Hierarchical Superquadric Decomposition with Implicit Space Separation

Jaka Šircelj[1,2], **Peter Peer**[1], **Franc Solina**[1], **Vitomir Štruc**[2]

[1]*Faculty of Computer and Information Science, UNI-LJ, Večna pot 113, Ljubljana, Slovenia*
[2]*Faculty of Electrical Engineering, UNI-LJ, Tržaška cesta 25, Ljubljana, Slovenia*

Abstract

We introduce a new method to reconstruct 3D objects using a set of volumetric primitives, i.e., superquadrics. The method hierarchically decomposes a target 3D object into pairs of superquadrics recovering finer and finer details. While such hierarchical methods have been studied before, we introduce a new way of splitting the object space using only properties of the predicted superquadrics. The method is trained and evaluated on the ShapeNet dataset. The results of our experiments suggest that reasonable reconstructions can be obtained with the proposed approach for a diverse set of objects with complex geometry.

1 Introduction

A vital field in 3D vision and robotics is 3D object reconstruction. By compressing input data into simpler representations such as meshgrids or sets of simpler objects, we can understand and interact which objects and their surroundings more easily, solving problems like collision avoidance [7] or grasp planning [2], [9].

A common type of reconstruction is to represent a given (3D) object with a set of simpler shape primitives, often referred to as *geons*. A popular approach in geon reconstruction is to estimate a fixed amount of geons and, concurrently, predict which of those should be kept in the final geon set, thus retaining the variability in geon numbers [8], [6]. In our previous work, for example, we used a MaskRCNN model to first segment the input depth image and then reconstruct the parts using a specific type of geons, called superquadrics [10].

Following the overall idea of our prior work, we introduce a novel method to reconstruct 3D objects with a set of superquadrics (SQ) in this paper. However, in this novel approach we base the procedure on a hierarchical tree decomposition algorithm. Instead of using a model that returns a fixed amount of geons at once, we introduce a hierarchical decomposition model, that incrementally splits the input object into more and more SQ representations, as illustrated in Figure 1. While a similar method was introduced by Paschalidou *et al.* in [5], we propose a splitting method based on the superquadrics characteristic, alleviating the model from determining how the object should be hierarchically split.

We evaluate the proposed approach on the ShapeNet dataset. Two models with different reconstruction capabilities are trained for the experiments. The first, facilitating reconstructions with a maximum of 4 superquadric, is trained and tested on the full ShapeNet dataset. The second, enabling reconstructions with up to 8 superquadrics, is assessed on the pistol ShapeNet subset. Results are presented in terms of IoU scores and with visual examples, and point to the feasability of the proposed solution.

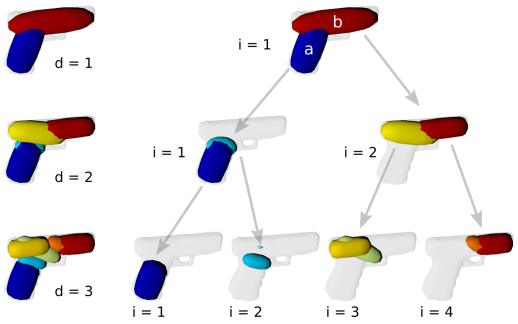

Figure 1: Visualization of the hierarchical decomposition with superquadrics. Each step splits the object into a pair of superquadrics (a and b). The split is driven by previous superquadrics as shown by the arrows. The tree levels are indexed with d, whereas the SQ pairs in each level are indexed with i.

2 Method

2.1 Superquadrics

To reconstruct the 3D shapes we use superquadrics, which are geometric shapes that can describe objects such as spheres, ellipsoids, cylinders and rectangular cuboids. A common way to describe the surface of a superquadric is using the implicit function $F(x, y, z) = 1$. Here F is called the *inside-outside function* and is defined as

$$F(x,y,z) = \left(\left(\frac{x}{a_1} \right)^{\frac{2}{\varepsilon_2}} + \left(\frac{y}{a_2} \right)^{\frac{2}{\varepsilon_2}} \right)^{\frac{\varepsilon_2}{\varepsilon_1}} + \left(\frac{z}{a_3} \right)^{\frac{2}{\varepsilon_1}}, \tag{1}$$

where a_1, a_2, a_3 define the size of the superquadric and $\varepsilon_1, \varepsilon_2$ its shape. We can also move the superquadric in space by offsetting the x, y, z coordinates by t_1, t_2, t_3 respectively and by rotating the coordinates using quaternion notation q_0, q_1, q_2, q_3. To abbreviate the notation we write the spatial coordinates using vector notation $\boldsymbol{x} = [x, y, z]$,

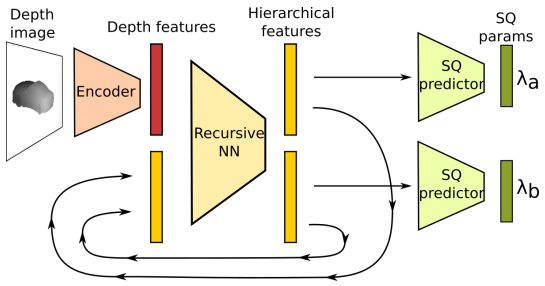

Figure 2: Recursive model architecture.

and all superquadric parameter as

$$\boldsymbol{\lambda} = [a_1, a_2, a_3, \varepsilon_1, \varepsilon_2, t_1, t_2, t_3, q_1, q_2, q_3, q_4].$$

A useful property of the *inside-outside function*, that will be used further on, is that it contains information whether the space is inside or outside the superquadric (hence the name). Points with $F(\boldsymbol{x}) < 1$ lie inside the object, $F(\boldsymbol{x}) > 1$ lie outside, and, as already suggested, points with $F(\boldsymbol{x}) = 1$ lie on its surface. For numerical stability we rather compare $F(\boldsymbol{x})^{\varepsilon_1}$ values [3], the spatial inequalities remains the same.

2.2 Model architecture

Similarly to Paschalidou *et al.* [5] we use a hierarchical decomposition procedure to reconstruct a complex 3D object using a set of superquadrics. This allows the decomposition to predict different numbers of superquadrics, increasing their number for fine detail, and decreasing for larger parts of the object. The procedure works by first extracting a feature vector out of the input depth image, which we refer to as a *depth feature*. This is concatenated with a *hierarchical feature* and passed into a recursive neural network which predicts two new *hierarchical features* each encoding data to predict a new superquadric. This prediction is done with an *SQ predictor* network which predicts the parameters $\boldsymbol{\lambda}$ previously described. Each of the two *hierarchical features* can again be concatenated with the depth feature vector and passed into the recursive neural network, thus splitting the initial superquadric feature into two. A hierarchical feature in the first step can be a simple vector filled with zeroes. The model architecture corresponding to the outlined idea is shown in Figure 2.

Such a model produces a superquadric-pair binary tree, where each node contains two sets of parameters and two links to two children, again containing two sets of parameters. We use the following notation to reference the superquadric parameters in the tree:

$$\text{SQ\_pair}_{d;i} = [\boldsymbol{\lambda}_a, \boldsymbol{\lambda}_b], \tag{2}$$

where d is the depth of the node and i is the index of the node in the tree. This notation is also shown in the tree example in Figure 1.

2.3 Training

For training, we use the natural property of the *inside-outside* function, whose sign describes the inside and the outside space of the superquadric. Similarly to [5], we

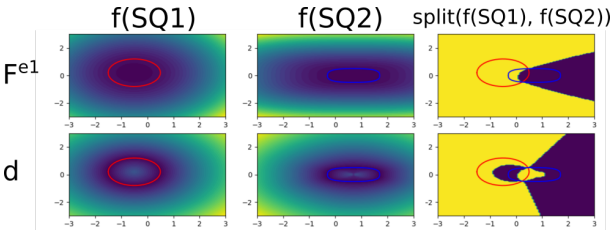

Figure 3: Rows: top - computations on the $f = F^{\varepsilon_1}$ function, bottom - computations on the radial Euclidean distance function ($f = d$). Columns: first - values of the f function around the first superquadric, shown in red, second - values of f around the second superquadric, shown in blue, third - space split according to the f function. We can see that F and F^{ε_1} split space better inside of the superquadrics while the radial distance function splits space more evenly outside of the superquadrics.

define the occupancy function $g(\boldsymbol{x}; \boldsymbol{\lambda})$, which translates this property to act as a binary classifier of the inside space

$$g(\boldsymbol{x}; \boldsymbol{\lambda}) = \sigma(s \quad (1 \quad F^{\ 1}(\boldsymbol{x}; \boldsymbol{\lambda}))), \tag{3}$$

where σ is the sigmoid function, and s is a scaling parameter that defines the slant of the values around the surface of the superquadric – for more information see [5].

Since we want to use this occupancy property of the superquadrics, we generate the training dataset by sampling points in space and annotating them with 1 if they lie inside the 3D object and 0 if they lie outside. The root node of the superquadric pair tree, Eq. (2), is trained on these sets of initial ground-truth points and labels, denoted as P and $L_{1;1}$ respectively.

The child SQ pairs should only cover the space in proximity to the space occupied by their parent superquadric, and should not be fitted to the full inside of the 3D model i.e. $L_{1;1}$. Paschalidou *et al.* [5] solve this problem by also predicting the centroid of the part covered by a predicted superquadric. The space is then split by matching every point to its closest centroid (see [5] and Figure 4). In this paper, we use the mathematical properties of superquadrics to split the space. More specifically, we found that it is best to split the space inside and outside of the superquadric pairs differently. The space inside of the superquadrics is split using the *inside-outside* function by finding the maximum of the *inside-outside* function, i.e.:

$$\arg\max_{i=a;b} F^{\varepsilon_1}(\boldsymbol{x}; \boldsymbol{\lambda}_i). \tag{4}$$

Conversely, the space outside of the superquadric pair is split using the radial Euclidean distance

$$d(\boldsymbol{x}) = j\boldsymbol{x}jj1 \quad F^{\ \frac{\varepsilon_1}{2}}j, \tag{5}$$

taking the superquadric with the minimum value

$$\arg\min_{i=a;b} d_i(\boldsymbol{x}). \tag{6}$$

This distance function is used outside because it behaves more properly as a distance function further away from a superquadric. For more information on the radial distance function, look at Jaklič *et al.* [3]. The space-splitting

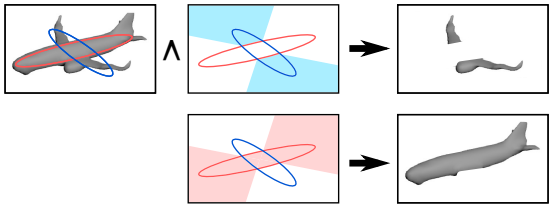

Figure 4: Left: Airplane with its insides noted dark $\mathcal{L}_{1;1}$, center top: outline of SQ 1 with its assigned space after split, center bottom: outline of SQ 2 with its assigned space, right: two outcomes of Equation (7), top is $\mathcal{L}_{2;1}$, bottom $\mathcal{L}_{2;2}$.

Table 1: IoU scores of ModelS (full ShapeNet).

SQ tree level	1	2
IoU [in %]	56.7%	58.8%

properties of the *inside-outside* function and the *radial Euclidean* distance function are illustrated in Figure 3. The proposed splitting method is simpler than the one used in [5], since we don't need to predict any additional values, and uses the space splitting capabilities of the superquadrics, thus, splitting the space more naturally and in accordance with the geometry of the superquadrics.

Since our model obtains a hierarchy of superquadrics, we must also construct a hierarchy of labels $L_{d;i}$ on which we compute the losses. These labels are computed recursively according to how the space is split using the superquadric pairs. The inside of the object is considered inside for the SQ-pair child only if its parent superquadrics' split space also contains it (see Figure 4). This translates to a logical AND operation between the parents ground truth occupancy $L_{d;i}$ and the space split of the parent

$$l_{d;i} = l_{parent\_node(d;i)} \wedge split(\boldsymbol{x}, \boldsymbol{\lambda}_{parent\_sq(d;i)}, \boldsymbol{\lambda}_{uncle\_sq(d;i)}), \tag{7}$$

where $l_{d;i} \in L_{d;i}$ is the label belonging to point $\boldsymbol{x} \in X$, $parent\_node(d,i)$ denotes the parent SQ pair node, $parent\_sq(d,i)$ is the parent superquadric (a or b) and $uncle\_sq(d,i)$ is the other superquadric from the parent pair node. For example, for node $(d,i) = (2,2)$:

$$\begin{aligned} parent\_node(2,2) &= (1,1) \\ parent\_sq(2,2) &= (1,1,a) \\ uncle\_sq(2,2) &= (1,1,b). \end{aligned}$$

Finally the occupancy loss is calculated using the occupancy values for each pair tree node and the ground truth occupancy values $L_{d;i}$

$$L = \sum_{d=1}^{} \sum_{i=1}^{2^{d-1}} \sum_{\substack{\boldsymbol{x} \in X \\ l_{d,i} \in L_{d,i}}} L_{BCE}(\max_{sq=a;b} g(\boldsymbol{x}; \boldsymbol{\lambda}_{sq}), l_{d;i}), \tag{8}$$

where L_{BCE} is the binary cross entropy loss. This loss is similar to the *part-reconstruction loss* term from Paschalidou *et al.* [5], but differs in that we look at how accurately an SQ pair matches the designated space of its parent superquadric.

Table 2: Per object category IoU scores of ModelS (full ShapeNet). Superquadrics taken from the last/second tree layer.

Label	**IoU** [in %]	**Label**	**IoU** [in %]
dishwasher	86.9%	rocket	64.7%
microwave	84.3%	printer	64.2%
bus	82.1%	monitor	62.9%
washer	81.2%	watercraft	61.9%
file	80.4%	bag	61.4%
can	78.1%	piano	60.4%
pillow	77.9%	plane	57.6%
car	76.8%	rifle	52.2%
phone	76.1%	knife	50.8%
laptop	73.1%	chair	47.5%
cabinet	71.8%	table	47.4%
speaker	69.9%	bench	46.7%
sofa	68.8%	lamp	41.9%
camera	67.4%	bathtub	41.2%
pistol	67.2%	bookcase	40.2%
mailbox	65.3%	bowl	34.3%
basket	23.8%		

Table 3: IoU scores of ModelP (pistols).

SQ tree level	1	2	3
IoU	63.5%	65.6%	64.7%

3 Results

In this section, we present the results for two trained hierarchic decomposition models. We use the ShapeNet dataset [1] for training along with the NVIDIA Kaolin v0.1 library [4], which is used to compute signed distance function values of the 3D objects and to infer the ground truth inside-outside space. The first model (abbreviated as ModelP) is trained on the Pistol subset of ShapeNet with the maximum depth of the superquadric pair tree set to 3 levels. The second model (ModelS) is trained on the full ShapeNet dataset with the maximum depth set to 2 level.

Performance Indicator. After a train-validation-test split, both models are evaluated on the test subset using the intersection-over-union metric (IoU)

$$IoU(\Lambda_d, L_{0;0}) = \frac{\sum_{\substack{\boldsymbol{x} \in X \\ l \in L_{0,0}}} \hat{l}_d(\boldsymbol{x}) \wedge l}{\sum_{\substack{\boldsymbol{x} \in X \\ l \in L_{0,0}}} \hat{l}_d(\boldsymbol{x}) \vee l}, \tag{9}$$

where Λ_d denotes all predicted superquadric parameters in the tree at layer d, $\Lambda_d = \{\lambda_{d;i} | i \in [1, 2^d - 1]\}$ and $\hat{l}_d(\boldsymbol{x})$ denotes the predicted inside-outside label for point $\boldsymbol{x}$. The latter is inferred as inside or $\hat{l}_d(\boldsymbol{x}) = 1$ if $g(\boldsymbol{x}; \boldsymbol{\lambda}_{d;i}) > 0.5$, for any superquadric in layer d, otherwise the point is outside of the reconstruction.

Model Comparison. The IoU results of the ModelS on the full ShapeNet dataset are shown in Table 1 and for separate subsets of Shapenet in Table 2. The IoU results of ModelP for the pistol subset are reported in Table 3. Since the full ShapeNet dataset contains much more variability, ModelS scores a lower IoU over the full ShapeNet dataset, than ModelP over the pistol subset. An interesting observation is that ModelS still performs better on the pistol