CMPT464 Project Report
Fitting 3D Shapes with Superellipsoid

Siu Yu Chau[1]

[1]syc54@sfu.ca

1 Abstract

Abstracting 3D shapes using geometric primitives is a fundamental topic in computer vision and computer graphics. In this paper, we explore the use of superellipsoids as the primary geometric primitive for shape representation. Leveraging a neural network-based approach, we estimate the parameters defining the superellipsoid while also predicting their existence probabilities. Our results demonstrate that superellipsoids are particularly effective for approximating simple convex shapes with minimal cavities, highlighting their potential in applications requiring efficient and flexible 3D shape modeling.

2 Introduction

The traditional way to describe a shape involves vertices, meshes, and point clouds. However, these representations can be memory-intensive and inefficient. To address this, we aim to find a more compact way to represent shapes without compromising their visual fidelity. Using geometric primitives is an excellent alternative. For example, representing a sphere using a mesh-based approach may require hundreds of sample points to define its surface, whereas a primitive-based representation only requires parameters like radius and position—resulting in significant compression. While increasing the number of primitives generally improves the representation's accuracy, our goal is to use as few primitives as possible to maintain memory and computational efficiency.

Traditional methods for primitive fitting include techniques like Random Sample Consensus (RANSAC) and Principal Component Analysis (PCA). However, the past decade has seen significant advancements in deep neural networks and parallel computing power, making neural network-based approaches an attractive alternative to traditional methods.

In this project, we followed the structure and approach of Paschalidou's paper[1]. We adopt a modern deep neural network-based approach due to its potential to provide

a simpler and more elegant solution to the problem. Specifically, we utilize the existing architecture OctNet and incorporate two loss functions: reconstruction loss and parsimony loss. Our approach involves passing 64×64×64 voxel inputs through the neural network to compute the optimal parameters for fitting superellipsoids to the given shapes.

3 Related Work

3.1 Primitive-based shape representation

One of the earliest learning-based approaches to volumetric primitive fitting is presented by Tulsiani et al. in 2016[2]. In their work, they employed a two-stage training process, where the model takes voxel input and outputs the parameters of a cuboid. Since then, many subsequent papers have built upon this approach, introducing variations such as accepting image input instead of voxel data and experimenting with different types of primitives [3] [4].

3.2 Shape fitting algorithms

There are various approaches for fitting algorithms, one of which employs an encoder-decoder architecture [6]. Other authors have utilized Invertible Neural Networks to generate primitives [5]. In our approach, an encoder combined with regressors architecture was used [1].

3.3 Relevant evaluation metrics

Chamfer distance is used as an evaluation metrics for the primitives fitting. It measure the similarity between two point cloud. Another metric is the Volumetric Intersection over Union (Volumetric IoU) [1], the intersection is the volume of space shared by the predicted 3D object and the ground truth object; the union is the total volume covered by both the predicted and ground truth objects.

4 Method

4.1 Primitive definition and parameterization

A superellipsoid is a 3D geometric shape that generalizes the concept of an ellipsoid. It is defined by a set of equations that combine spherical and rectangular coordinates, and it can take a variety of shapes depending on its parameters. Superellipsoids are useful for modelling a wide range of shapes in computer graphics, computer vision, and 3D modelling due to their flexibility and simplicity compared to more complex models like meshes. Some special cases of superellipsoid are sphere, cube, and bi-cone.

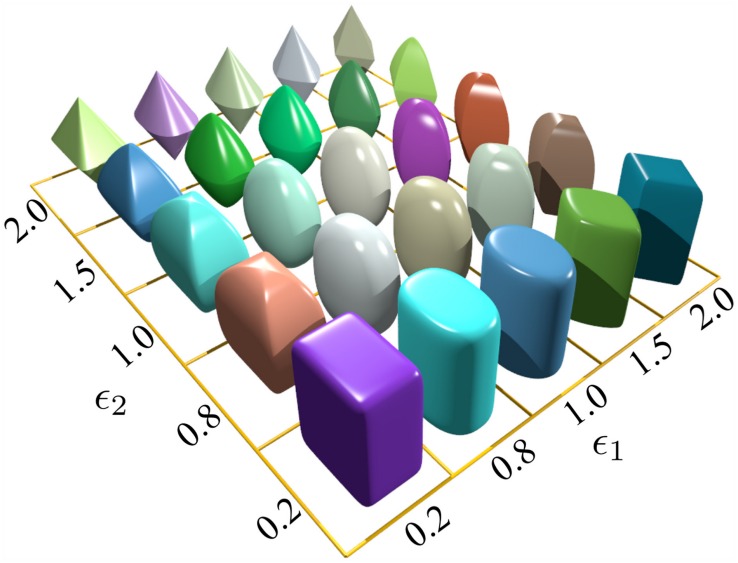


Figure 1: Superellipsoids with different squareness

The parameterization of superellipsoid can be described as follows:

$$\begin{aligned} x(\eta,\omega) &= \alpha_1 \cos^{\epsilon_1}\eta \cos^{\epsilon_2}\omega \\ y(\eta,\omega) &= \alpha_2 \cos^{\epsilon_1}\eta \sin^{\epsilon_2}\omega \\ z(\eta,\omega) &= \alpha_3 \sin^{\epsilon_1}\eta \end{aligned} \tag{1}$$

where η and ω are variables along the longitude and latitude respectively. α describes the scale of the super ellipsoid along the x, y, z direction. And ϵ represents the squareness along the longitude and latitude. This parameterization will later be used to sample points on the surface

As shown in Fig.1, we can see that increasing the value of ϵ makes the shape sharper. However, if ϵ becomes too large, the shape may become concave and excessively sharp, which is not ideal for fitting most 3D shapes, as they are typically convex. Therefore, it is recommended to keep the ϵ value within the range of $[0.1, 1.9]$.

The implicit function of superellipsoid can be described as follows:

$$f(x,y,z) = \left(\left(\frac{x}{\alpha_1}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{\alpha_2}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{\alpha_3}\right)^{\frac{2}{\epsilon_1}} \tag{2}$$

If $f(x,y,z)$ at a point is evaluated to be $= 1$, then this point is on the surface, if it is > 1, then it is outside the superellipsoid; if it is < 1, then it is inside the superellipsoid. The implicit function is helpful to determine the SDF of the superellipsoid.

4.2 Fitting algorithm details

In this project we will use the same neural network architecture as[1]. This network consists of a feature encoder and five regressors for the output parameters of the ellipsoid. The output parameters will be 12 values. Which are three size parameters:

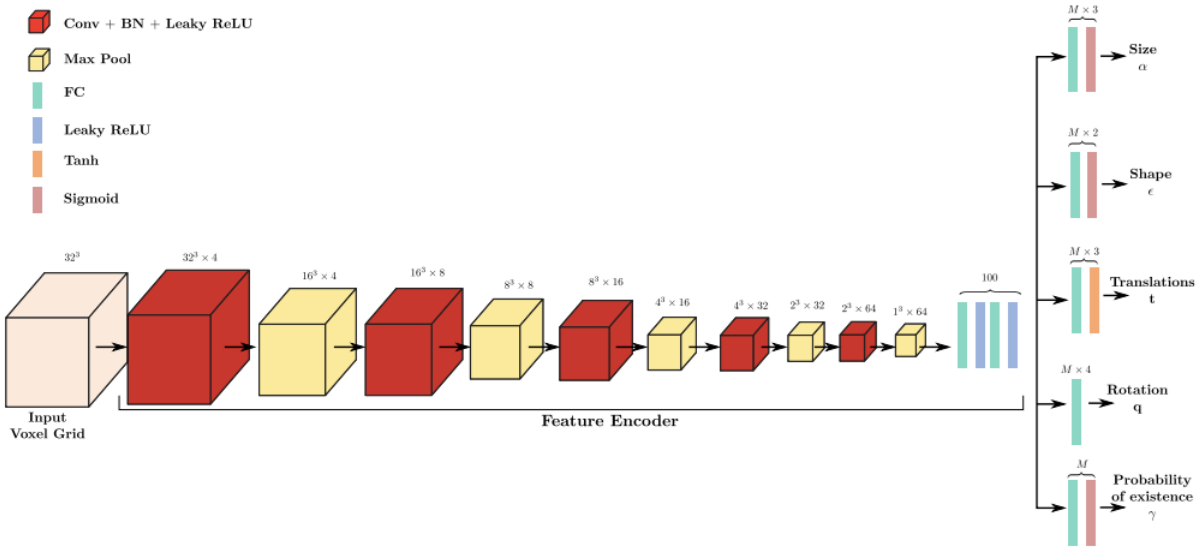


Figure 2: Neural Network structure: OctNet

$\alpha_1, \alpha_2, \alpha_3$; two shape parameter: ϵ_1, ϵ_2; three translation parameters: t_x, t_y, t_z; four rotational parameters(quaternion): q_1, q_2, q_3, q_4; and one probability of existence γ_m for each primitive.

The neural network we used is shown in Fig.2. We modified it to accept 64x64x64 binary occupancy grid. We expanded the network so that it reduced from 64^3 dimension to 1^3 x 64. To obtain the predicted parameters, we first feed the voxel into the network. The network generates a prediction, and a loss value is computed based on the prediction. Using this loss, we perform backpropagation to adjust the network's parameters. After several iterations, the network is trained to provide an accurate prediction for primitive fitting.

4.3 Loss functions/optimization objectives

The loss function will be composed of two parts, a reconstruction loss which is calculated by chamfer distance, and a parsimony loss which encourages sparsity and penalizes trivial solutions.

$$\mathcal{L} = \mathcal{L}_{\text{reconstruction}} + \mathcal{L}_{\text{parsimony}} \tag{3}$$

The reconstruction loss is shown below. Since we want to address the asymmetry of chamfer distance. We want to include both primitive the target(left term) and target to primitive(right term) distance. the $\mathcal{L}^m_{P\to X}(P, X)$ term represent the distance from a primitive P to the target object X. z is a random variable that represents the existence of the primitive. The $\min_{m|z_m=1} \Delta^m_i$ term represents the minimal distance of a point x_i to the surface of m'th primitive. We take the expectation, that the higher the

probability of existence of the primitive, the more contribution it will have.

$$\mathcal{L}_{\text{reconstruction}} = \mathbb{E}_{p(z)}[\sum_{m=1}^{M} \mathcal{L}_{P\to X}^{m}(P, X)] + \mathbb{E}_{p(z)}[\sum_{x_i \in X} \min_{m|z_m=1} \Delta_i^m] \tag{4}$$

The parsimony loss consists of two key components. The left max term ensures that the minimum number of primitives is at least one, thereby preventing trivial solutions where all existence probabilities are 0. The right term acts as a sparsity regularizer, encouraging reducing the number of primitives. Typically, the coefficients are set as $\alpha = 1$ and $\beta = 10^{-3}$. These terms help the model determine the optimal number of primitives required for an effective fit.

$$\mathcal{L}_{\text{parsimony}} = \max(\alpha - \alpha \sum_{m=1}^{M} \gamma_m, 0) + \beta \sqrt{\sum_{m=1}^{M} \gamma_m} \tag{5}$$

4.4 Implementation details

The optimizer used is ADAM, with a learning rate of 0.0005. Training is conducted over 2000 iterations, with 500 samples generated for each primitive. The maximum number of primitives is capped at 24. An existence probability threshold of 0.7 is applied, discarding any primitives with probabilities below this value. These parameters will serve as the baseline for our result outputs. A significant portion of the code used in this project is adapted from Paschalidou's paper [1], as their implementation is both well-written and highly optimized.

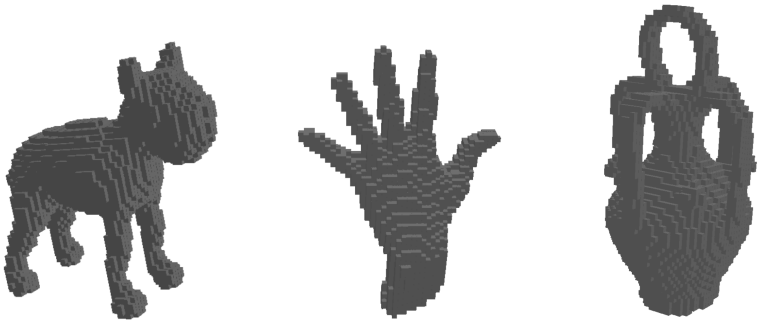
5 Experiments

5.1 Dataset description

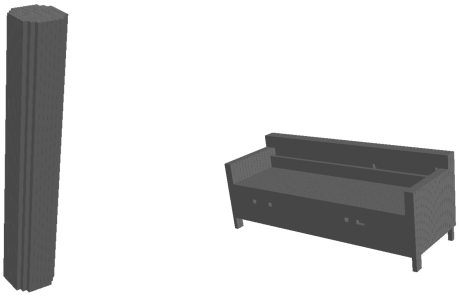
The dataset we used includes five models: dog, hand, pot, rod, and sofa. These models are available in various formats, such as voxels, point clouds, and signed distance fields (SDFs). Given their diverse surfaces, geometries and complexity, we expect our model to perform better on some models while potentially underperforming on others. Overall, this dataset provides a comprehensive representation of the general applicability and robustness of our model and design.

5.2 Evaluation metrics

For evaluation, we used Chamfer Distance and Volumetric Intersection over Union (IoU). Chamfer Distance measures the similarity between two point clouds by calculating the average distance from each point in one cloud to the nearest point in the other. On the other hand, Volumetric IoU evaluates the overlap between predicted and ground truth 3D shapes, providing a quantitative measure of their intersection relative to their union.

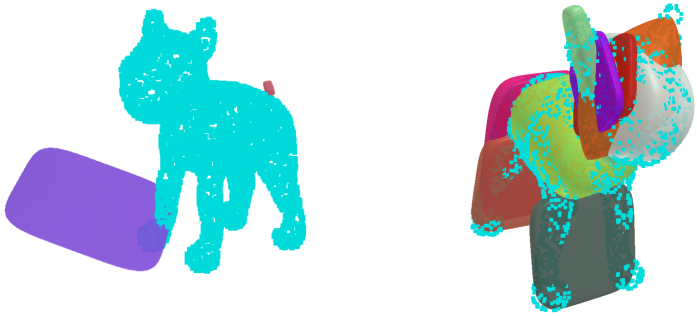


(a) Dog (b) Hand (c) Pot



(d) Rod (e) Sofa

Figure 3: Dataset models



(a) No reconstruction loss (b) No parsimony loss

5.3 Baseline implementation

The experiment will be conducted on an RTX 3060 GPU using the CUDA platform. Each model's 64x64x64 binary occupancy grid will be fed into the network and trained individually. Due to computational limitations, the experiment will be limited to 2000 iterations to ensure that the results are produced within a reasonable time frame. But more iteration can produce better results.

5.4 Ablation studies

As shown in Fig. 4a, reconstruction plays a crucial role in the model's performance. In Fig. 4b, the effect of removing the parsimony loss is evident. Without this constraint, the number of primitives increases significantly, and they tend to overlap more.

5.5 Quantitative results

Iteration	Dog	Hand	Pot	Rod	Sofa
1	0.14229	0.09451	0.13627	0.18053	0.14070
400	0.00223	0.00130	0.00247	0.00063	0.00472
800	0.00214	0.00128	0.00235	0.00056	0.00448
1200	0.00207	0.00129	0.00235	0.00056	0.00448
1600	0.00206	0.00128	0.00234	0.00056	0.00446
2000	0.00200	0.00128	0.00234	0.00056	0.00447

Table 1: Chamfer Distance Results for Different Models and Iterations

5.6 Qualitative visualizations

The result can be seen in Fig.5.

6 Discussion

6.1 Result analysis

For the dog model, 6 primitives were used; however, two of them overlap in the body. Additionally, the legs are represented by two large superellipsoids instead of four individual superellipsoids, one for each leg.

For the hand model, the ideal representation would consist of 5 long, thin superellipsoids for the fingers and 1 large superellipsoid for the palm. However, the model failed to achieve this.

For the pot model, the model struggles due to the many cavities, and it resorts to using a large cube that covers everything, which doesn't capture the details well.

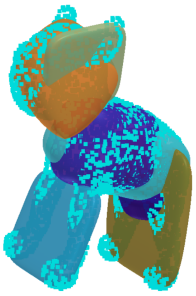
For the rod model, the fit is very good, but more primitives were used than expected.

For the sofa model, the model successfully captures the sofa frame but fails to represent the side handles.

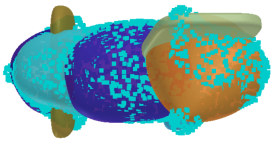
Overall, the results show that the model tends to use a suitable number of primitives to represent each object. Additionally, the coverage is generally excellent, with the point cloud remaining very close to the surface of the primitives.

6.2 Limitations

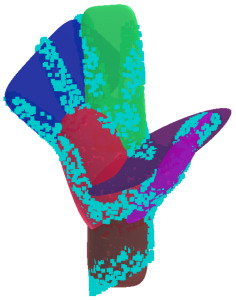
Overall, the model struggles to accurately capture objects with non-box-like shapes, complex surface areas, and numerous cavities. This is expected, as the superellipsoid is a convex shape and is not well-suited for representing such models.



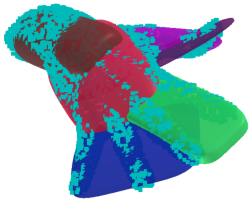
(a) Dog result 1



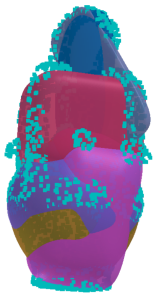
(b) Dog result 2



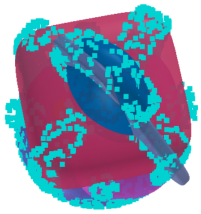
(c) Hand result 1



(d) Hand result 2

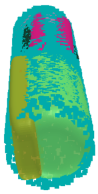


(e) Pot result 1

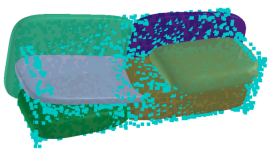


(f) Pot result 2

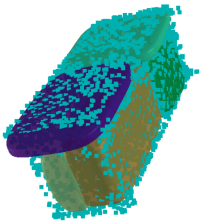
(g) Rod result 1



(h) Rod result 2



(i) Sofa result 1



(j) Sofa result 2

Figure 5: Result of fitting dataset models

Another issue is that the model is not very efficient or fast. This may be due to the sampling process or the Chamfer distance slowing down the training. Running over 5000 iterations can take more than 20 minutes.

6.3 Future improvements

Overlapping remains a significant issue in the results presented above. To address this, we can introduce an overlapping loss function in the future, which would encourage each primitive to avoid overlapping with others. This would help the model fit more precisely by reducing redundant parts and ensuring a more distinct representation of each primitive.

Additionally, the superellipsoid shape can be enhanced by incorporating deformations. Currently, it only represents basic convex shapes. However, to model more complex surfaces, such as the detailed geometry of the hand model, we can explore deforming the superellipsoid. This would make the primitives more expressive and capable of better capturing intricate surface details.

References

[1] Despoina Paschalidou, Ali Hakan Ulusoy, and A. Geiger, "Superquadrics Revisited: Learning 3D Shape Parsing Beyond Cuboids," Computer Vision and Pattern Recognition, 2019

[2] Shubham Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik, "Learning Shape Abstractions by Assembling Volumetric Primitives," Computer Vision and Pattern Recognition, 2017

[3] Niu, Chengjie, et al. "Im2Struct: Recovering 3D Shape Structure from a Single RGB Image." Computer Vision and Pattern Recognition, 2018,

[4] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomir Mech. Parsenet: A parametric surface fitting network for 3d point clouds. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16, pages 261–276. Springer, 2020.

[5] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3204–3215, 2021

[6] 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