)$ , can be simplified as follows,

$$(\dagger) = \gamma_1 \Delta_i^1 \underbrace{\sum_{z_2} \cdots \sum_{z_M} \prod_{\bar{m}=2}^M p(z_{\bar{m}})}_{\text{this term evaluates to 1}} = \gamma_1 \Delta_i^1 \quad (20)$$

Following this strategy, we can iteratively simplify the remaining terms in (19) and arrive at the analytical form of the pointcloud-to-primitive distance stated in Eq. 11 in the main paper:

$$\begin{aligned} \mathcal{L}_{X \rightarrow P}(\mathbf{X}, \mathbf{P}) &= \sum_{\mathbf{x}_i \in \mathbf{X}} [\gamma_1 \Delta_i^1 + (1 - \gamma_1) \gamma_2 \Delta_i^2 + \cdots + (1 - \gamma_1)(1 - \gamma_2) \cdots \gamma_M \Delta_i^M] \\ &= \sum_{\mathbf{x}_i \in \mathbf{X}} \sum_{m=1}^M \Delta_i^m \gamma_m \prod_{\bar{m}=1}^{m-1} (1 - \gamma_{\bar{m}}) \end{aligned} \quad (21)$$

Note that our current formulation assumes that at least one primitive exists in the scene. However, this assumption can be easily relaxed by introducing a “virtual primitive” with a fixed distance to every 3D point on the target point cloud.

## C. Qualitative Results on SURREAL

In this section, we provide additional qualitative results on the SURREAL human body dataset. In Fig. 12, we illustrate the predicted primitives of humans in various poses and articulations.



**Figure 12: Qualitative Results on SURREAL.** Our network learns semantic mappings of body parts across different body shapes and articulations. For instance, the network uses the same primitive for the left forearm across instances.

We remark that our model is able to accurately capture the various human body parts using superquadric surfaces. Another

interesting aspect of our model, which is also observed in [37], is related to the fact that our model uses the same primitive (highlighted with the same color) to represent the same actual human body part. For example, the head is typically captured using the primitive illustrated with red. For some poses these correspondences are lost. We speculate that this is because the network does not know whether the human is facing in front or behind.

## D. Qualitative Results on ShapeNet

In this section, we provide additional qualitative results on various object types from the ShapeNet dataset [5]. We also demonstrate the ability of our model to capture fine details in more complicated objects such as *motorcycle-bikes* and *cars*. Due to their diverse shape vocabulary, superquadrics can accurately capture the structure of complex objects such as motorbikes and cars. We observe that our model successfully represents the wheels of all bikes using a flattened ellipsoid and the front fork using a pointy ellipsoid. Again, we note that our network consistently associates the same primitive with the same semantic part. For instance, for the motorcycles object category, the primitive colored in red is associated with the saddle, the primitive colored in green is associated with the front wheel etc. Fig. 15+16 demonstrate several predictions for both classes using superquadric surfaces as geometric primitives.

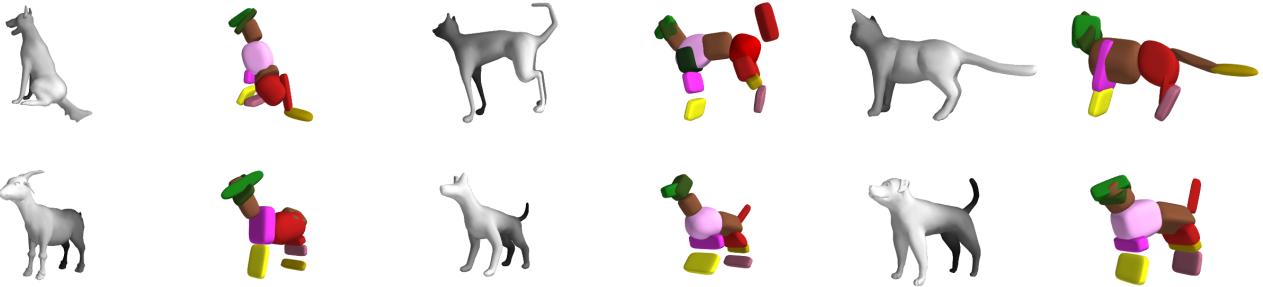


Figure 13: **Qualitative Results on *animals* from ShapeNet.** We visualize the predictions of our network on the *animals* class of the ShapeNet dataset. We remark the consistency across primitives and animal parts as well as the ability of our model to capture details such as ears and tails of animals that could have not been captured using cuboidal primitives

Due to superquadrics’ large shape vocabulary, our approach derives expressive scene abstractions that allow for differentiating between different types of vehicles both for motorcycles (scooter, racing bike, chopper etc.) (Fig. 15), cars (sedan, convertible, coupe, etc.) (Fig. 16), animals (dogs, cats) (Fig. 13) despite that our model leverages only up to 20 primitives per object. Note that for cars, wheels are not as easily recovered as for motorbikes due to the lack of supervision and since they are “geometrically occluded” by the body of the car.



Figure 14: **Qualitative Results on *chairs* from ShapeNet.** We visualize the predictions of our network on the *chairs* class of the ShapeNet dataset. We observe the consistency across corespondences between primitives and object parts as well as the ability of our model to capture the shape of rounded parts.

Fig. 13+14 depicts additional predictions on the *animal* and the *chair* object class of the ShapeNet dataset. We observe that for both categories our model consistently captures both the structure and the fine details of the depicted object. Note that chairs that have rounded legs are associated with flattened ellipsoids (Fig. 14), this would not have been possible only with cuboids.

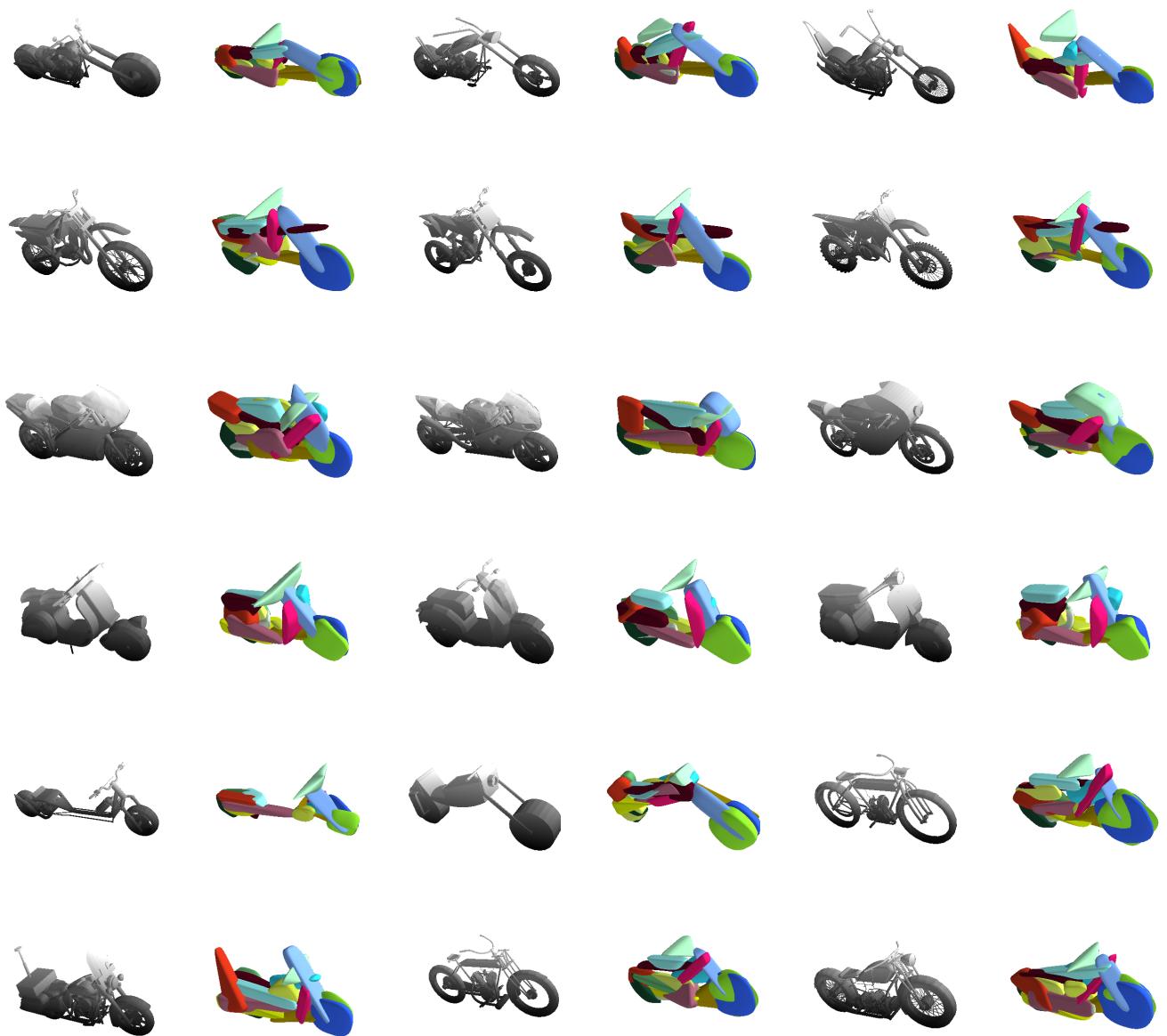


Figure 15: **Qualitative Results on *motorbikes* from ShapeNet.** Our network learns semantic mappings of various object parts of different objects within the same category. Our expressive shape abstractions allow for differentiating between different types of motorbikes (scooter, racing bike, chopper etc.), by successfully capturing the shape of various indicative parts such as the wheels or the front fork of the bike.

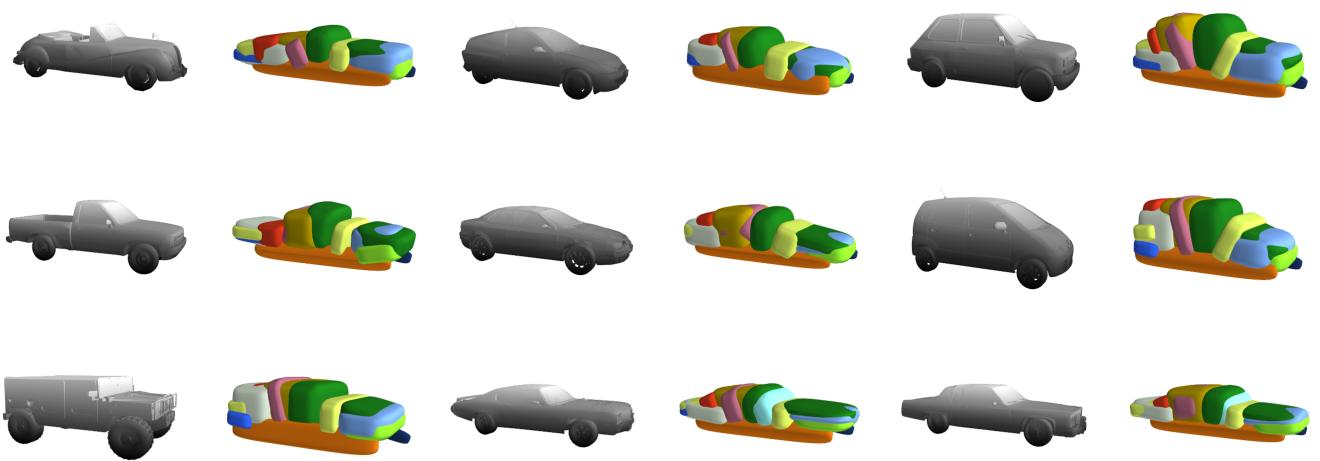


Figure 16: **Qualitative Results on *cars* from ShapeNet.** We visualize predictions for the object categories *car* from the ShapeNet dataset. Our expressive shape abstractions allow us to differentiate between different car types such as sedan, coupe etc. This would not have been possible with cuboidal primitives that cannnot model rounded surfaces

## E. Network Architecture Details

In this section, we detail the network architecture used throughout our experimental evaluations. Our network comprises of two main parts, an encoder that learns a low-dimensional feature representation for the input and five regressors that predict the parameters of the superquadrics (size  $\alpha$ , shape  $\epsilon$ , translations  $\mathbf{t}$ , rotations  $\mathbf{q}$  and  $\gamma$  probabilities of existence). As we already explained in our main submission, the encoder architecture is chosen based on the input type (image, voxelized input etc.). In our experiments, we consider a binary occupancy grid as an input and the sequence of layers comprising both the encoder and the regressors are depicted in Figure 17.

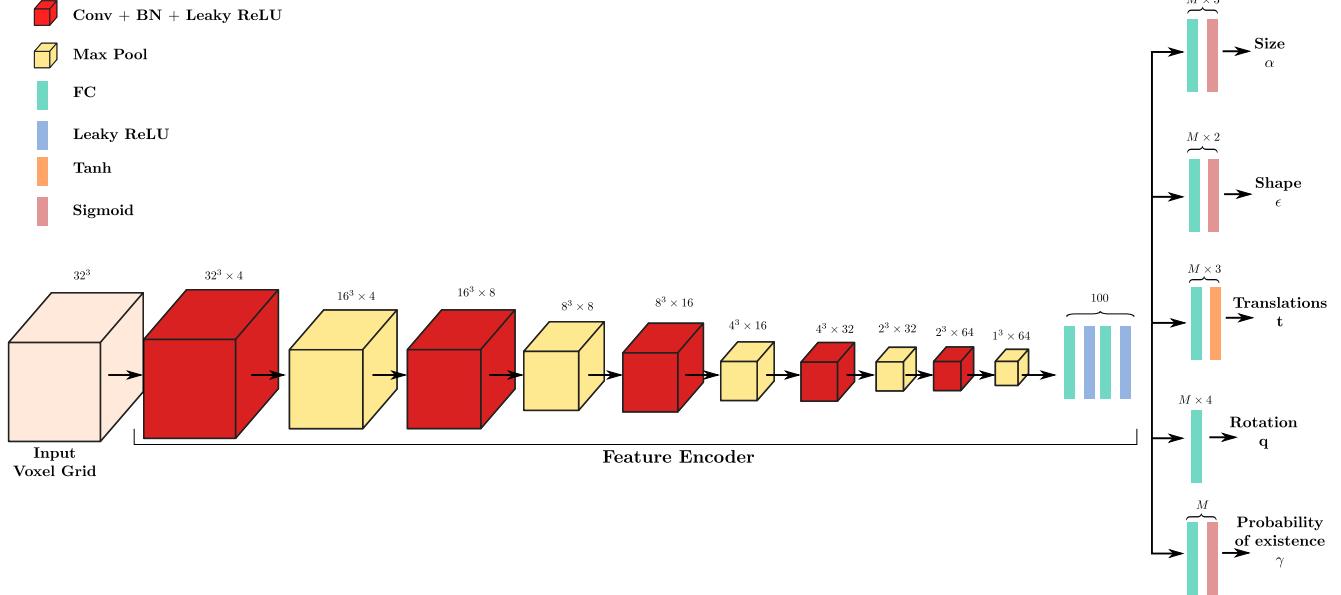


Figure 17: **Volume-based network architecture.** We visualize the layers that comprise our network architecture. Cubes denote operations that are conducted on 3-dimensional volumes, while rectangles correspond to  $K$ -dimensional features. The number above each shape (cube or rectangle) corresponds to the dimensionality of that layer. For instance,  $16^3 \times 4$  denotes a feature map of size  $16^3$  and 4 channels. Following, our notation,  $M$  corresponds to the maximum number of primitives predicted.

Note that, for the image-based experiment of section F in the supplementary, where we consider an image as an input to our model, we replace the encoder architecture in Fig. 17 with a ResNet18 [15].

### E.1. Parsimony Loss Details

We would also like to briefly provide some additional details for our parsimony loss. For completeness, we restate the parsimony loss of Equation 12 in our main submission,

$$\mathcal{L}_\gamma(\mathbf{P}) = \max \left( \alpha - \alpha \sum_{m=1}^M \gamma_m, 0 \right) + \beta \sqrt{\sum_{m=1}^M \gamma_m} \quad (22)$$

Note that the  $\sum_{m=1}^M \gamma_m$  corresponds to the expected number of primitives in the predicted parsing. As already mentioned, in our main submission, our model suffers from the trivial solution  $\mathcal{L}_D(\mathbf{P}, \mathbf{X}) = 0$  which is attained for  $\gamma_1 = \dots = \gamma_M = 0$ . To avoid this solution, we introduce the first term of Eq. 22 that penalizes the prediction when the expected number of primitives is less than 1. The second term penalizes the prediction when the expected number of primitives is large. Note that the maximum value of the second term is  $\beta\sqrt{M}$ , while the maximum value of the first term is  $\alpha$ . Therefore, in order to allow the model to use more than one primitive, we set  $\beta$  to a value smaller than  $\alpha$ . Typically  $\alpha = 1.0$  and  $\beta = 10^{-3}$ .

## F. Shape Abstraction from a Single RGB Image

In this section, we use the proposed reconstruction loss of Eq. 3, in the main submission, to extract shape primitives from RGB images instead of occupancy grids. Towards this goal, we render the ShapeNet models to images, and train an image-based network to minimize the same reconstruction loss also used for our volume-based architecture.

More specifically, we replace the encoder architecture, described in Section 3.4 in our main submission, with the ResNet18 architecture [15], without the last fully connected layer. The extracted features are subsequently passed to five independent heads that regress translation  $t$ , rotation  $q$ , size  $\alpha$ , shape  $\epsilon$  and probability of existence  $\gamma$  for each primitive. During training, we uniformly sample 1000 points, from the surface of the target object, as well as 200 points from the surface of every superquadric. For optimization, we use ADAM [19] with a learning rate of 0.001 and a batch size of 32 for 40k iterations. We observe that our model accurately captures shape primitives even from a single RGB image as input.

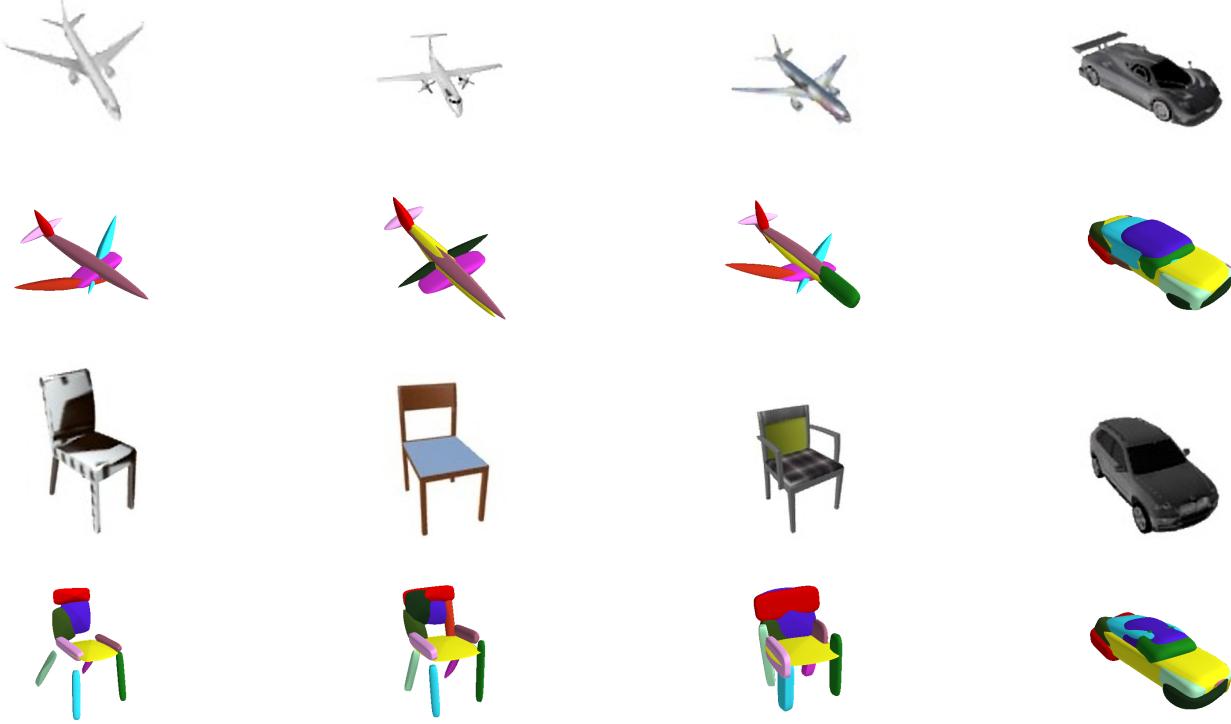


Figure 18: **Shape Abstraction from a Single RGB Image.** We visualize predictions for various ShapeNet object categories using a single RGB image as input to our model.

## G. Quantitative Analysis

In this section, we provide additional details regarding the quantitative comparison of Table 1 in our main paper. For evaluation, we report two metrics the mean *Chamfer distance* and the mean *Volumetric IoU*. Volumetric IoU is defined as the quotient of the volume of the two meshes' intersection and the volume of their union. We obtain unbiased estimates of the volume of the intersection and the union by randomly sampling points from the bounding volume and determining if the points lie inside or outside the ground truth / predicted mesh. The computation of the Chamfer distance is discussed in detail in our main submission throughout Section 3. Regarding the comparison in Table 1 of our main submission, we want to mention that cuboids are a special case of superquadrics, thus fitting objects with cuboids is expected to lead to worse results compared to superquadrics.

## H. Empirical Analysis of Reconstruction Loss

In this section, we provide empirical evidence regarding our claim that our Chamfer-based reconstruction loss leads to more stable training compared to the truncated bi-directional loss of Tulsiani et al. [37]. Towards this goal, we directly opti-

mize/train for the primitive parameters, i.e., not optimizing the weights of a neural network but directly fitting the primitives. We perform this experiment on a 2D toy example and compare the results when using the proposed loss to the results using the truncated distance formulation in [37]. We visualize the evolution of parameters for both optimization objectives as training progresses. We observe that the truncated loss proposed in [37] is more likely to converge to local minima (e.g. figures 20k-20o), while our loss consistently avoids them.



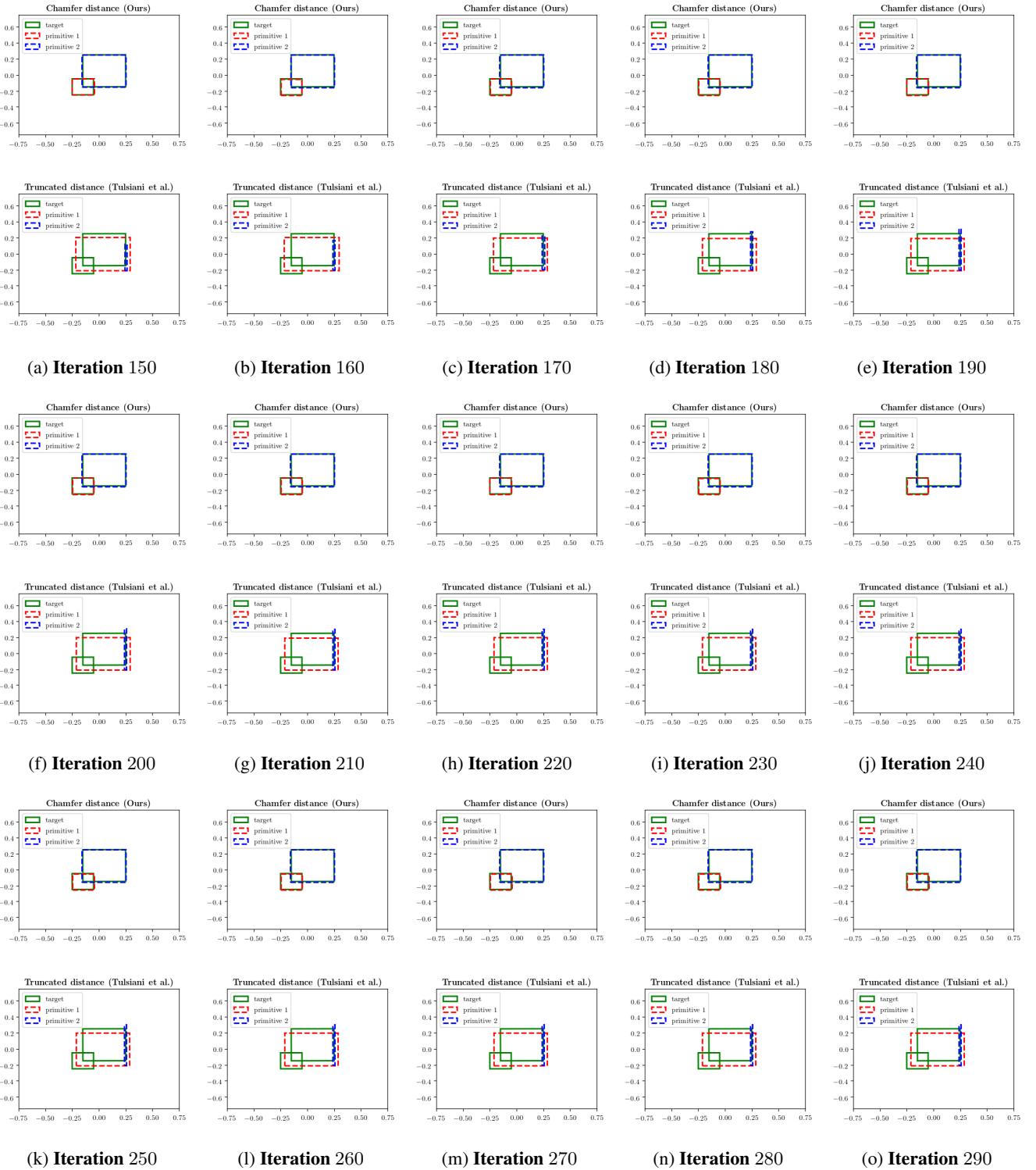


Figure 20: **Empirical Analysis of Reconstruction Loss.** We illustrate the evolution of two cuboid abstractions using our reconstruction loss with Chamfer distance and the truncated bi-directional loss of Tulsiani et al. [37].