



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공학석사학위논문

부분 관측된 점 구름으로부터의 로봇 파지:
변형 가능한 모델과 딥러닝의 병합

**Robot Grasping from Partial Point Cloud Data:
Merging Deformable Models with Deep Learning**

2021년 2월

서울대학교 대학원

기계공학부

안 태 군

부분 관측된 점 구름으로부터의 로봇 파지:
변형 가능한 모델과 딥러닝의 병합

**Robot Grasping from Partial Point Cloud Data:
Merging Deformable Models with Deep Learning**

지도교수 박종우

이 논문을 공학석사 학위논문으로 제출함

2021년 2월


서울대학교 대학원

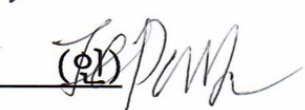
기계공학부


안태균

안태균의 공학석사 학위논문을 인준함

2021년 2월

위원장 : 이경수 (인) 

부위원장 : 박종우 (인) 

위원 : 이동준 (인) 

ABSTRACT

Robot Grasping from Partial Point Cloud Data: Merging Deformable Models with Deep Learning

by

Taegyun Ahn

Department of Mechanical Engineering

Seoul National University

Grasping novel and complex objects with partial observation is a challenging task for robots. One solution for this problem is through full shape estimation. Previous works estimated the full geometry by optimization of fitting loss between observed partial point cloud to a set of shape primitive surfaces. However, these optimization-based fitting cannot constrain the shape of the unobserved region, where points do not exist. Thus, they show disappointing performance of full shape estimation.

In order to solve such issue, we propose a novel supervised learning framework for full geometry estimation of partially observed objects. Also, we propose deformable-superquadrics, which can represent various shapes in continuous parameter space, as shape primitives to estimate the full geometry of complex objects. We show that our new learning framework well estimates the full geometry of complex household objects. By sampling antipodal points on the estimated surface, we could successfully find grasp poses of the robot gripper.

Keywords: grasp pose generation, partially observed point cloud, full shape estimation, supervised learning, deformable-superquadrics

Student Number: 2019-23899

Contents

Abstract	i
List of Tables	vi
List of Figures	vii
1 Introduction	1
1.1 Related Works	4
1.2 Contributions of Our Work	6
1.3 Organization	7
2 Preliminaries	8
2.1 Superquadrics	9
2.1.1 Definition of Superquadrics	9
2.1.2 Radial Distance of Superquadrics	10
2.1.3 Error Metric for Superquadric Fitting	11
2.2 Deformable-Superquadrics	13
2.2.1 Tapering Deformation	13

2.2.2	Bending Deformation	14
2.2.3	Combined Deformation	16
2.3	Deep Learning Networks for Point Cloud	18
2.3.1	Permutation Invariance	18
2.3.2	Convolution on Point Cloud Data	19
3	Grasping Pipeline	20
3.1	Recognition Module	20
3.1.1	Shape Primitives Segmentation	22
3.1.2	Deformable-Superquadric Fitting	23
3.2	Grasping Module	26
3.2.1	Antipodal Grasp Pose Sampling	26
3.2.2	Collision and Kinematic Feasibility Check	28
3.2.3	Grasp Pose Selection	29
4	Experiments	30
4.1	Evaluation Metric	30
4.2	Optimization vs Learning	31
4.3	Recognition Module	34
4.3.1	Dataset	34
4.3.2	Segmentation	36
4.3.3	Fitting	36
4.4	Grasping Results	38
5	Conclusion	43
	Bibliography	45

List of Tables

4.1	Fitting of Partially Observed Point Cloud	33
4.2	Full Shape Estimation of Objects	38
4.3	Grasping Performance	42

List of Figures

1.1	Optimization solution of partial point cloud fitting can have several solutions.	3
1.2	End-to-end grasping approach vs shape primitive based two-step approach.	4
2.1	Various shapes of superquadrics when $a_1 = a_2 = a_3 = 1$	9
2.2	Euclidean distance and radial distance.	11
2.3	Bending plane of bending deformation.	14
2.4	Examples of deformed superquadrics.	16
3.1	Entire grasping pipeline.	21
3.2	Supervised Deformable-Superquadric Fitting Network.	24
3.3	Grasping module.	26
3.4	Antipodal points of deformable-superquadrics.	28
4.1	Comparison of optimization and learning based fitting performance.	32
4.2	Dataset used for training our networks and grasping experiment.	35
4.3	Results of full shape estimation of various synthetic objects.	39

4.4 Grasping Result.	41
------------------------------	----

1

Introduction

Robot grasping problem is a grasp pose $g \in SE(3)$ generation problem given observations of an object such as RGB or depth images. Robot grasping is important in the manufacturing and logistics industries for automation. Especially in logistics, the robot has to quickly grasp a variety of unseen objects with limited observations. Thus, grasping unseen objects from partial observation is a key task.

Due to the limitation of classical analytic methods on grasping novel objects, deep learning-based grasping methods have recently been widely studied [1]. One popular approach is end-to-end grasping approach, where a grasp pose is directly learned from an object image, uses millions of grasp data to generalize grasp strategy from experience. However, acquiring grasp label data (pair of a grasp image with success/fail labeled grasp pose) is very expensive. Also, end-to-end grasping approach requires totally new data and re-training for the different gripper.

Another method to grasp a partially observed object is two-step approach [2, 3, 4, 5]: 1) given a partially observed point cloud, estimate the full geometry of an object, then 2) find a grasp pose of a gripper. This method relatively suffers less

from data collection than the end-to-end approach, since the collection of visual data is usually easier compared to the grasp data. Also, since the estimation of the full geometry does not involve grasping, the estimation is gripper independent. When the full geometry is estimated, the grasp pose generation problem becomes much easier because grasp points can be sampled from the full surface and analytic grasp pose evaluations are possible.

However, existing two-step grasping approaches have 3 major issues. 1) Primitives used in existing approaches are too simple and simply replacing with more complex primitives doesn't work because algorithms are designed in a primitive-dependent way. 2) Previous methods require optimization as an intermediate step, which is slow. 3) These optimization-based methods don't work well for partial point cloud data: specifically, reconstructed objects in many cases don't look at all like the actual object. The majority of existing approaches used bounding box, cylinder, and sphere to approximate the geometry of the object, thus they cannot express complex objects such as bottles or mugs. Also, most used optimization algorithms such as RANSAC or Iterative Closest Points (ICP) to match shape primitive surfaces with observed point clouds. However, such optimization algorithms require a lot of computation time and are also prone to find sub-optimal solutions. Also, even if a globally minimum solution is found (as in Figure 1.1), the estimated geometry in the unobserved region can be far from the true shape. This is due to the ill-posed nature of the original problem; partial point cloud can be part of several different geometries. For example, a rectangle plane can be fitted to a thin box or a thick box and both can have a small fitting loss.

In this paper, we propose a two-step grasping method designed for partial point cloud data. Specifically, 1) we propose a richer set of primitives: deformable-superquadrics that achieves a good balance between computational simplicity and

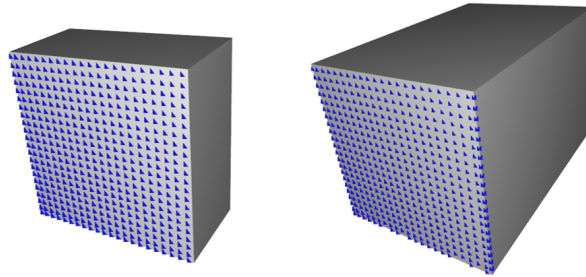


Figure 1.1: Optimization solution of partial point cloud fitting can have several solutions.

expressiveness, using 7 continuous parameters. Also, an analytical closed-form equation of deformable-superquadrics gives advantages for loss function evaluation in network training. 2) We propose a Supervised Deformable-Superquadric Fitting Network (SDSFN) for fast full shape estimation. 3) We created synthetic data to augment data: our data consists of partially observed point clouds and the full point clouds of synthetic objects. This novel data is required to train SDSFN since it uses the full geometry of objects as supervision to estimate the unobserved regions of objects. 4) In practice, actual implementation requires several methods and choices of parameters, which are not trivial. We show in detail how to integrate these methods and choose parameters in a compatible way. 5) Finally, we demonstrate that our method enables robots to grasp household objects only with partial observation.

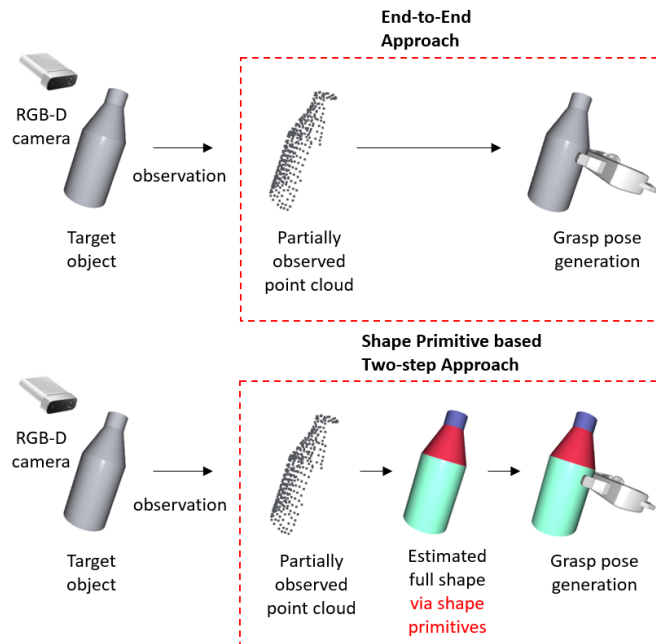


Figure 1.2: End-to-end grasping approach vs shape primitive based two-step approach.

1.1 Related Works

Classical analytic grasping methods calculate grasp quality measures such as force closure to evaluate grasp poses [6, 7, 8]. However, these methods require full knowledge on geometric and physical models of the objects [1], thus cannot grasp partially observed, novel objects. Recently, deep learning-based methods are studied to generalize grasping skills for novel objects. Many, including Dex-net 2.0 [9] and 6dof-graspnet [10] used end-to-end approach, where a grasp pose is learned directly from raw input such as an image of the object. Dex-net learns robust epsilon quality of the grasp poses directly from a depth image. 6dof-graspnet uses Variational

Auto Encoder to learn the latent space of successful grasp poses. However, these methods require a huge amount of labeled data, which is not trivial to obtain. Levine used 14 robots and spent two months collecting 800,000 grasp data [11]. Dex-net and 6dof-graspnet increased the data scale to 6.7 million and 7 million grasp data, and recently [12] published 1 billion grasp data for grasp learning. Another drawback of end-to-end approach is that the learned grasp policy is only compatible with a single gripper. When the gripper is changed, the success/fail labels of the original data needs to be changed. Thus, the recollection of the grasp data with the new gripper and retraining of the network is required to apply this method to another gripper. However, the two-step grasp learning approach, where full geometry of the object is learned first and a grasp pose is generated from the knowledge of full geometry, does not suffer from these issues.

Shape primitives fitting has a long history in various fields due to their computational simplicity and memory efficiency [13]. In robotics, [2, 3] used minimum volume bounding box to approximate fully observed point clouds with a set of boxes and grasped objects by grasping one of the boxes. Efficient RANSAC [14] algorithms is another popular method to detect shape primitives. Few points are randomly sampling from the point cloud and the primitive surface that passes them is determined. A primitive surface with most point inliers that fit the surface is selected. [4] used this algorithm to detect cylinders and spheres in the partially observed point clouds of objects and grasped them. [5] extended shape primitives to cylinder, ring, cuboid, stick, semi-sphere, sphere to express household objects. They used Mask R-CNN to segment objects into shape primitives and used ICP algorithm to register segmented point clouds to one of the primitives in the primitive shape DB. However, the performance of ICP algorithm degrades when registering partial point clouds. Also, in order to estimate the parameters of

their shape primitives, they matched with multiple primitive shapes with discrete parameter values. Not only this brute-force method is inefficient, but also their expressive power of shape primitives. On the contrary, our method uses deep learning to avoid local minima of shape primitive fitting and can generalize the shapes of the unobserved region. In addition, we can find shape parameters in continuous space with single network inference. Also, our deformable-superquadric primitives can represent more variety of geometries than previous works.

[15, 16, 17] used superquadrics to estimate the full geometry and proposed grasping pipelines for grasping superquadrics. However, they used optimization algorithms that are slow and have limitations when estimating the geometry of the unobserved region. Unlike their work, our network can estimate the full geometry faster with higher accuracy. Also, we expanded superquadrics to deformable-superquadrics to increase the expressive power.

1.2 Contributions of Our Work

The main contribution of our work can be summarized into four. 1) We propose deformable-superquadrics as a new shape primitive to reconstruct complex objects that include cones and rings. 2) We propose a new learning framework for estimating the full geometries of partially observed point clouds. 3) We created a novel synthetic data set that can give our network supervision of full geometry. 4) Lastly, we incorporated the above to create a full grasping algorithm that can grasp unseen objects, even with partial observation.

Through experiments, we show that our method out-performs at estimating the full geometry of partially observed objects compared to the classical optimization-based methods. Also, we show that the computation time of SDSFN required for

the full shape estimation is much shorter than previous works. We demonstrate that our method can effectively generate grasp poses for partial point clouds by showing grasping success rate similar to the previous works.

1.3 Organization

In chapter 2, we review the concept of deformable-superquadrics. Here we explain superquadrics that can represent various geometries with a single equation. We extend superquadrics to deformable-superquadrics by tapering and bending original superquadrics to express cones and rings that cannot be expressed by original superquadrics. The radial distance used for fitting superquadrics with point clouds is also introduced. We also, review deep learning network architectures that are specified for point cloud inputs. We explain the concept of permutation invariance and graph convolution that are required for point cloud network.

In chapter 3, we introduce our two-step grasping approach. We use deep learning to segment partially observed object point cloud to shape primitives and fit them to deformable-superquadrics. We used antipodal sampling method to select a grasp pose from the estimated full 3d shape of the object.

In chapter 4, we validate our claim that optimization-based shape primitive fitting algorithms cannot estimate the unobserved region of the objects, thus a learning-based primitive fitting method should be used. Then, we evaluate our performance of segmentation and fitting networks that are used for estimating full geometry from partial observation. We also evaluated the grasping performance of our entire framework and shows that our methods achieve a similar performance of grasping compared to the state-of-the-art methods.

We conclude our thesis by summarizing our methods and results in chapter 5.

2

Preliminaries

In this chapter, we review the concept of superquadrics and deep learning networks specified for point cloud inputs. In section 2.1 we explain the definition of superquadrics. Also, we explain the radial distance between a point and superquadric surface that is used in our loss function of the fitting network. In section 2.2 We repeat these explanations on deformable-superquadrics that are obtained by tapering and bending superquadrics. In section 2.3 we introduce operations required for a point cloud-specific network. We explain the concept of point permutation invariance a deep learning network should satisfy. Also, we explain how a convolution operation can be done on point cloud data.

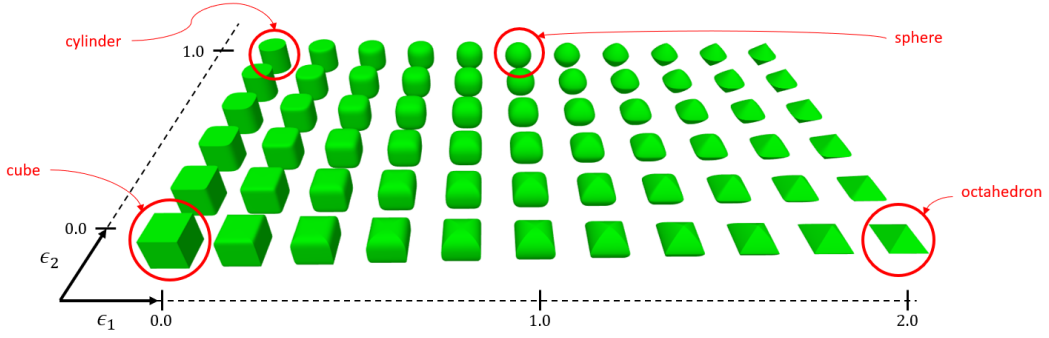


Figure 2.1: Various shapes of superquadrics when $a_1 = a_2 = a_3 = 1$.

2.1 Superquadrics

2.1.1 Definition of Superquadrics

Superquadrics are generalized quadrics that can represent various shape. It includes superellipsoids, supertoroids, superhyperboloids but in many works, superellipsoids commonly referred as superquadrics, thus we also use term superquadrics as superellipsoid from now on. Superquadrics are expressed using following equation,

$$F(x, y, z) = \left(\left| \frac{x}{a_1} \right|^{\frac{2}{\epsilon_2}} + \left| \frac{y}{a_2} \right|^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left| \frac{z}{a_3} \right|^{\frac{2}{\epsilon_1}} = 1. \quad (2.1.1)$$

$\epsilon_1, \epsilon_2 \in \mathbb{R}$ in the exponent term of the equation are called shape parameters. Varying these parameters, superquadrics can become box, sphere, cylinder and octahedrons. $a_1, a_2, a_3 \in \mathbb{R}$ at the denominator terms of the equations accounts for the size of the superquadrics, thus are called size parameters. Thus, superquadrics can express variety shape and size of objects with single equation in continuous parameter space. Figure 2.1 shows variety of shapes superquadrics can represent when $a_1 = a_2 = a_3 = 1$. Note that shape parameters ϵ_1, ϵ_2 are outside $(0, 2)$,

the equation 2.1.1 and its gradient becomes numerically unstable. Thus, shape parameters of superquadrics are bounded within 0 and 2 in most practical applications [18].

Similar to parametric representation of a sphere, superquadrics can also be expressed in explicit form,

$$\mathbf{r}(\eta, \omega) = \begin{bmatrix} a_1 \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega \\ a_2 \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega \\ a_3 \sin^{\epsilon_1} \eta \end{bmatrix}, \quad \begin{array}{l} -\frac{\pi}{2} \leq \eta \leq \frac{\pi}{2} \\ -\pi \leq \omega < \pi. \end{array} \quad (2.1.2)$$

Note that exponential ϵ of $f(x)$ is a signed power function

$$f^\epsilon(x) = \text{sign}(f(x))|f(x)|^\epsilon. \quad (2.1.3)$$

Normal vector of superquadric surface at a point $\mathbf{r}(\eta, \omega)$ can be calculated with a cross product of the tangent vectors along η and ω ,

$$\mathbf{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \frac{\partial \mathbf{r}}{\partial \eta} \times \frac{\partial \mathbf{r}}{\partial \omega} = \begin{bmatrix} \frac{1}{a_1} \cos^{2-\epsilon_1}(\eta) \cos^{2-\epsilon_2}(\omega) \\ \frac{1}{a_2} \cos^{2-\epsilon_1}(\eta) \sin^{2-\epsilon_2}(\omega) \\ \frac{1}{a_3} \sin^{2-\epsilon_1}(\eta) \end{bmatrix}. \quad (2.1.4)$$

2.1.2 Radial Distance of Superquadrics

It is hard to calculate Euclidean distance between a point and the surface of a superquadric due to the lack of closed-form solution. Instead, radial distance δ between a point $\mathbf{x}_0 \in \mathbb{R}^3$ and the superquadrics surface $F(\mathbf{x}) = 1$ can be easily computed. The radial distance between a point and the superquadric surface is the distance between the point and the intersection point of superquadric surface and the line that connects the point and the center of the superquadric. The intersection point between the superquadric surface and the line connecting its center

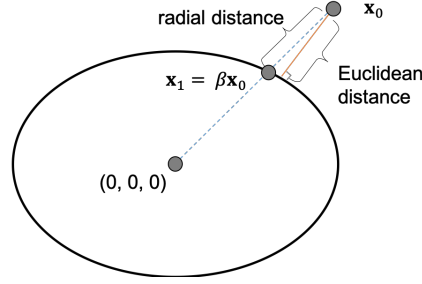


Figure 2.2: Euclidean distance and radial distance.

and point $\mathbf{x}_0 = (x_0, y_0, z_0)$ will be $\mathbf{x}_1 = \beta\mathbf{x}_0$. Then, the radial distance can be,

$$\|(1 - \beta)\mathbf{x}_0\|. \quad (2.1.5)$$

Since point \mathbf{x}_1 lies on the surface of the superquadrics, it has to satisfy $F(\mathbf{x}_1) = 1$.

$$F(\mathbf{x}_1) = \left\{ \left\{ \left(\frac{\beta x_0}{a_1} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{\beta y_0}{a_2} \right)^{\frac{2}{\epsilon_2}} \right\}^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{\beta z_0}{a_3} \right)^{\frac{2}{\epsilon_1}} \right\} = \beta^{\frac{2}{\epsilon_2}} F(\mathbf{x}_0). \quad (2.1.6)$$

Thus, radial distance δ between point \mathbf{x}_0 and the superquadric $F(\mathbf{x}) = 1$ can be calculated with a close form solution,

$$\delta = \|\mathbf{x}_0\| |1 - F^{-\frac{\epsilon_1}{2}}(\mathbf{x}_0)|. \quad (2.1.7)$$

2.1.3 Error Metric for Superquadric Fitting

Given a point cloud, superquadric fitting is the process of finding a superquadric parameter set $\{a_1, a_2, a_3, \epsilon_1, \epsilon_2\}$ that best fits it. Two error metrics are commonly used for superquadric fitting. [19] directly applied the surface function $F(x, y, z) = 1$ to a data point $\mathbf{x}_0 = (x_0, y_0, z_0) \in \mathbb{R}^3$ to calculate surface fitting loss,

$$L = \sqrt{a_1 a_2 a_3} |1 - F^{\epsilon_1}(\mathbf{x}_0)|. \quad (2.1.8)$$

$\sqrt{a_1 a_2 a_3}$ is multiplied to prevent local minimum by fitting superquadrics with larger volumes, and F is powered to ϵ_1 to avoid numerical instability and remove

bias towards fitting superquadrics with larger values of ϵ_1 . This error metric has several issues [20]. The equation is equal to

$$L = \sqrt{a_1 a_2 a_3} \left| 2 \frac{\delta}{\|\mathbf{x}_1\|} + \frac{\delta^2}{\|\mathbf{x}_1\|^2} \right|, \quad (2.1.9)$$

where δ in this equation omit absolute operation in $|1 - F^{-\frac{\epsilon_1}{2}}(\mathbf{x}_0)|$. Thus, points inside the superquadric surface has lower error than points outside the surface with same $|\delta|, \|\mathbf{x}_1\|$. Also, the error term $\frac{\delta}{\|\mathbf{x}_1\|}$ decreases as $\|\mathbf{x}_1\|$ increases, which gives the error metric tendency to grow the volume of the superquadrics. The multiplication of $\sqrt{a_1 a_2 a_3}$ to prevent this issue is only empirically proven to work.

[21] used the radial distance δ of equation 2.1.7 for superquadric fitting. We tested both error metric for the fitting loss in our training network and found that radial distance based fitting loss showed better performance.

2.2 Deformable-Superquadrics

Although superquadrics can express a variety of shapes, it cannot express shapes such as cones and rings. The range of shapes expressed by superquadrics is expanded by applying global deformations to it. Here we, introduce two popular deformations, tapering and bending. We use the term deformable-superquadrics to represent superquadrics and their deformed forms by these two deformations.

2.2.1 Tapering Deformation

where points (x, y, z) on the surface of original superquadrics, (X, Y, Z) are the deformed points, and $t_k(z)$ is the tapering function. For linear tapering,

$$\begin{aligned} X &= t_k(z)x \\ Y &= t_k(z)y \\ Z &= z, \end{aligned} \tag{2.2.10}$$

$$t_k(z) = \frac{k}{a_3}z + 1, \tag{2.2.11}$$

is used, where $-1 \leq k \leq 1$ is a tapering parameter. When $k = 0$, deformation is not applied, thus the deformable-superquadrics and superquadrics are identical. This tapering can make pointy shapes from original superquadrics. For example, when tapering is applied to a cylinder with $k = 1$, the deformed superquadric becomes a cone. For fitting deformable-superquadrics to point clouds, a surface equation expressed in X, Y, Z is required. Thus we need an inverse transformation of tapering D_t^{-1} , which is given by the following,

$$\begin{aligned} x &= \frac{a_3}{kZ+a_3}X \\ y &= \frac{a_3}{kZ+a_3}Y \\ z &= Z, \end{aligned} \tag{2.2.12}$$

2.2.2 Bending Deformation

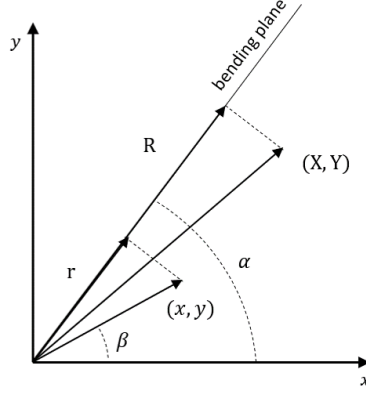


Figure 2.3: Bending plane of bending deformation.

Bending deformation bends the z -axis of the superquadric to a circular section while preserving its length. The direction of the bending is described the bending plane. The bending plane includes the original z -axis and the bent circular section, and the angle between the bending plane and the x - z plane is α .

The points of the bent superquadrics can be calculated by first projecting original points to the bending plane, performing bending on projected points and projecting bent points back to the original plane. The projection of original points to the bending plane can be calculated as follows,

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \cos(\alpha - \beta), \\ \beta &= \arctan \frac{y}{x}. \end{aligned} \quad (2.2.13)$$

The projected points r are bent to

$$R = \frac{1}{b} - \left(\frac{1}{b} - r\right) \cos \gamma \quad (2.2.14)$$

where $\gamma = zb$. By re-projecting bent points to original planes, superquadrics with bending deformation can be obtained,

$$\begin{aligned} X &= x + (R - r) \cos \alpha \\ Y &= y + (R - r) \sin \alpha \\ Z &= \left(\frac{1}{b} - r\right) \sin \gamma. \end{aligned} \tag{2.2.15}$$

In our case, we use bending deformation when $\alpha = 0$. Thus, the bending deformation D_b along x-axis is

$$\begin{aligned} X &= \frac{1}{b} - \cos(\gamma)\left(\frac{1}{b} - x\right) \\ Y &= y \\ Z &= \sin(\gamma)\left(\frac{1}{b} - x\right), \end{aligned} \tag{2.2.16}$$

where $b > 0$ is a bending parameter and γ is the corresponding bending angle. As in the case of tapering, deformable-superquadrics becomes superquadrics when b is close to 0. Using bending deformations, ring shapes can be obtained from cylinders and boxes. The inverse of bending deformation D_b^{-1} can be obtained with,

$$\begin{aligned} x &= \frac{1}{b} - \sqrt{Z^2 + \left(\frac{1}{b} - X\right)^2} \\ y &= Y \\ z &= \frac{\gamma'}{b}, \end{aligned} \tag{2.2.17}$$

where $\gamma' = \text{atan2}(Z, \frac{1}{b} - X)$. Examples of deformable-superquadrics can be seen in Figure 2.4

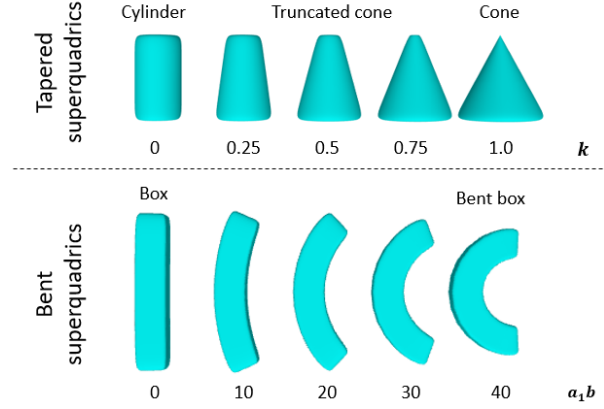


Figure 2.4: Examples of deformed superquadrics.

2.2.3 Combined Deformation

When combining tapering and bending deformation, the order of deformation should be carefully selected. The commutation of deformations results in different final shape *i.e.*, $D_b \circ D_t \neq D_t \circ D_b$. In this work, we deform our superquadrics by tapering first and bending. Thus the deformation and inverse of Deformation is obtained by,

$$(X, Y, Z) = D_b \circ D_t(x, y, z), \quad (2.2.18)$$

$$(x, y, z) = D_t^{-1} \circ D_b^{-1}(X, Y, Z). \quad (2.2.19)$$

Our deformable-superquadrics have total 7 parameters; $a_1, a_2, a_3, \epsilon_1, \epsilon_2, k, b$. In such case, normal vectors of deformed superquadrics can be obtained from following,

$$J_t(\mathbf{x}) = \begin{bmatrix} \frac{k}{a_3}z + 1 & 0 & \frac{k}{a_3}x \\ 0 & \frac{k}{a_3}z + 1 & \frac{k}{a_3}y \\ 0 & 0 & 1 \end{bmatrix}, \quad (2.2.20)$$

$$J_b(\mathbf{x}) = \begin{bmatrix} \cos(\gamma) & 0 & \sin(\gamma)(1 - bx) \\ 0 & 1 & 0 \\ -\sin(\gamma) & 0 & \cos(\gamma)(1 - bx) \end{bmatrix}, \quad (2.2.21)$$

$$\tilde{\mathbf{n}} = \det JJ^{-T} \mathbf{n}. \quad (2.2.22)$$

\mathbf{n} is the normal vector of original superquadric surface from equation 2.1.4 and $J = J_t(\mathbf{x})$ for tapering deformation and $J = J_b(D_t(\mathbf{x}))J_t(\mathbf{x})$ for combined deformation .

In the case of bending deformation, radial distance can be calculated with the same equation 2.1.7 of superquadrics. However, this is not true for tapering, but nevertheless, the error is marginal and can be approximated as true value [22].

2.3 Deep Learning Networks for Point Cloud

Throughout the past decade, deep learning showed remarkable achievements in computer vision, with their powerful convolution operations on image data. However, the same architectures cannot be directly applied to deep learning on point cloud data, due to the data difference. A major difference between an image and a point cloud is regularity. An image has a regular grid structure, where pixel indices of the x, y-axis indicate the location of the pixel. Also, each pixel in the images (except for the ones on the edges and corners of images) has exactly 8 neighbor pixels, which makes the implementation of convolution operation easier. However, point clouds have irregular and unordered form. A point cloud is represented by a set of points. Thus the order of the points in the set does not change the geometry of the point cloud. Also, explicit neighbor points of a point are not defined as in the case of an image, thus original convolution operation on the regular grid cannot be directly used. Thus we review specialized network architectures for point cloud data to solve such issues.

2.3.1 Permutation Invariance

A point cloud is represented with a set of points $\mathcal{P} = \{p_1, \dots, p_n\}, p_i \in \mathbb{R}^3$, but in practice matrix $\mathbf{P} \in \mathbb{R}^{n \times 3}$ is commonly used instead. Instead, for arbitrary permutation matrix $\pi \in \mathbb{R}^{n \times n}$, $\pi\mathbf{P}$ represents same point cloud. However, for arbitrary function f whose inputs are point clouds \mathbf{P} , the outputs of permuted inputs are different *i.e.*, $f(\mathbf{P}) \neq f(\pi\mathbf{P})$. However, since \mathbf{P} and $\pi\mathbf{P}$ originally represent same geometry, we wish the function to output same value. This property of function f is called permutation invariant,

$$f(\mathbf{P}) = f(\pi\mathbf{P}). \quad (2.3.23)$$

For networks whose inputs are n points of point clouds, they have to satisfy this permutation invariance for $n!$ permutation of point orders. PointNet [23] used max pooling at the end of the network to extract a permutation invariant global feature. Other symmetric aggregation operation such as mean pooling can be also used.

2.3.2 Convolution on Point Cloud Data

Wang et al. introduced Edge Convolution to implement convolution-like operation on point cloud data [24]. For finding neighbor points, k -nearest neighbor graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed. Edge features are defined as $e_{ij} = h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j)$, where $h_{\Theta} : \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}^{F'}$ is a learnable edge function with parameters Θ . Edge convolution is applied to the graph by applying symmetric aggregation operation \square on all edge features connected to \mathbf{x}_i .

$$\mathbf{x}'_i = \square_{j:(i,j) \in \mathcal{E}} h_{\Theta}(\mathbf{x}_i, \mathbf{x}_j). \quad (2.3.24)$$

Compared to convolution operation on image, \mathbf{x}_i and $\{\mathbf{x}_j | (i, j) \in \mathcal{E}\}$ can be thought as central pixel and patch around it in 2d convolution. In fact, when $\square = \sum$ and $h_{\Theta} = \theta \cdot \mathbf{x}_j$, the equation 2.3.24 becomes identical to the standard convolution on images. Wang presented several choice of edge function and aggregation operation. Among them, they used following MLP as edge function with max pooling operator.

$$\mathbf{x}'_i = \max_{j:(i,j) \in \mathcal{E}} \text{ReLU}(\theta_m \cdot (\mathbf{x}_j - \mathbf{x}_i) + \phi_m \cdot \mathbf{x}). \quad (2.3.25)$$

3

Grasping Pipeline

Our grasping pipeline is divided into two stages. First, the recognition module estimates the full 3d shape of an object from a partially observed point cloud. Then, the grasping module generates a grasp pose for the gripper. Figure 3.4 shows the entire process of our grasping algorithm.

3.1 Recognition Module

The recognition module approximates the full 3d shape of the object using a set of deformable-superquadrics. The partially observed point cloud of an object first passes through a segmentation network and decomposed into shape primitives. Then, the primitive fitting network (SDSFN) estimates the full geometry of each segmented point cloud by finding the parameters and $SE(3)$ of the deformable-superquadrics that best fits it. Assembling all deformable-superquadrics, we can get a full shape estimation of the partially observed object.

In order to recognize the full shape of the object from arbitrary angles, we

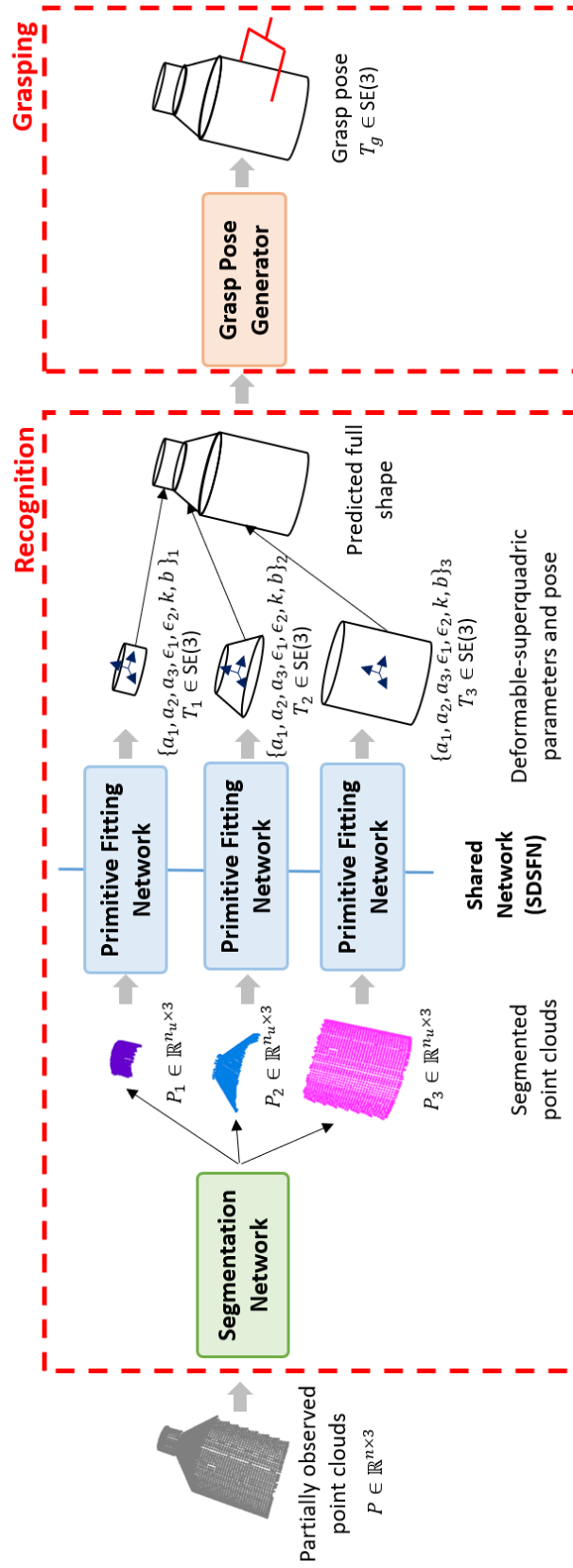


Figure 3.1: Entire grasping pipeline.

trained our segmentation and fitting networks with data that contains 16 view directions of the same objects.

A point cloud of an object can be represented with coordinates of different frames. However, learning segmentation and fitting of point clouds with arbitrary frame is difficult and requires lots of data. Thus, we pre-processed our point cloud data to have a consistent point cloud frame. We transformed point cloud to the point cloud frame, whose center is the center of the point cloud and axes are principal axes calculated from principal component analysis (PCA). Eigenvectors with largest and smallest eigenvalues are chosen as z-axis and y-axis respectively. However, eigenvector found by PCA can be one of two directions $e_i, -e_i$. Thus to reduce this stochasticity, we further rotated point cloud frame with respect to x and z axes so that the thicker part of the point cloud always lies on the -z coordinate. The thickness of the point cloud is measured by calculating the mean of x, y norm of points. If the mean norm of the points with positive z values is greater than the value of the points with negative z values, the point cloud frame is rotated π around the x-axis. The same procedure was done for the x-axis of the point cloud frame.

3.1.1 Shape Primitives Segmentation

The segmentation network split partially observed point cloud input into shape primitives. In order to train the segmentation network, we created a synthetic data set that has a ground-truth segmentation label. Our data set is consisted of partially observed point clouds of synthetic household objects which are made using box, cylinder, sphere, cone, truncated cone, and sliced torus. The network is trained using cross-entropy loss. A Point-wise one-hot vector of segmentation label is given to the network as ground-truth. Thus the network predicts confidence

of m segment label (membership) of each point. Since permutation of segmentation label is unimportant, we used Hungarian matching [25] to reorder prediction segmentation label that has a maximum match with the ground-truth label.

3.1.2 Deformable-Superquadric Fitting

After the object point cloud is segmented into shape primitives via the segmentation network, the primitive fitting network (SDSFN) fits deformable-superquadrics to the segmented partial point clouds. The SDSFN predicts 7 parameters (2 shape, 3 size, 2 deformation parameters) of deformable-superquadrics and their poses $T \in SE(3)$.

To estimate the full 3d shape of the partial point cloud, we gave our network supervision of the full 3d shape with full point clouds of objects. We minimize radial distance δ of equation 2.1.7 between the predicted deformable-superquadric surface F and the full points $G = \{g_1, \dots, g_{n_g}\} \in \mathbb{R}^{n_g \times 3}$ uniformly sampled from the ground-truth primitives.

In order to estimate the pose of the deformable-superquadric, the ground-truth points g_k are transformed to superquadric frame by multiplying the inverse of its pose T to g_k .

We trained our SDSFN network with 6 shape primitives where each primitive is defined in a canonical frame. However, pre-processing of point cloud with the transformation to PCA frame, makes the distribution of the z-axis of those primitives very discrete. For example, point clouds of elongated cylinders, whose heights are greater than diameters, have the z-axis of the PCA frame in the height direction. On the contrary, flat cylinders have z-axis of point cloud frame in radius directions. This hinders point cloud fitting to the superquadric surface since x and y in superquadric equation 2.1.1 are commutative but not with z . When the z-axis

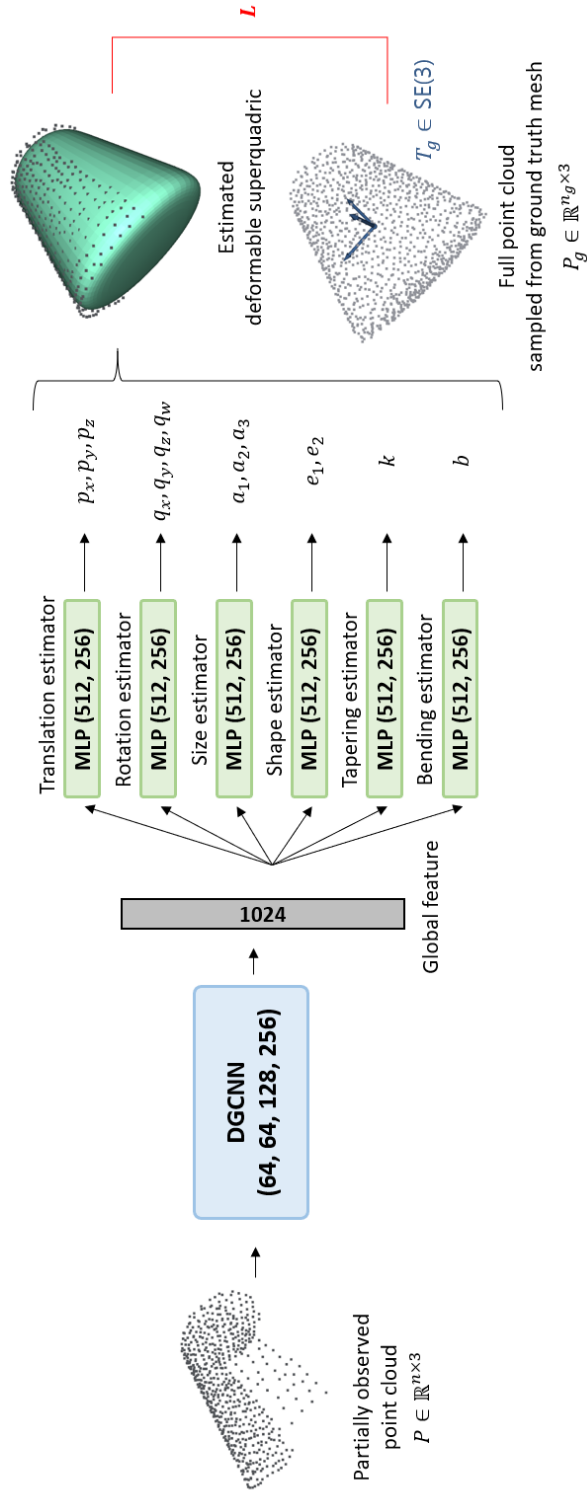


Figure 3.2: Supervised Deformable-Superquadric Fitting Network.

of the superquadric is mismatched, the performance of the superquadric fitting degrades. This is especially true for our case since the training data set has a biased distribution of z-axis, thus tries to fit superquadrics with certain z-axis. Therefore, we added supervision of the z-axis of shape primitives. Thus, the loss function of the primitive fitting network becomes as follow,

$$L_{superquadrics} = \frac{1}{n_g} \sum_{k=1}^{n_g} \delta^2(T^{-1}g_k) + w \|z \times z_g\|^2, \quad (3.1.1)$$

where w is a hyperparameter that regulates two loss terms.

By giving supervision of the full geometry, our network can well generalize the shapes of the unobserved objects. Note that we give point cloud, not parameters of deformable-superquadrics as supervision data. Thus we do not need to know the exact parameters of the object to learn the full geometry. Instead, our network figures them out for us.

3.2 Grasping Module

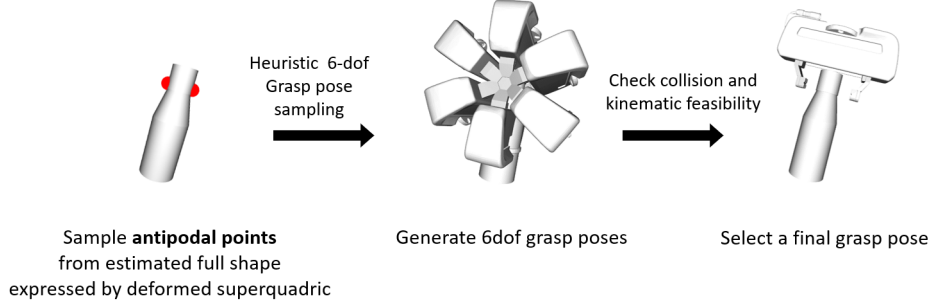


Figure 3.3: Grasping module.

Once a partially observed point cloud of an object is fitted to a set of deformable-superquadrics, the grasping module generates a grasp pose using antipodal sampling-based algorithm. Antipodal points are two points on the surface that are located on the opposite parts of an object, and frequently used as candidate points for grasping points of parallel two-finger grippers [26, 9, 17]. The entire grasping module consists of 3 parts; 1) grasp pose sampling via antipodal sampling, 2) collision and kinematic feasibility check, and 3) grasp pose selection.

3.2.1 Antipodal Grasp Pose Sampling

First, we sampled points uniformly on the surface of the predicted full 3d mesh of an object. Then, antipodal point of each sampled point was obtained by finding intersection points between the object mesh and a line that passes the sampled point and is normal to the surface at that point. Normals are calculated with equation 2.1.4. We chose the farthest two intersection points as two antipodal points.

Finding intersection point with line $\mathbf{l}(t) = \mathbf{p} + \mathbf{d} \cdot t$, $t \in \mathbb{R}$ superquadrics are not easy, since we cannot get closed-form solution. Thus, the intersection point should

be found using an iterative solver. We used Newton-Raphson based method used in [27] for finding line intersection points. Newton-Raphson method finds the solution of $f(t) = 0$ in iterative manner,

$$t_{new} = t_{old} - \frac{f(t_{old})}{f'(t_{old})}. \quad (3.2.2)$$

For initial guess of t , we used intersection points of bounding sphere whose radius is $r = \max(a_1, a_2, a_3)$. The intersection point of the bounding sphere can be easily found by solving quadratic equation $at^2 + bt + c = 0$, where $a = \mathbf{d} \cdot \mathbf{d}$, $b = \mathbf{d} \cdot \mathbf{p}$, and $c = \mathbf{p} \cdot \mathbf{p} - r^2$. Since \mathbf{p} of the line $\mathbf{l}(t)$ is the point on the superquadric, we know that $t = 0$ is a solution. For antipodal searching problem, solution $t \neq 0$ is our interest. Thus we choose solution t with greater absolute value as our initial guess t_{init} of the Newton-Raphson method. For line-superquadric intersection the $f(t)$ of equation 3.2.2 becomes,

$$f(t) = S(\mathbf{l}(t)), \quad (3.2.3)$$

$$f'(t) = \mathbf{d} \cdot \nabla S(\mathbf{l}(t)), \quad (3.2.4)$$

where $S(\mathbf{x}) = F(\mathbf{x}) - 1$, $\mathbf{x} \in \mathbb{R}^3$.

Then, normals at the two antipodal points are compared and antipodal points are kept, if the angle between the two vectors is smaller than 15 degrees.

After the antipodal points are sampled, 6-dof grasp poses are generated heuristically. The Z-axis of the gripper that directs the center of the object is selected as an initial candidate. The Y-axis of the gripper is aligned with a line that connects two antipodal points. Grasp pose is rotated 30 degrees along the y-axis to generate 12 6-dof grasp poses for each antipodal points pair.

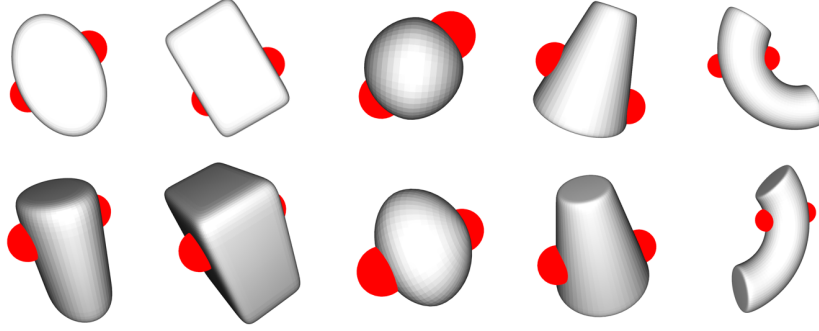


Figure 3.4: Antipodal points of deformable-superquadrics.

3.2.2 Collision and Kinematic Feasibility Check

Collisions of the gripper with surroundings are checked using Flexible Collision Library [28]. Candidate grasp pose that does not collide with both object and table is rejected.

Kinematic feasibility of the grasp pose is checked by finding the inverse kinematics solution of the robot. Given a grasp pose $T_g \in SE(3)$ Solution of the inverse kinematics of a candidate grasp pose was found using Newton-Raphson iterative algorithm [29],

$$\theta_{i+1} = \theta_i + J_b^\dagger(\theta_i)\mathcal{V}_b. \quad (3.2.5)$$

$J_b^\dagger \in \mathbb{R}^{6 \times n}$ is pseudo body Jacobian of the robot, $\mathcal{V}_b \in \mathbb{R}^6$ is the body twist calculated from,

$$[\mathcal{V}_b] = \log(T_{sb}^{-1}(\theta_i)T_g). \quad (3.2.6)$$

If the candidate grasp pose is found to be unreachable, the grasp pose is excluded from the candidate.

3.2.3 Grasp Pose Selection

A final grasp pose is selected among grasp pose candidates that passed collision and kinematic feasibility check, by favoring a top-down grasping.

4

Experiments

4.1 Evaluation Metric

The main contribution of our work is the recognition module, where we estimate the full 3d geometry of the partially observed point cloud. To evaluate our recognition module, we used the intersection over union(IoU) metric which is defined by,

$$IoU = \frac{|A \cap B|}{|A \cup B|}. \quad (4.1.1)$$

Our recognition module outputs a full 3d mesh of an observed object. To measure how much our predicted mesh resembles the ground-truth object mesh, we used volume IoU which is the volume of intersection of two meshes divided by the volume of the union of two meshes. However, figuring out intersection mesh and union mesh of two 3d meshes is not easy. Thus we voxelized meshes, which makes intersection and union calculation and volume calculation easier. However, the voxelized mesh has only surface voxels since mesh has only surface information. To obtain voxelized mesh with voxels inside the mesh, voxel carving method [30] was

used. The voxel can be represented with $V \in \mathbb{R}^{n_x \times n_y \times n_z}$ where $V_{i,j,k} = 1$ if voxel at (i, j, k) index exists and 0 if not. Thus using voxel, IoU of two voxels V_1 and V_2 be calculated as,

$$IoU = \frac{V_1 \cdot V_2}{V_1 + V_2}, \quad (4.1.2)$$

where \cdot and $+$ are logical and logical or operation. The IoU score becomes 1 as two meshes becomes similar.

4.2 Optimization vs Learning

Unlike our approach, most of the previous works of fitting shape primitives to partially observed point clouds were done using optimization algorithms. Thus, we compared the performance of deformable-superquadric fitting using optimization and network learning. Specifically, gradient descent optimization and our SDSFN were compared. Similar to the superquadric fitting loss 3.1.1 used in SDSFN, the sum of radial distance δ between points and the superquadric surface is minimized in optimization. The matching loss of the z-axis is the original SDSFN loss was omitted, since the supervision of the z-axis cannot be given in the optimization scheme. For learning, SDSFN trained in section 4.3.3 was used. We fitted deformable-superquadrics to 6 partially observed primitive shapes. Each primitive shape in the test data has a total of 30 different data; 10 different shapes are seen from randomly selected 3 different viewpoints.

Figure 4.1 shows the full shapes of primitive shapes estimated by the two methods. Both results shows closely fitted point clouds to the estimated surfaces. However, cone, cylinder, and truncated cone estimated by the optimization method are larger than the ground-truth meshes.

Table 4.1 shows the IoU of predicted and ground truth meshes. Although the

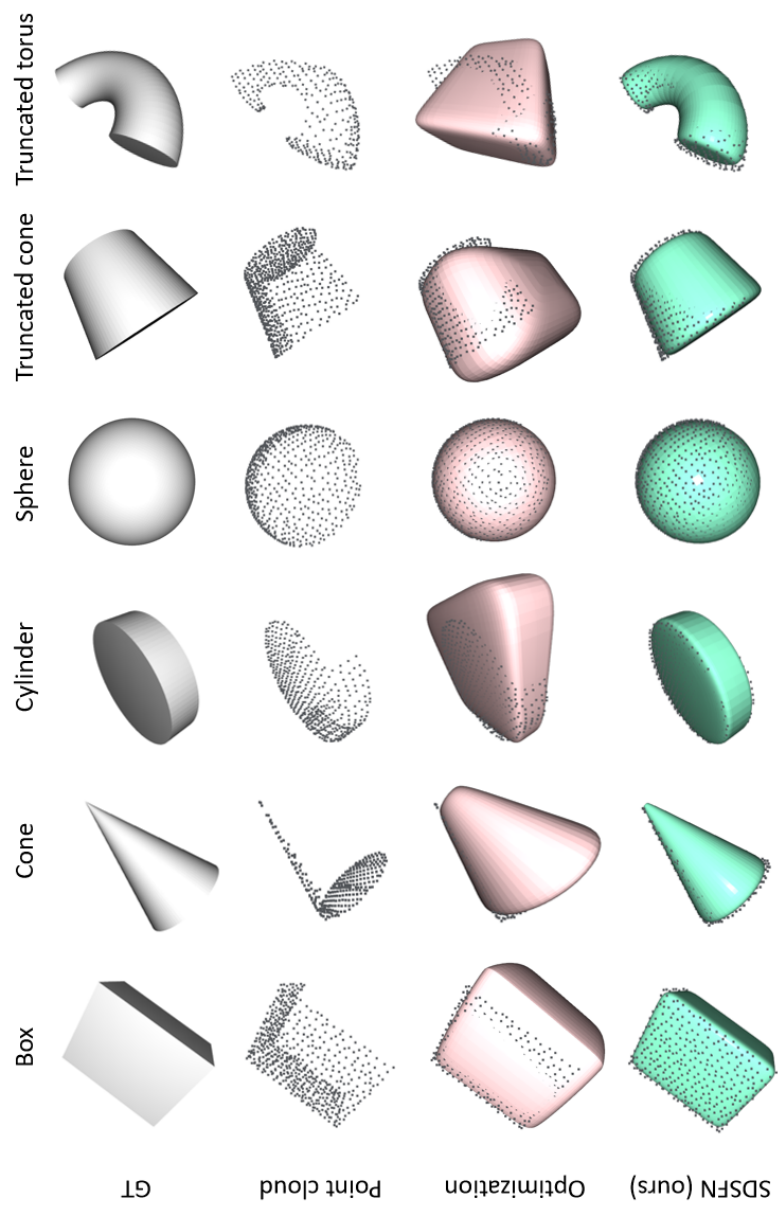


Figure 4.1: Comparison of optimization and learning based fitting performance.

Table 4.1: Fitting of Partially Observed Point Cloud

	box	sphere	cylinder	cone	truncated cone	truncated torus
SDSFN	0.8187	0.9763	0.8417	0.8899	0.7273	0.8695
optim	0.4161	0.9642	0.5219	0.3350	0.4344	0.2977

loss optimized by gradient descent method was much smaller than the lowest validation loss of our SDSFN, SDSFN showed greater performance in predicting the full 3d shape. The optimization method fitted every observed point close to the surface, thus the globally optimum solution was found. However IoU score was low since most of the predicted full shape was bigger than the ground-truth shape. A partially observed point cloud does not have enough information to bound the predicted shape in the unobserved region. Thus predicted shape of the region outside the observed point cloud can have any shapes, and yet the predicted shape can have a globally minimum fitting loss.

For optimization, fitting points with a smaller volume of the superquadric can increase the risk of higher fitting loss, while fitting points with bigger superquadrics does not have such risk. Thus fitting results obtained from optimization have predicted volume outside the ground-truth shape. This is undesirable for robot grasping, since the grasp planner may attempt to grasp a region that actually does not exist.

On the other hand, our learning framework shows that we can estimate the full 3d shape of an object even with a partially observed point cloud. Our network uses past experience to generalize the mapping from partial observation to full 3d shape and well estimate the geometry of the unobserved region. Also, our SDSFN

outperformed the optimization method in terms of computation speed. The computation time for estimating the full geometry took average of 0.023 seconds, while the optimization method took 21.43 seconds. Thus, when estimating the full geometry of a partially observed point cloud, learning-based methods are advantageous than classical optimization-based methods.

4.3 Recognition Module

Since we cannot know the ground-truth segmentation label and ground-truth full 3d mesh for real object data, we tested the performance of our recognition on synthetic data. Both segmentation network and primitive fitting network (SDSFN) are trained using DGCNN [24].

4.3.1 Dataset

In order to train our network, we created a novel data set that fits our requirements. For the segmentation network, we need ground-truth labels of objects' primitive segmentation. For the fitting network, we require full point clouds of shape primitives. Thus we created a dataset consisted of synthetic objects that resemble household objects.

The synthetic objects are made with combinations of shape primitives. Although our framework can fit superquadrics to arbitrary shapes, we selected and used 6 shape primitives to generate complex household objects for convenience. We choose box, cylinder, sphere, cone, truncated cone, and truncated torus as our shape primitives, as they are simple yet variety enough to generate household objects. Note that cone, truncated cone, and truncated torus cannot be represented with simple primitives used in the previous works. Also, these shapes cannot be

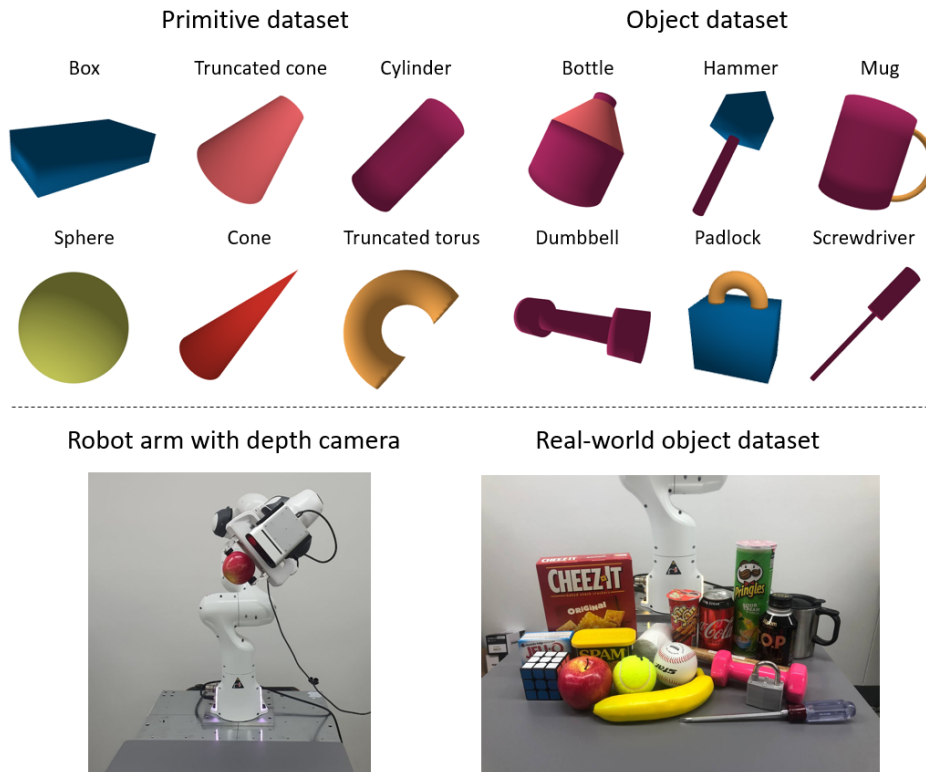


Figure 4.2: Dataset used for training our networks and grasping experiment.

expressed using superquadrics, thus deformable-superquadrics are needed to estimate their shapes.

Our household objects include objects with each of 6 primitives and 6 more objects with several shape primitives put together; bottle, cup, screwdriver, padlock, hammer, dumbbell. Using these shape primitives and synthetic household objects, we created two separate data sets for segmentation training and fitting training.

The segmentation data set has a total of 19 thousand data with each 12 object class has 100 objects with different shapes. A single data consists of partially

observed point clouds with 3000 points and their point-wise label. The partially observed point cloud is obtained by rendering depth images of objects from one of 16 pre-defined viewpoints.

The fitting data set has total 9 thousand data with each 6 shape primitives has 100 different shapes. Data in this data set has partially observed point cloud seen from one of 16 viewpoints and full point cloud and mesh of the object. For fitting data, the partially observed point cloud has 1500 points.

All point clouds and ground-truth meshes in the data set are transformed to the point cloud frame of partially observed point cloud described in section 3.1.

4.3.2 Segmentation

When training, noises sampled from a standard normal distribution are added to the synthetically generated input point clouds to increase the robustness of the segmentation performance on real data. For evaluation, an average of point-wise segmentation accuracy was used.

4.3.3 Fitting

As the training data of the segmentation network, noises are added to the training data of the fitting network. Also, input point clouds are normalized to efficiently learn surface fitting of point cloud that has various scales.

We learned two separate SDSFN; one that fits box, cylinder, sphere, cone and truncated cone (superquadrics and tapered superquadrics) and the other that fits truncated torus (bent superquadrics). The reason why we trained the networks separately is SDSFN could not fit truncated torus when trained with all 6 primitives. We believe this is due to the data imbalance, since bent primitive data

occupy small portion of the entire data. We are planing to solve this issue in the future work. For now, we used the results from both networks and compared the radial-distance based fitting score. We chose the deformable-superquadrics parameters with smaller fitting loss.

We compared our result with our implementation of [5]. We used our segmentation network, instead of the Mask-RCNN they used, thus the difference between their work and ours is the ICP algorithm and SDSFN used for primitive fitting. For ICP based primitive fitting, 6 shape primitives that were used for our SDSFN training were used. Shape primitives were created with 5 discrete values of each shape primitives, thus a total $5^{(\text{number of parameters})}$ shape primitives were fitted to the point cloud. To avoid local minima, we used multi-start ICP as in the original paper. Shape primitive parameters with the highest fitting score was chosen as the final primitive shape.

Table 4.2 shows mean IoU values of two methods evaluated on synthetic data. SDSFN shows higher fitting performance than the ICP algorithm in all cases, except dumbbell where the difference is very small. The average inference time of our SDSFN was 0.036 seconds, while ICP algorithm took 11.51 seconds. Thus, our method outperforms the ICP based fitting algorithm both in terms of estimation performance and computation speed.

The result of our recognition module can be seen in Figure 4.3. Compared to the ICP algorithm, our SDSFFN well estimate the full geometry, close to the ground-truth shape. Even with multi-start ICP, the ICP algorithm fell to local minima and showed poor fitting performance of complex objects. Due to the discrete parameter space, the ICP algorithm could not estimate shape parameters precisely. In order to estimate precise parameters with the ICP algorithm, one

Table 4.2: Full Shape Estimation of Objects

	box	cone	truncated cone	sphere	cylinder	truncated torus
ICP	0.7768	0.7686	0.7270	0.8082	0.7941	0.2426
SDSFN	0.8080	0.8765	0.8927	0.9021	0.9400	0.8052
	hammer	mug	screwdriver	padlock	dumbbell	bottle
ICP	0.5937	0.7918	0.4568	0.7192	0.7268	0.7962
SDSFN	0.8278	0.8792	0.7104	0.8295	0.7264	0.8506

should increase the resolution of parameter space in the compensation of the computation time.

4.4 Grasping Results

For the grasping experiment, we used 7-dof Franka-Emika-Panda robot with its parallel two-finger gripper. Kinect Azure camera was mounted on the end-effector of the robot to get point clouds of objects.

Partially observed point clouds of objects were obtained from raw point cloud through the following procedure. First, we transformed coordinates of raw point clouds represented in the camera frame to world frame (base frame of the robot) using forward kinematics. Then, we only excluded points that are not above the table. Points on the table surface are removed by detecting a plane using RANSAC algorithm. We further removed statistical outliers by removing points that are further away than the average distance of neighborhood points to obtain a segmented object point cloud.

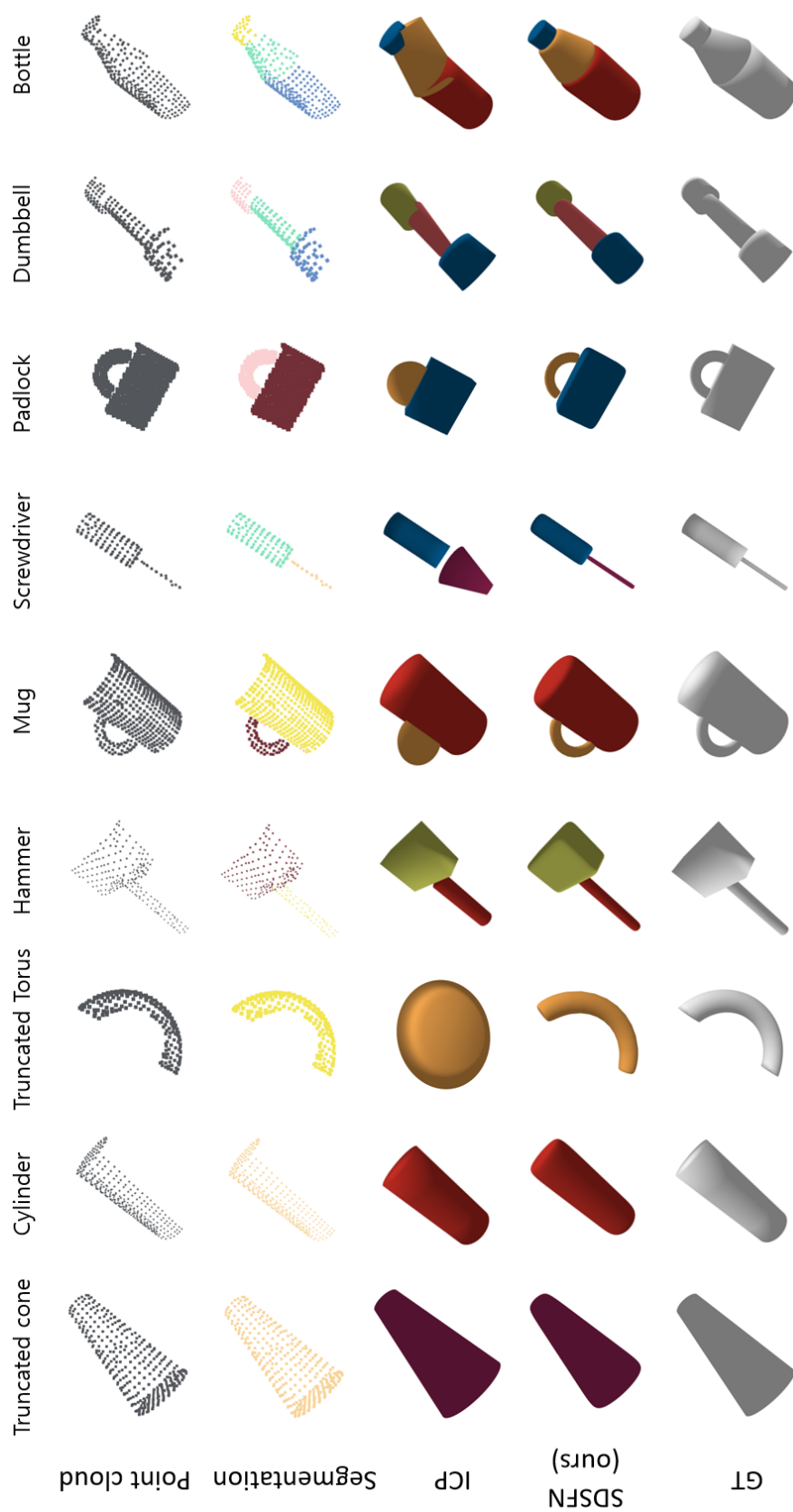


Figure 4.3: Results of full shape estimation of various synthetic objects.

Segmentation and fitting networks are trained with 1500 points and 3000 points of point cloud. Thus we up-sampled or down-sampled obtained object point cloud to 1500 and 3000 points. For up-sampling, points are randomly sampled and noises are added. For down-sampling, an optimization algorithm based on voxelization was used.

Figure 4.4 shows an example of grasping experiment done on real data. At first, point cloud of a hammer was acquired using a depth image seen from a single viewpoint. Notice that only top part of the hammer exists in the point cloud at this stage. After passing this point cloud to the segmentation network, the point cloud is split into two; head and handle, which both have a cylinder shape. The primitive fitting network uses these segmented point clouds to predict the full 3d shape of each point cloud as cylinders. It can be seen that our primitive fitting networks predicted the full shape of the hammer just using the partially observed point cloud. Notice that the predicted full 3d shape and the observed point cloud overlaps, which indicates the high performance of our fitting network. Using our grasping algorithm based on antipodal sampling, we were able to successfully grasp the hammer.

Further experiments of grasping were conducted with the following objects. These objects are selected from YCB object dataset [31], which are widely used for the benchmark of robot grasping. The dataset includes 77 household objects from 5 different categories; food item, kitchen item, shape item, task item. We selected 14 among the dataset that best resembles the original dataset, which can be seen in Figure 4.2. We evaluated grasp success rate on all objects. Each object was randomly placed on a table and the 5 tests were done for each object. In the experiment, the robot grasped the objects with the grasp pose generated by our method, and lifted it up, moved and placed in an empty bin. The grasp was

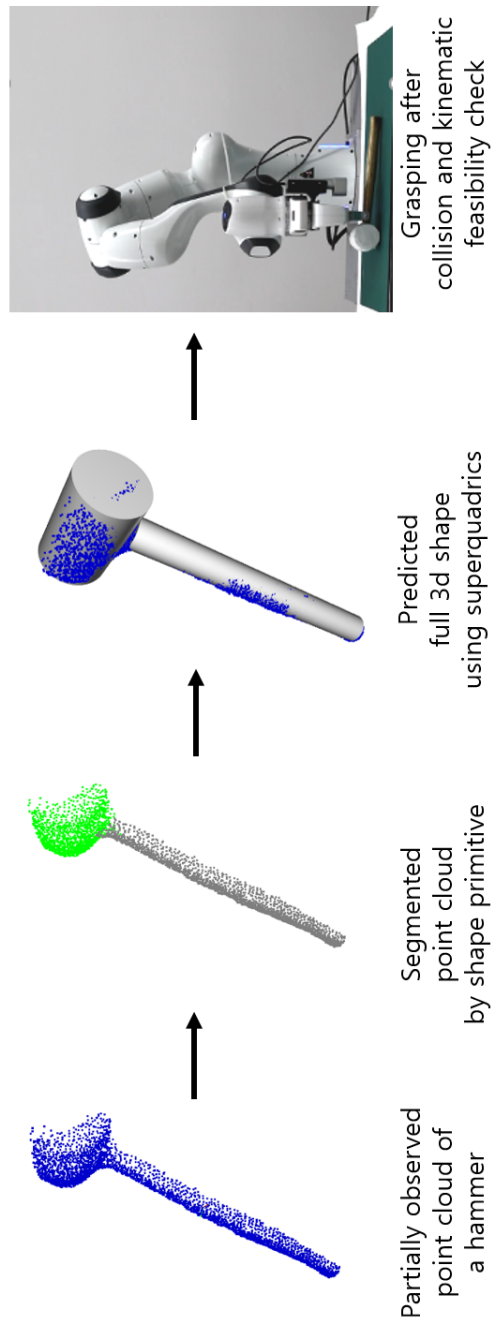


Figure 4.4: Grasping Result.

Table 4.3: Grasping Performance

snack box	cube	spam	pringles	baseball	tennis ball	apple
5/5	5/5	4/5	4/5	5/5	5/5	3/5
bottle	screwdriver	hammer	dumbbell	banana	mug	padlock
5/5	5/5	3/5	1/5	4/5	5/5	5/5

evaluated as successful if the object is successfully placed in the bin.

Table 4.3 shows the overall grasping performance. For simple shapes that consists of single primitive such as box, cube, ball, the success rate was 100 %. Apple is not a perfect sphere thus full shape estimation result was relatively poor compared to the balls, but still was successfully grasped at 60 %. The result shows that our method well reconstructs and successfully grasp objects with multiple primitives such as bottle and screwdriver. In every case of hammer and dumbbell, shape estimation result was good to the tester’s eye, and the gripper successfully reached antipodal grasp poses generated by our method. However, the robot failed to lift them up due to their heavy weight. We were able to successfully grasp objects with bent shapes such as banana, mug, and padlock. Among three, the grasping result on banana was lowest due to the irregular shape.

5

Conclusion

We developed a grasping method that can grasp an unseen object from partial observation. We proposed a new learning framework called SDSFN (Supervised Deformable-Superquadric Fitting Network) to estimate the full geometry of partially observed point clouds with deformable-superquadric primitives. Previous works used optimization algorithms to fit partially observed point cloud to shape primitives. However, optimization algorithms cannot inform or constrain the geometry where points do not exist. Thus, although they show a low fitting error of partially observed point clouds, the estimated full shapes are often far from the true geometry. Thus we used supervised learning whereas supervision of ground-truth full geometries of objects are given to the network. To represent complex objects with shape primitives, we proposed deformable-superquadrics as our shape primitives. We showed that using our supervision learning framework with radial distance loss, we could successfully learn the full geometry of partially observed box, cylinder, sphere, cone, truncated cone, and sliced torus.

Through experiment, we validated our claim that the learning framework is superior compared to the optimization-based methods, when estimating the full geometries of partially observed objects. Also, we showed that our recognition framework well estimates the full geometry of both synthetic objects and real-world objects, and even works well when estimating complex objects. The computation time for shape estimation took about 0.04 seconds which was much faster than the previous work. Using our recognition framework with antipodal grasping method, the robot could grasp a variety of household objects.

Our method could learn to grasp with much fewer data than the end-to-end learning-based approaches. Also, compared to their methods, our grasping pipeline can be easily implemented to various grippers with different shapes. However, our simple sampling-based grasp pose planner requires a quiet amount of computation time to evaluate all sampled grasp poses. We believe that this could be improved through an implementation of other advanced grasping algorithms.

Bibliography

- [1] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2013.
- [2] Kai Huebner, Steffen Ruthotto, and Danica Kragic. Minimum volume bounding box decomposition for shape approximation in robot grasping. In *2008 IEEE International Conference on Robotics and Automation*, pages 1628–1633. IEEE, 2008.
- [3] Manuel Bonilla, Daniela Resasco, Marco Gabiccini, and Antonio Bicchi. Grasp planning with soft hands using bounding box object decomposition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 518–523. IEEE, 2015.
- [4] Matthias Nieuwenhuisen, Jörg Stückler, Alexander Berner, Reinhard Klein, and Sven Behnke. Shape-primitive based object recognition and grasping. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–5. VDE, 2012.
- [5] Yunzhi Lin, Chao Tang, Fu-Jen Chu, and Patricio A Vela. Using synthetic data and deep networks to recognize primitive shapes for object grasping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10494–10501. IEEE, 2020.
- [6] V-D Nguyen. Constructing force-closure grasps in 3d. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 240–245. IEEE, 1987.

- [7] Van-Duc Nguyen. Constructing force-closure grasps. *The International Journal of Robotics Research*, 7(3):3–16, 1988.
- [8] Xiangyang Zhu and Jun Wang. Synthesis of force-closure grasps on 3-d objects based on the q distance. *IEEE Transactions on robotics and Automation*, 19(4):669–679, 2003.
- [9] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.
- [10] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2901–2910, 2019.
- [11] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.
- [12] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: a large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11444–11453, 2020.
- [13] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. A survey of simple geometric primitives detection methods for captured 3d data. In *Computer Graphics Forum*, volume 38, pages 167–196. Wiley Online Library, 2019.

- [14] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007.
- [15] Giulia Vezzani, Ugo Pattacini, and Lorenzo Natale. A grasping approach based on superquadric models. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1579–1586. IEEE, 2017.
- [16] Giulia Vezzani, Ugo Pattacini, Giulia Pasquale, and Lorenzo Natale. Improving superquadric modeling and grasping with prior on object shapes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6875–6882. IEEE, 2018.
- [17] Abhijit Makhal, Federico Thomas, and Alba Perez Gracia. Grasping unknown objects in clutter by superquadric representation. In *2018 Second IEEE International Conference on Robotic Computing (IRC)*, pages 292–299. IEEE, 2018.
- [18] Narunas Vaskevicius and Andreas Birk. Revisiting superquadric fitting: A numerically stable formulation. *IEEE transactions on pattern analysis and machine intelligence*, 41(1):220–233, 2017.
- [19] Franc Solina and Ruzena Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE transactions on pattern analysis and machine intelligence*, 12(2):131–147, 1990.
- [20] Erik Roeland Van Dop and Paul PL Regtien. Fitting undeformed superquadrics to range data: improving model recovery and classification. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)*, pages 396–401. IEEE, 1998.

- [21] Terrance E Boulton and Ari D Gross. Recovery of superquadrics from depth information. In *Proc. Workshop on Spatial Reasoning and Multi-Sensor Fusion*, pages 128–137, 1987.
- [22] Alok Gupta, Luca Bogoni, and Ruzena Bajcsy. Quantitative and qualitative measures for the evaluation of the superquadric models. *Technical Reports (CIS)*, page 843, 1989.
- [23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.
- [24] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.
- [25] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
- [26] Andreas ten Pas and Robert Platt. Using geometry to detect grasps in 3d point clouds. *arXiv preprint arXiv:1501.03100*, 2015.
- [27] Lokbondo Kung Parker Won, Nailen Matschke. Introduction to computer graphics. <http://courses.cms.caltech.edu/cs171/assignments/hw7/hw7-notes/notes-hw7.html>, 2018.
- [28] Jia Pan, Sachin Chitta, and Dinesh Manocha. Fcl: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pages 3859–3866. IEEE, 2012.

- [29] Kevin M Lynch and Frank C Park. *Modern Robotics*. Cambridge University Press, 2017.
- [30] Thomas Bonfort and Peter Sturm. Voxel carving for specular surfaces. In *9th IEEE International Conference on Computer Vision (ICCV'03)*, volume 1, pages 691–696. IEEE Computer Society, 2003.
- [31] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.

국문초록

로봇 파지에서 새로 본 복잡한 물체를 부분 관측만으로 파지하는 것은 어려운 문제이다. 이 문제를 해결하는 하나의 방법은 전체 형상을 먼저 예측한 후 파지하는 것이다. 이전의 연구들은 부분 관측된 점 구름과 단위 형상의 표면 사이의 거리를 최소화하는 최적화하는 문제를 풀어 부분 관측된 물체의 전체 형상을 예측하였다. 하지만, 이러한 최적화 기반의 방법들은 점 구름이 존재하지 않는 부분에 대해 맞춰질 단위 형상의 모양에 대해 정보를 주지 못한다. 이로 인해 기존의 방법들은 부분 관측된 물체의 전체 형상 예측 성능이 떨어진다. 이러한 문제를 해결하기 위해, 우리는 부분 관측된 전체 형상을 예측할 수 있는 새로운 지도 학습 방법을 제안한다. 또한, 우리는 다양한 형상을 연속적인 변수 공간에서 표현할 수 있는, 변형 가능한 슈퍼 쿼드릭을 복잡한 물체를 근사하기 위한 단위 형상으로 제안을 한다. 이를 이용하면 이전의 연구와 비교하여 부분 관측된 물체의 전체 형상을 잘 예측하는 것을 실험적으로 확인하였다. 우리의 방법으로 예측한 전체 형상에서 정반대의 두 점을 찾아 물체를 성공적으로 파지할 수 있는 자세를 찾았다.

주요어: 파지 자세 생성, 부분 관측된 점 구름, 전체 형상 예측, 지도 학습, 변형 가능한 슈퍼 쿼드릭

학번: 2019-23899