Use Authorization

In presenting this dissertation in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission to download and/or print my dissertation for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this dissertation for financial gain shall not be allowed without my written permission.

Signature .....

Date .....

Autonomous Robotic Grasp Planning by SuperEllipsoid Representation

by

Abhijit Makhal

A Dissertation

submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in The Department of Mechanical Engineering

Idaho State University

Spring 2018

Committee Approval

To the Graduate Faculty:
The members of the committee appointed to examine the thesis of ABHIJIT MAKHAL find it satisfactory and recommend that it be accepted.

.....
Alba Perez Gracia
(Major Advisor)

.....
Marco P. Schoen
(Committee Member)

.....
Ken Bosworth
(Committee Member)

.....
Vitit Kantabutra
(Committee Member)

.....
Steven Moody
(Graduate Faculty Representative)

Acknowledgments

THANK YOU VERY MUCH, I would like to express my gratitude to my advisor Dr.Alba Perez Gracia for her invaluable guidance. Also, to all my laboratory members at Measurement and Control Engineering Research Center at Idaho State University, especially Omid Heidari for all the help and thoughtful discussions that contributed to the experiments. I am thankful to Prof. Frederico Thomas and Prof. Maya Cakmak for inviting me to IRI-Barcelona and University of Washington, where my knowledge and motivation in the field of robotics have been significantly boosted by all the new techniques and cutting-edge hardware. My special gratitude towards Open Source Robotics Foundation (OSRF), ROS-Industrial and Google Summer of Code for the opportunity provided to me. I am grateful to Alex K. Goins and Shaun Edwards for their guidance throughout my journey in becoming a developer. My thanks To Kevin M. Croft from Idaho National Laboratory for providing us with a UR5 arm for the experiments. Thus, I would like to acknowledge the work of open source and free software communities such as Robot Operating System(ROS), Point Cloud Library(PCL), OMPL(Open motion planning library), MoveIt and OpenRave as the Ph.D. thesis is highly dependent on such marvelous endeavors. My sincere thanks and respect to my parents for their support and motivation. My gratitude towards all my colleagues in several labs and friends from all over the world who helped me in different ways in the development of the work. Last and not the least, my wife Debashree and son Abhraneel, who encouraged me in every way possible to carry on towards my goal.

This work is supported by the National Science Foundation under Grant No. 1208385. The content is solely the authors' responsibility.

Abhijit Makhal
Pocatello, Idaho

This thesis is dedicated to Debashree and Mistu.

Table of Contents

List of Figures

List of Tables

Autonomous Robotic Grasp Planning by SuperEllipsoid Representation
Dissertation Abstract – Idaho State University (2018)

Though robots have been used in manufacturing industries for decades in precise and repetitive tasks, nowadays they are entering the realm of household environments. While the ”dream” is of an autonomous robotic butler, still the present scope is limited to only vacuum cleaners and lawn mowers, due to the limited capability specifically in the field of grasping and manipulation in the unstructured environment. The thesis approaches towards a goal of grasping novel objects using low-cost and noisy sensors with an incomplete object model, by incorporating a superellipsoid modeling. A complete grasping system is proposed relying on real-time superellipsoid representation of partial-view objects and incomplete object modeling, well suited for unknown symmetric objects in cluttered scenario followed by optimized antipodal grasping. The incomplete object models are processed through a mirroring algorithm that assumes symmetry to first create an approximate complete model and then fit for superquadric representation. The grasping algorithm is designed for maximum force balance and stability, taking advantage of the quick retrieval of dimension and surface curvature information from the superquadric parameters. The pose of the superquadric with respect to the direction of gravity is calculated and used together with the parameters of the superquadric and specification of the gripper, to select the best direction of approach and contact points. The grasp hypotheses are filtered by incorporating a reachability map ensuring the execution of grasps. The methods are evaluated on custom datasets containing objects in isolation as well as in clutter on three different robotic platforms a) a 7DOF WAM arm, b) a 6DOF UR5 arm with 2-finger gripper and c) a PR2 mobile manipulator. Though the method is based on simplistic shape and geometry information, it outperforms existing state-of-art learning-based algorithms in cluttered environments measured against time efficiency and accuracy.

Key Words: Grasp Planning, Unknown Objects, Superellipsoid, Symmetry model

Chapter 1: Introduction

It has always been a childhood dream of many people to have a companion, a "Robot" in the home. But as simple and cordial the term sounds, the essence of the word is not so soothing. "Robot" is a synonym of "forced labor". Despite the prejudicial tone, *a robot is a machine, especially one programmable by a computer, capable of carrying out a complex series of actions automatically* [Oxford English Dictionary]. As the technologies continue to comfort us with technical advancements, human desire to rely more on technologies is increasing rapidly; even for simplistic tasks such as bringing or fetching objects, cleaning and even cooking. Some may find this human behavior of leaning towards technologies for such trivial tasks offensive. But according to 2017 U.S Census Bureau, 15% of Americans are above the age of 65 and the number has a rapid growth. The statistics also suggest that one in five people have some disabilities. In the year 2060, the number will reach 25%; *i.e.* one in four of people will need some help in doing the regular household chores. The field of robotics has the potential to bring cheap and reliable solutions to such healthcare problems.

According to Wikipedia, the term "Robot" is first used by the Czech writer Karel Ćapek in his 1920 play Rossumovi Univerzální Roboti (Rossum's Universal Robots). But history suggests that the efforts of creating artificial human started far before that, even in ancient Greece or Chinese culture. In the current evolution phase of artificial humans, we can find robots mostly in industrial settings taking care of the precise and repetitive tasks. But the journey from the industrial scenario to a human environment has many constraints that need to be solved before we can see a robot "butler" in our home.

The core problem preventing robots from working in the household environment is that unlike the factory environment, the domestic environment is dynamic, unstructured and uncontrolled. The conditions of the home environment make robot incapable working with offline conventional programming. A method relying on the complete knowledge of the state of the environment and robot is most suitable for such scenes. The challenges are similar in grasping problems. Robotic grasping enables a robot to fetch or manipulate objects such as picking, placing, pouring, cleaning, using tools, etc which are an essential capability of a service robot. To enable safe and efficient grasping through a robot, the grasping points or suitable regions where the robot's hand can be placed need to be identified. This can be formulated as an optimization problem over the grasp location and the surface of the object. In a home environment, the range of graspable objects can be huge. This situation makes the approach of creating a database of possible objects infeasible. Thus algorithms are needed to generate grasps online for the objects rather than using stored 3D models of the objects.

Figure 1.0.1: Several service robots present today: From left to right, top to bottom, a) PR2 by Willow Garage, b) Relay from Saviaoke, c) Amigo robot of Eindhoven University of Technology, d) Care-O-Bot 4 by Fraunhofer, e) Reem robot from PAL robotics, f) Pepper from SoftBank Robotics.

1.1 Motivation

Just after the moment of birth, a human child can hold onto the finger of the mother, before knowing any basics about grasping. Enabling such kind of capability on a robot involves various sub-tasks such as identifying objects, searching for appropriate regions or grasps on the surface of the object, and execution of those grasps depending upon the task.

The main sensor for object identification for a human is the eye, which is the result of

billion of years of evolution. Unlike such complex mechanism, the sensors in a service robot rely on noisy and incomplete data. From this data, an approximate model of the object needs to be constructed prior to grasp synthesis. If a previous instance of the object model is present in the robot memory, it is possible to overlap the existing model with the perceived sensor data and search for grasp regions. But to grasp unknown objects, this method is not suitable as no previous occurrence of the object is present. The system has to cope with missing information to generate an approximate full model while planning for grasping.

There are existing algorithms such as GraspIt [86] which search for grasps offline, assuming a complete model of the object is present. But the absence of a complete model of the object makes these algorithms infeasible for unknown objects. Nevertheless, while an approximated model can be created by estimating the occluded and missing part of the object, grasping directly on these sensor data may lead to erroneous grasping which may further result in damage on the object, robot, environment or the participating human. Feasible grasps can be found when the approximated model is estimated by a mathematical model and geometry-based grasps are searched on the mathematical model. Additionally, all the object identification and grasp synthesis tasks need to be performed in a time-efficient manner.

Before allowing the robot system to perform the grasp, it should be ensured that the grasp can be executable by incorporating environment and robot constraints. A representation of the environment and kinematics of the robot should be utilized by which the grasps that may be outside the reach of the robot or may bring collision to the system are discarded.

1.2 Problem Statement

This thesis investigates the problem of grasping unknown objects, from partial and incomplete sensor data where the grasp is specified by high-dimensional grasp space of grasp position and orientation, gripper opening angle and an approach direction of the robot arm. The scenarios for grasping unknown objects may vary depending upon the robot configuration and setting of the environment. For a typical end-to-end robot pick and place operation, the following commands may be provided by the user:

- *"Robot, pick up the object"*
- *"Robot, pick up all the objects on the table and put them in the box"*
- *"Robot, Clear the table"*

As the robot is supposed to work in a scenario where the objects are not specified, it is the responsibility of the system to define which object it is supposed to pick up and

how. Moreover, the system should perform following tasks autonomously in a safe and time-efficient manner.

1. Model point cloud data to object model
2. Grasp unknown object in isolation
3. Grasp unknown object in clutter
4. Ensure safe and reliable grasping

1.2.1 Formal problem definition

Formally, the problem can be defined as:

Stable and executable grasps G should be found on a robot R, where the point cloud $PC \subset \mathbb{R}^3$ and some obstacles $\mathbb{OB}$ in a robot co-ordinate frame C_R is provided. The obstacles are 3D structures that are not graspable such as the table. A grasp G can be defined by a vector $G = (P, O, \alpha, d)$ where P is the position and O is the orientation of the gripper while grasping. α denotes the opening angle of the gripper and d is the approach direction. A grasp can be considered stable if it maintains the property of force-closure[95] or form-closure[17]. It is executable if the physical robot R can reach and execute the grasp trajectory without any collision.

1.2.2 Challenges in Grasping

The thesis deals with grasping objects in real time in real environments. These are some following issues which generate difficulties in grasping in the real world.

- Perception: The complete grasping system relies on the perception module as the main input. While various sensors utilize different strategies for perceiving the environment, proper and noiseless object surfaces cannot be obtained every time. Some surfaces such as textureless or shiny surfaces are very hard to perceive. The environmental constraints such as lightning and occlusions also affect the perception system.
- Calibration: calibration is one of the huge issues in terms of grasping. Wrong calibration of hand camera system may lead to catastrophic damage to the system or the object.
- Inverse Kinematics Solutions: Without a valid inverse kinematics solution, the whole grasping system becomes incapable. The grasping system can output such grasp pose which may not be reachable by the IK solver.

- Robot Modelling: The robotic system should be modeled beforehand the grasping task. In many grasping systems, URDF (Unified Robot Description Format) is used to model robots. An incorrect modeling can lead to improper grasping.

1.2.3 Thesis Goals

This work aims to provide a new direction towards grasping unknown objects by incorporating superellipsoid modeling and find grasps directly on the model based on its parameters. A complete grasping system framework is presented with the goals of:

1. Exploring new directions in superellipsoid representations by separating the process of fitting the model and pose estimation, which can lead to time-efficient fitting.
2. A novel mirroring scheme which can approximate the occluded regions of the object, which has the proficiency of efficient model fitting.
3. A novel grasp synthesis algorithm that can search grasps directly on the mathematical model considering the geometry of the object and gripper constraints.
4. Combining robot kinematics and environment information in the grasp synthesis process which can result in safe and reliable grasp execution.

1.3 Proposed Framework

The core idea of the proposed grasping system can be viewed in the Fig.5.5.1. After obtaining the point cloud data from the sensor device, the workspace of the environment containing the table and the objects on the table are considered. The table and the objects are separated out and the point cloud supported by the table are further forwarded for processing. If multiple objects are present on the table, the clusters of points are further segmented to separate out multiple clusters where individual clusters can represent individual objects. The system highly relies on the fitting of superellipsoid models on the approximated sensor data. The system can be considered as a combination of multiple optimization problems. The first optimization process estimates the pose of the object which is forwarded to the second optimization process that searches for the parameters of the superellipse that can be best fitted to the point cloud data. The final optimization step occurs during grasping where the system searches for the best grasp that considers all the geometry constraints of the fitted superellipse and gripper as well as the constraints posed by the environment and robot kinematics.

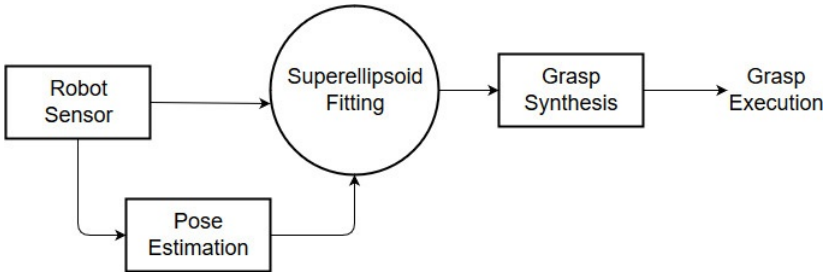


Figure 1.3.1: Scheme of grasping system

Each individual object model is searched for a pose which is a representation of position and orientation of the object in the world frame. The poses and object point cloud are further processed to construct a mirrored surface which in combination to the original point cloud can represent an approximated model of the object. The model is fitted with superellipse and the minimum parameters that best fit the approximated model are obtained.

The minimum parameters that represent the object are forwarded for grasp synthesis which can find multiple grasp hypothesis dependent on the object geometry and gripper parameters. A grasp hypothesis may contain information such as an optimized angle which determines how much the gripper has to open, a pose of the gripper which decides where the gripper angles should be placed and finally approach information deciding the direction by which the arm has to come towards the object. The grasp hypotheses are filtered and searched for optimized final grasp which can ensure grasp execution based on robot kinematics and environment setup.

1.4 Thesis Structure

This thesis provides a system of grasping unknown objects in the isolated and cluttered environment by superellipsoid representation. The structure of the thesis is as follows:

Chapter 2 discusses the representation of objects from sensor data. It provides a background on processing 3d point cloud, feature estimation and a detailed analysis of various segmentation and clustering techniques.

Chapter 3 presents the mathematical background behind superellipsoids. It provides a detailed explanation of superquadric sampling, fitting and normal estimation. Finally, it formulates the process of pose estimation by iterations.

Chapter 4 discusses the concepts of reachability map incorporated in the work of grasp filtering. After a brief explanation of robot kinematics, it presents a discussion on reachability maps created by Reuleaux library.

Chapter 5 introduces the main work of grasping of unknown objects by superellipsoid

fitting. After a discussion on the basic concepts of robotic grasping and a state-of-the-art of various types of robotic grasping techniques, it explains the process of grasp hypothesis generation and grasp filtering. The mirroring algorithm for object shape approximation and the full grasping pipeline is discussed.

Chapter 6 presents the experimental results on three different robotic platforms for grasping in isolation and table clearing tasks. The results of Reuleaux library and superquadric fitting are also presented.

Chapter 7 presents the conclusion, objectives achieved and future work related to the thesis.

1.5 Glossary

For the sake of clarity, the basic terminologies used throughout the thesis are defined here:

Sensor: A sensor primarily detects or measures a physical property of the environment. In the thesis, mainly low-cost 3D sensors such as Kinect1 and Kinect2 are used.

Point Cloud: The point cloud is a 3D representation of sensor data captured by a 3D sensor represented mainly by (x, y, z) coordinates. Additional features such as color, normal information can also be incorporated.

Segmentation: It is a process by which the object of interest can be defined separately in a scene from rest of the objects.

Pose: A pose is a collection position and rotation information. The rotation can be represented by Euler angles or quaternions.

Arm: An arm is a serial kinematic chain constructed by multiple joints and is mainly a carrying device of the hand. The arm can also be termed as a manipulator.

Configuration: A configuration is a set of joint values which can be applied to the robot arm.

End-effector: An end-effector is a gripper attached to the final link of the robot arm which is generally used to manipulate objects. It can also be referred as the robot hand.

Mobile Manipulator: A mobile manipulator is a specific kind of robot which is capable of manipulating objects by its attached manipulators and able to navigate through the wheels.

Robot Kinematics: Robot Kinematics is related to the field of mechanics by which the motion of the robot manipulator can be defined. By forward kinematics, given joint angles, the final pose of the end-effector can be determined. With inverse kinematics, while provided a final pose of the end-effector, the joint angles needed to reach the pose can be determined.

Rechability Map: The reachability map is a collection of all the poses that the robot manipulator can reach. It can also be represented by the cluster of areas defining reachable

space.

Superellipsoid: A superellipsoid is a mathematical model mainly popular in computer graphics community. Typical superellipsoids are superquadric and supertoroid.

Grasp: A grasp is a particular configuration of the end-effector where the gripper is in contact with the object, not necessarily satisfying any task constraints. In our thesis, a grasp is considered by mainly three components, a) a final pose where the gripper is touching the object, b) an opening angle of the gripper and c) approach direction which determines from the way the robot is going to approach the object.

Object in Isolation: A condition where the object in consideration is only present in the scene.

Object in Clutter: A condition where the object under consideration is present with several other objects, while some or all of them is in contact with each other.

Chapter 2: Object Representation

Our primary goal is to put the robots in a home environment. When the term "robot" appears, the first image comes to our mind is of enormous arms working day and night in industries and factories with repetition and accuracy stealing jobs of human. But the situation is changing rapidly, as robots are not only limited to industries only. Researchers around the world are trying to move them in our human environment as a social device caring for elders, helping children even in personal services. In the industrial environment, the task of the robot is quite easy, as the scenario is built up for the benefit of the robot. The sensors are dedicated in detecting only the objects in the production. On the other hand, in the unstructured environment, such in our home, lives the most uncertain being in the universe, the human. The setting is modeled for the benefit of the human. Therefore, the sensors should be modeled in such a way where the robot has to acquire the continually changing human environment. There is no guaranty that an object would still be in the same place where it was half an hour ago. So the robot has to detect and recognize the object and the environment every time while finding it.

Representing the object is one of the most significant tasks for any robotic grasping system. It has to classify beforehand which object in the environment it has to manipulate, which objects are obstacles and which objects can be considered as support. As shown in Fig.6.0.1, the system has to understand that the table is just a support surface on which the objects are sitting, the objects upon the table which should be manipulated and rest in the scene are obstacles.

Figure 2.0.1: Objects and the environment

Representations of objects can be categorized based on the output dimension of the sensors.

We can represent objects in 1D, 2D even in 3D based on the application need and sensor structure. The next section discusses different types of sensors and processing data from the sensor output.

2.1 1D Object Representation

Objects can be most basically represented by just 1 dimension. Though not very useful for the complete object representation, 1D representation can be used in applications such as avoiding obstacles for mobile robot navigation, where the robot has to decide how much far or near the object is from the robot. Basic example of sensors that can provide 1D representation are IR, ultrasonic, laser, haptic sensors etc.

Figure 2.1.1: Different 1D sensors(from the left to right) a)ultrasonic, b)IR, c)Laser.

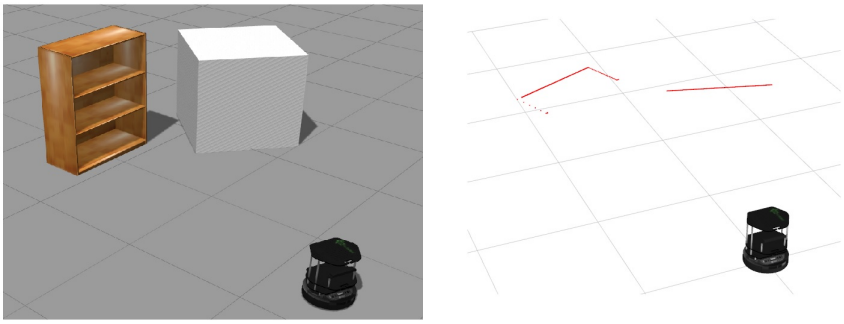


Figure 2.1.2: 1D sensor visualization

As displayed in fig.2.1.2, the robot detected 2 objects in the environment by the laser sensor visualized by the red lines.

2.2 2D Object Representation

In a 2D representation, an object is represented in the form of image which is a composition of pixels. Each pixel in the image has own (x, y) coordinates in the window and based on

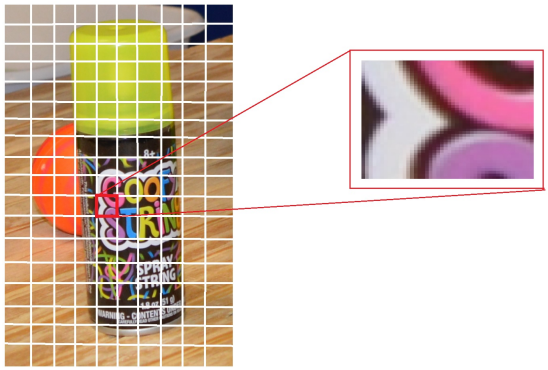


Figure 2.2.1: 2D image pixels

their color, grayscale or RGB, it stores intensity values.

The resolution of an image is the total number of pixels in the image. To obtain an object from a 2D image, we have to segment out the pixels which may contain the object. If the process further needs the recognition of the object, it has to select some predefined features of the object in the segmented pixels and compare it with a image database stored separately.

Image representation and recognition as a different field of study, is beyond the scope of this document. For more details, please refer to [65].

2.3 3D Object Representation

Humans perceive their environment in 3D and interpret the world on the basis of what they see, among other sensing inputs. Looking at an object we can decide the object's distance from us. The preliminary results and progress over 2D community suggest that for robots the world can be perceived in similar way by image recognition. For few years, it was substantially believed that by only perceiving the world in 2D, we can achieve the same results of perception as human[65]. But this erroneous theory miscomprehendes the facts of simple constraints such as lighting conditions, scale, occlusions and so on. In fig 2.3.1 and 2.3.2 we can see how framing the perception problem in this way can lead to failures in capturing the true semantic meaning of the world.

There are mainly two reasons for preferring 3D over 2D representation. The first one is monocular computer vision applications are flustered by both fundamental deficiencies in the current generation of camera devices and limitations in the datastream itself. The former will most likely be addressed in time, as technology progresses and better camera sensors are developed. An example of such a deficiency is shown in Figure 2.3.1,where due to the low dynamic range of the camera sensor, the right part of the image is completely

Figure 2.3.1: Model matching failure in underexposed 2D images. None of the features extracted from the model (left) can be successfully matched onto the object of interest (right)(image from https://pointclouds.org/).

Figure 2.3.2: An example of a good model match using features extracted from 2D images (left), where a beer bottle template is successfully identified in an image. However, zooming out from the image, we observe that the bottle of beer is in fact another picture stitched to a completely different 3D object (in this case a mug). The semantics of the objects in this case are thus completely wrong (image from https://pointclouds.org/).

underexposed. This makes it very hard for 2D image processing applications to recover the necessary information for recognizing objects in such scenes. On the other hand computer vision applications make use of 2D camera images mostly, which are inherently capturing only a projection of the 3D world. Figure 2.3.2 attempts to capture this problem by showing two examples of matching a certain object template in 2D camera images. While the system apparently matched the model template (a beer bottle in this case) successfully in the left part of the image, after we zoom out, we observe that the bottle of beer was in fact another picture in itself, stitched to a completely different 3D geometric surface (the body of a mug in this case). This is a clear example which shows that the semantics of a particular solution obtained only using 2D image processing can be lost if the geometry of the object is not incorporated in the reasoning process. [111]

2.4 3D Capturing Devices

Three dimensional (3D) point clouds provide very important cues to analyze objects or environments. They are, for instance, heavily used in topographical mapping, where an airplane or a satellite passes over an area and takes several snapshots with a laser range finder. In [39], authors used laser range finders to build height map of the scanned area. The work of [61] tried to develop such kind of distance sensor in the field of mobile robotics, where the robot tries to find a path, avoiding obstacles from the 3D data. Also in navigation, the robot has to recognize certain features that can be used as landmarks. During the DARPA Grand Challenge and Urban Challenge, laser range finders have been used extensively on board of the different autonomous vehicles to build a 3D map of the immediate environment. One of the most important properties for mobile robots is their ability to process information from the surrounding environment. There has been a lot of work done in the area of laser range finders and scanners for perceiving 3D. The work of [96] used point clouds to reconstruct historic monuments. [60] discusses the used of radial basis function to reduce the number of point clouds in a point cloud dataset. Similar work has been done in [24] where algorithms are presented to reconstruct objects from a point cloud dataset. In order to perform its tasks, the robot needs to be able to perceive its environment.

In order to grasp an object, the robot needs to first find the object in the scene. To execute this task, 3D object detection and classification have to be performed. [66] presented their work where the 3D descriptors are presented in terms of spin images for object recognition.

There are several types of 3D capturing devices available. The use of cheap devices such as Microsoft kinect has revolutionized the 3D capturing system for RGBD images. Figure 2.4.1presents some of the devices used for capturing the environment in 3D. The Time-

Figure 2.4.1: 3D capturing devices: (from the left) Laser range finder, Time-of-flight cameras, Kinect Camera, Stereo Camera, PR2 head with two stereo camera pair.

of-flight cameras or Laser Measurement systems (LMS) or LIDAR systems sends rays of light(laser) or sound(sonar) in the world, which will reflect and return back to sensor. With the knowledge of the speed with which a ray propagates, and using precise circuitry to measure the exact time when the ray was emitted and when the signal returned, the distance d can be estimated as (simplified):

$$d = \frac{ct}{2}$$

where c is the speed of ray and t is time for signal emitted and coming back to the sensor.In contrast, triangulation techniques usually estimate distances using the following equation (simplified):

$$d = ||\frac{ft}{x_1 - x_2}||$$

where f represents the focal distance of both sensors, t the distance between the sensors, and x_1 and x_2 are the corresponding points in the two sensors [65]. Though many different triangulation systems exist, the most popular system used in mobile robotics applications is the stereo camera.

An example of 3D point cloud has been presented in Figure 2.4.2. Dense 3D point clouds can be processed and meaningful data can be extracted by Point Cloud Library(PCL). [111]

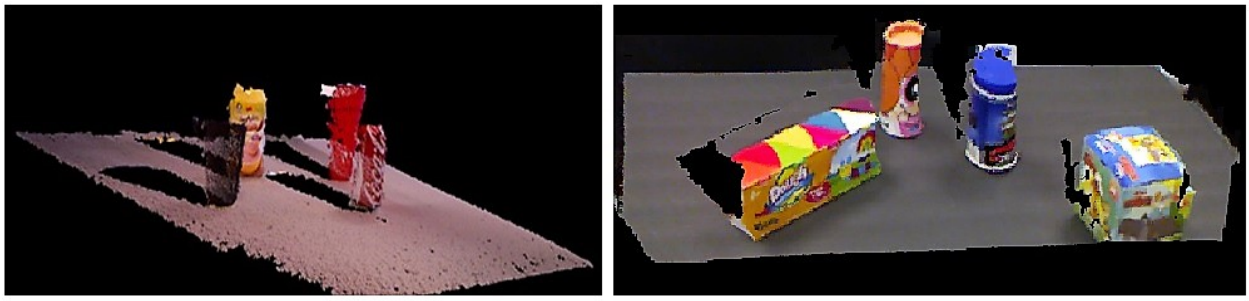


Figure 2.4.2: An example of point cloud data with some example objects: cup, glass, gerber-box,cokecan, boxes, toy cylinder.

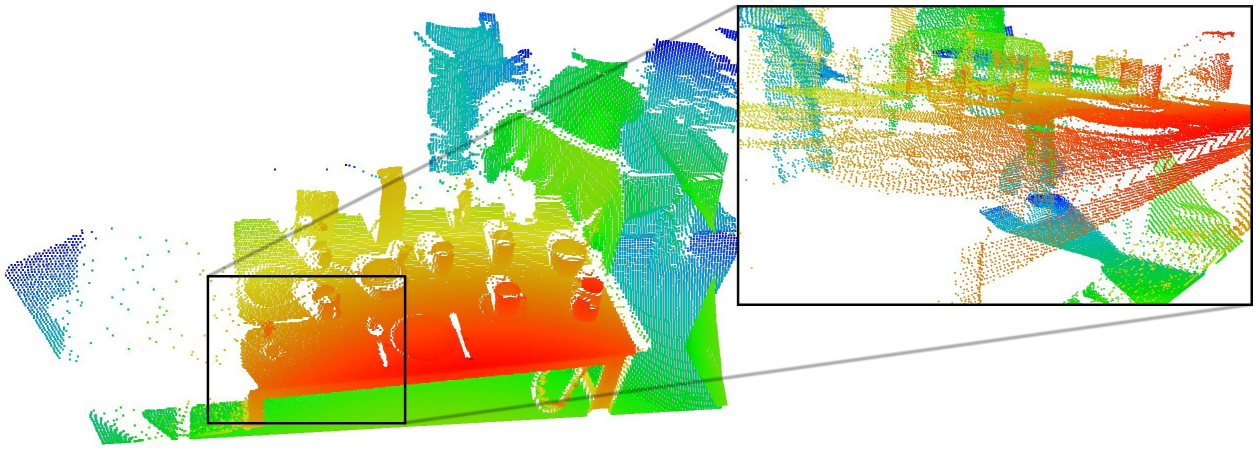


Figure 2.4.3: Point Cloud Dataset showed in distance (red is close, blue is far away) spectrum(image from https://pointclouds.org/).

$$\begin{bmatrix} x_1 & y_1 & z_1 & r_1 & g_1 & b_1 & l_1 & d_1 \cdots \\ x_2 & y_2 & z_2 & r_2 & g_2 & b_2 & l_2 & d_2 \cdots \\ & & & & & & & \cdots \\ x_n & y_n & z_n & r_n & g_n & b_n & l_n & d_n \cdots \end{bmatrix}$$

Figure 2.5.1: Point cloud data shown in terms of x,y,z,r,g,b,d

2.5 Processing in 3D

An important aspect when dealing with point cloud representations is that they can store much more information than just the 3D positions of points as acquired from the input sensing device. During the acquisition process of a point cloud P, the distances from the sensor to the surfaces in the world can be saved as properties for each resultant 3D point $p_i \in P$. The representation of pointcloud based on their distances is presented in fig 2.4.3 where the close objects are shown in red and far objects are indicated in blue.

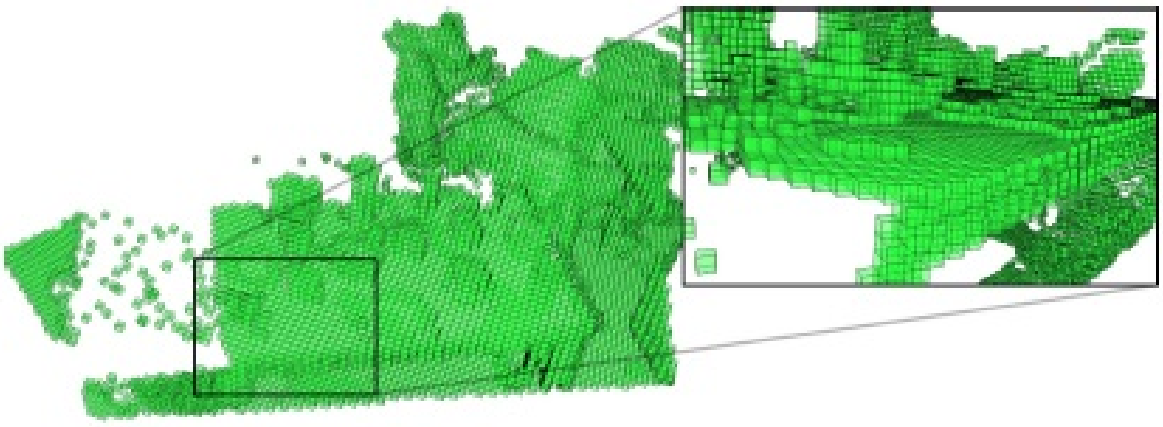


Figure 2.5.2: Volumetric representation using a octree structure, with a leaf size of 1.5cm(image from https://pointclouds.org/)

2.5.1 3D data representation

Given the fact that point cloud representations can hold multiple properties per point, the definition of a point $p_i = \{x_i, y_i, z_i\}$ changes to that of $p_i = \{f_1, f_2, f_3....., f_n\}$, where f_i denotes a feature value in a given space (color, class level, geometry etc.) thus changing the concept of 3D to nD. From these requirements, we can deduce that an appropriate I/O data storage format for a point cloud P, would be able to save each point with all its

attribute values on a new line in a file, and thus have a file with n lines, for the n total number of points in P. A fictitious example is shown in the Fig. 2.5.1, where x_i, y_1, z_i represents the 3D coordinates, r_i, g_1, b_i are color associated with each point, l_i is the class level, and d_i is the distance from the surface. To obtain the geometry around a query point p_q, most geometric processing steps need to discover a collection of neighboring points p^k, that represents the underlying scanned surface through sampling approximations. A solution is to use spatial decomposition techniques such as kd-tree or octree [54], and partition the point cloud data into P chunks. Though by implementation most spatial decomposition techniques can represent as a volumetric representation of P as shown in fig. 2.5.2. Here all the points are enclosed in boxes with different widths namely "voxels".

2.5.2 Normal Estimation

Surface normals are important properties of a geometric surface and are heavily used in many areas such as computer graphics applications, to apply the correct light sources that generate shadings and other visual effects. Given a geometric surface, it is usually trivial to infer the direction of the normal at a certain point on the surface as the vector perpendicular to the surface at that point. However, since the point cloud datasets that we acquire represent a set of point samples on the surface, there are two possibilities:

1. obtain the underlying surface from the acquired point cloud dataset, using surface meshing techniques, and then compute the surface normals from the mesh;

2. use approximations to infer the surface normals from the point cloud dataset directly.

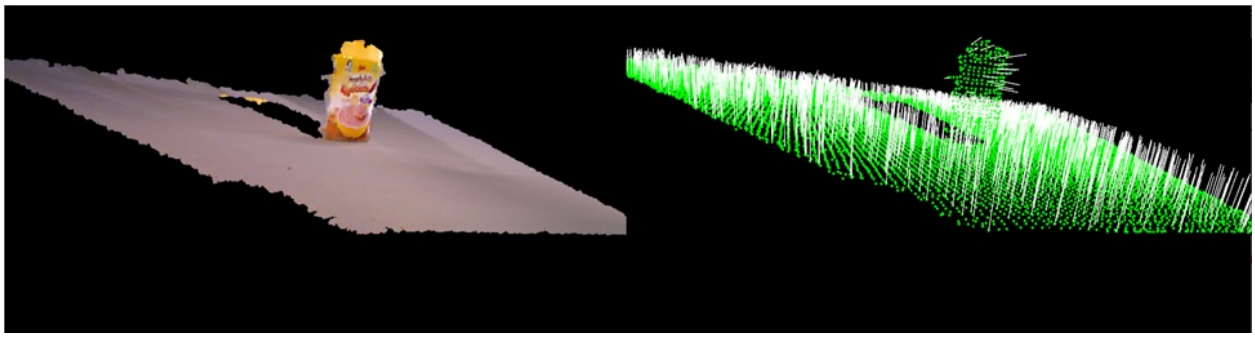


Figure 2.5.3: Point cloud data of a gerber box(left), the estimation of normals (right)

The solution for estimating the surface normal can be reduced to an analysis of the eigenvectors and eigenvalues (or PCA – Principal Component Analysis) of a covariance matrix

created from the nearest neighbors of the query point. More specifically, for each point p_i, we can assemble the covariance matrix C as follows:

$$C = \frac{1}{k}.\sum_{i=1}^{k}.(p_i - \vec{p}).(p_i - \vec{p})^T, C.\vec{v_j} = \lambda_j.\vec{v_j}, j \in \{0, 1, 2\}$$

where k is the number of point neighbors considered in the neighborhood of p_i, $\vec{p}$ represents the 3D centroid of the nearest neighbors, λ_j is the j-th eigenvalue of the covariance matrix and $\vec{v_j}$ is the j-th eigenvector. An estimation of surface normal of a point cloud data has been presented in Fig. 2.5.3.

2.5.3 3D Filtering

Point clouds data can be filtered in several ways. In the process of acquisition of 3D point cloud data, there may be several noise due to sensor inefficiency. Otherwise the point cloud becomes dense, where more data is present than we need. So processing the unneeded data makes the system unstable or increases the time needed for processing. In a voxel grid filter, the cloud is divided in multiple cube-shaped regions with the desired resolution. Then, all points inside every voxel are processed so only one remains. The simplest way would be to randomly select one of them, but a more accurate approach would be to compute the centroid, which is the point whose coordinates are the mean values of all the points that belong to the voxel[111].

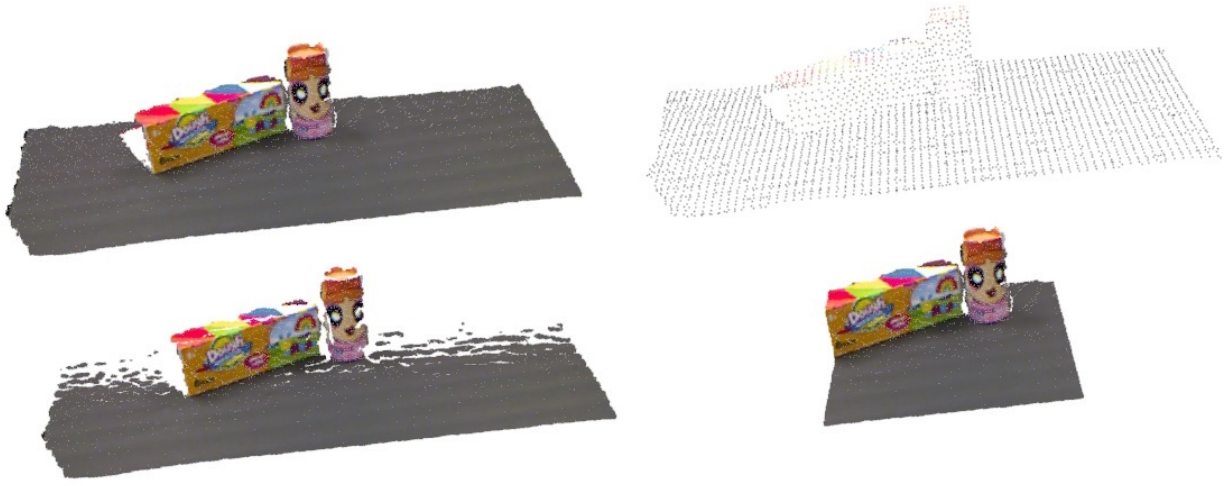


Figure 2.5.4: Filtering point clouds. From left to right, up to down, a) Original captured image, b) voxel grid filtered, c) statistical outlier removal filtered, d) passthrough filtered

In a passthrough filter, certain points will be removed from a point cloud dataset whose

values do not fall within a user-provided certain range. As for example, filtering point cloud data based on y value will remove all the points situated on the table. In a conditional removal filtering, all the points will be filtered based on a certain condition provided by the user. In a statistical outlier removal filtering process, or every point, the mean distance to its K neighbors is computed. Then, if we assume that the result is a normal (gaussian) distribution with a mean and a standard deviation , we can consider it safe to remove all points with mean distances that fall out of the global mean plus deviation. It runs a statistical analysis of the distances between neighboring points, and trims all which are not considered "normal". So the points outside of the region, will be removed. Some of the filtering process have been shown in the fig. 2.5.4

2.6 Object Segmentation

Segmentation is the process of dividing the cloud data into segments or clusters, where each cluster may represent meaningful objects in the scene. Segmentation can be performed based on points, normals or even textures. By segmentation, we can obtain which part of sensed points correspond to the objects we want to manipulate [111].

First, the dominant plane should be detected, which corresponds to the support surface on which the objects are resting on. The point cloud scene can be processed by Random Sample Consensus (RANSAC) [47] to find the dominant plane in the scene. This iterative method helps in the estimation of the parameters of a mathematical model from a set of observed data containing outliers. The points belonging inside the mathematical model are considered "inliers", and the points outside the model are "outliers". To find a tabletop surface, a plane model can be fitted to the scene, where the points inside the plane model are points corresponding to the table.

After segmenting out the table plane model in the scene, the table shape model is fitted with 2D convex hull polygon by which an estimation of the table can be estimated. Another benefit of this process is that the points normal to the plane can be considered as clusters representing the objects to be manipulated. 2.6.1 represents a general overview of the tabletop segmentation pipeline. After the RANSAC algorithm, a plane is detected in 2.6.1b. The convex hull of the plane is shown in 2.6.1d. The points resting on the table plane can be considered as objects clusters 2.6.1c. The objects clusters now should be segmented out for representation of meaningful objects in the scene.

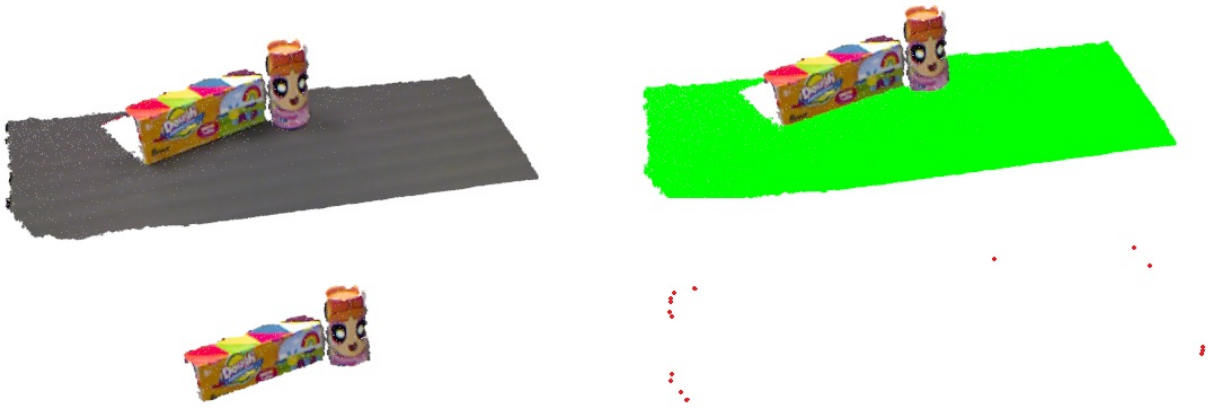


Figure 2.6.1: Tabletop segmentation pipeline. From left to right, up to down, a) Original captured image, b) table plane, c) clusters on the table, d) convex hull

2.6.1 Clustering

To divide an unorganized point cloud set into smaller parts, a clustering method should be implemented. The main benefit of clustering method is that it significantly reduces processing time. Similar to flood fill algorithm for 2D image pixels, we can produce an euclidean clustering algorithm for point clouds using the nearest neighbors. We can use the volumetric representation for the occupied space, or the octree structure of fixed width boxes [111]. Additionally, user-defined conditions can also be implied for a modified conditional euclidean clustering technique, where after the euclidean clustering the user-defined condition is evaluated which can be based on pure euclidean distance, smoothness or RGB information.

A region growing segmentation can also be performed where the algorithm merges the points that are close enough based on smoothness constraints. The points belonging to same smoothness constraints are clustered together. The algorithm first sorts the points based on minimum curvature as the minimum curvature is located in the flat areas. From there it starts to grow its region by comparing its neighbors for the angle between its normal and seed point. While the angle is less than a predefined threshold, the point is added to the region. The region growing segmentation can be modified to use color information instead of normal information. After segmentation, two neighboring clusters which have small color difference average are merged together.

A min-cut segmentation approach performs a binary segmentation, where the output cloud is divided into two sets, which are background and foreground. The algorithm takes object centers and its radius and constructs a weighted graph containing every point as vertices

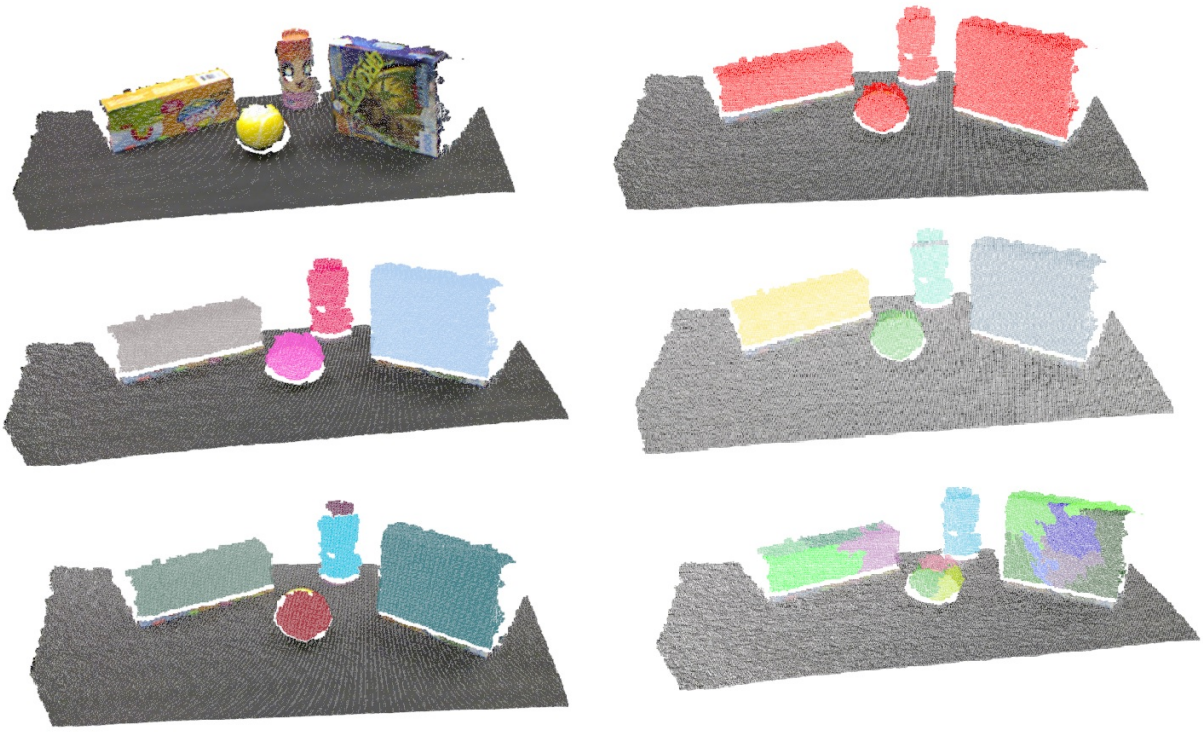


Figure 2.6.2: Different object clustering techniques. From left to right, up to down, a) Original captured image, b) Min-Cut segmentation, c) Euclidean clustering, d) Condition euclidean clustering, e) Region growing segmentation, f) Color based segmentation

and connected with nearest neighbors. Assigning background and foreground penalties, the point cloud is cut into two sections. Different object clustering techniques are presented in the fig. 2.6.2

2.7 Object Oversegmentation

For a dense scene, where objects are closely touching each other, segmentation provides poor performance, as simplistic clustering does not maintain the semantic relationship between the points of the same objects. Superpixel is a well known method in image processing field, which reduces the number of regions to be considered for algorithms that are computationally expensive. Voxel Cloud Connectivity Segmentation(VCCS) algorithm [99] performs oversegmentation on the point clusters using voxel relationship maintaining spatial geometry of the scene. VCCS creates supervoxels using a variation of K-Mean clustering enforcing two constraints:

1. The seeding of supervoxel clustering is performed in the 3d space, instead of the projected image space.

2. Enforcing strict spatial connectivity on occupied voxels it ensures that supervoxels cannot cross boundaries for disjoint set in 3D Space.

First an adjacency graph of the voxel cloud is created and maintained throughout the algorithm where voxels are considered neighbors which are within the octree leaf resolution. Then it incrementally expand supervoxels by a set of seed points which are evenly distributed in space. The expansion process from the seed points are monitored by distance metric D, where D_s is spatial distance normalized by seeding resolution, D_c is the color euclidean distance normalized in RGB space and D_n is the angle between the surface normal vectors. Different weights are implied where W_s is the spatial weight, W_c is the color weight and W_n is the normal vector weight. R_{seed} is the seed resolution.

$$D = \sqrt{W_c {D_c}^2 + \frac{W_s {D_s}^2}{3{R_{seed}}^2} + W_n {D_n}^2} \tag{2.1}$$

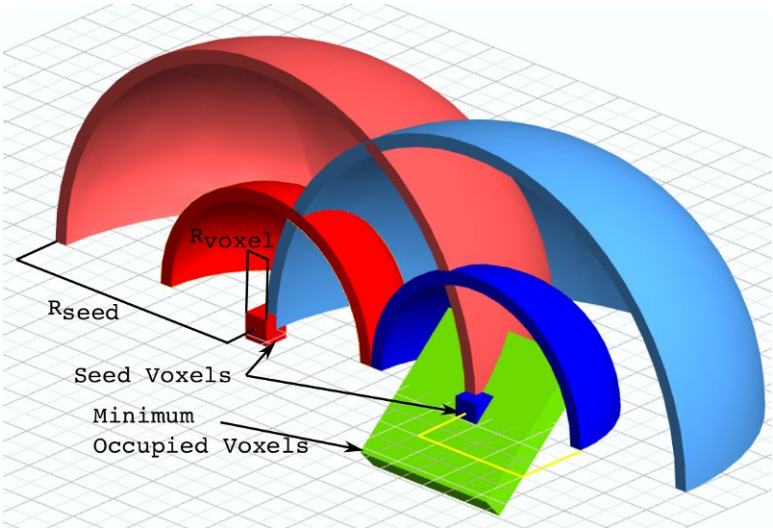


Figure 2.7.1: Supervoxel clustering parameters (image from https://pointclouds.org/)

Fig. 2.7.2 represents the output from the VCCS algorithm, where 2.7.2 is captured scene of unorganized point clouds and 2.7.2b shows the supervoxels and adjacency graphs.

After the supervoxels and adjacencies are extracted from the unorganized point clouds, the edges of the graph can be classified as either convex or concave by imposing some simple criteria described in as Locally Convex Connected Patches (LCCP) [127] algorithm. Individual supervoxels are represented as $\vec{p_i} = (\vec{x_i}, \vec{n_i}, N_i)$ where $\vec{x_i}$ is the centroid, $\vec{n_i}$ is the normal vector $e \in N_i$. By applying an extended convexity criterion it can decided if the

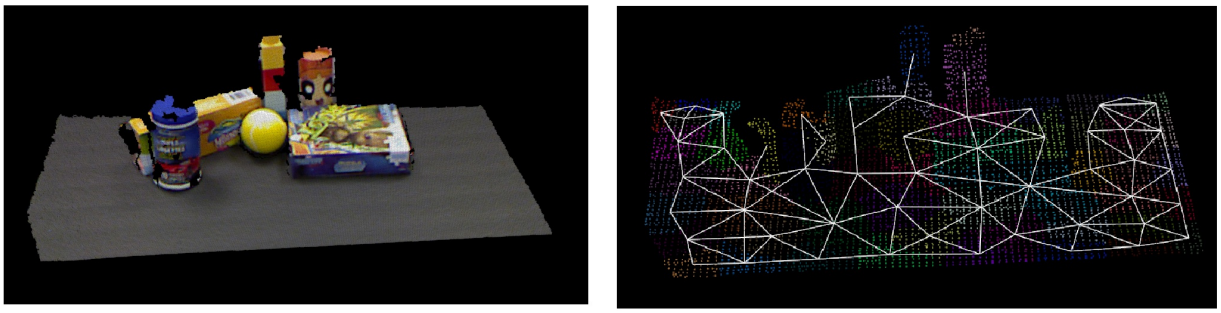


Figure 2.7.2: a) Captured scene, b) Supervoxels and supervoxel adjacency

connection $e = (\vec{p_i}, \vec{p_j})$ between two supervoxels are convex or concave. The basic convexity criterion can be defined by:

$$CC_b(\vec{p_i}, \vec{p_j}) = \begin{cases} true, & (\vec{n_1} - \vec{n_2}) \cdot \hat{d} > 0 \vee (\beta < \beta_{Thresh}), \\ false, & otherwise, \end{cases} \tag{2.2}$$

where $\vec{n_1}$ and $\vec{n_2}$ are the normals from the centroids $\vec{x_1}$ and $\vec{x_2}$. β_{Thresh} is the concavity tolerance threshold and β is the the bias for similar normals. To calculate the angle of the normals to the vector $\hat{d} = \vec{x_1} - \vec{x_2}$ which joins the centroid can be calculated by the dot product $\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos(\alpha)$ with $\alpha = \measuredangle(\vec{a}, \vec{b})$. If the connection is convex, α_1 would be smaller than α_2, which can expressed as:

$$\alpha_1 < \alpha_2 \Rightarrow \cos(\alpha_1) - \cos(\alpha_2) > 0 \Leftrightarrow \vec{n_1} \cdot \hat{d} - \vec{n_2} \cdot \hat{d} > 0, \tag{2.3}$$

where $\hat{d} = \dfrac{\vec{x_1} - \vec{x_2}}{\|\vec{x_1} - \vec{x_2}\|}$. For a concave connection,

$$\alpha_1 > \alpha_2 \Leftrightarrow \vec{n_1} \cdot \hat{d} - \vec{n_2} \cdot \hat{d} < 0, \tag{2.4}$$

For an extended convexity criterion, the neighborhood information should also be included. To consider a connection $e = (\vec{p_i}, \vec{p_j})$ to be convex there should exist a common neighbor $\vec{p_c}$ with $\vec{p_i}$ and $\vec{p_i}$ that has convex connection to both of them. Now, the extended convexity criterion can be defined as:

$$CC_e(\vec{p_i}, \vec{p_j}) = CC_b(\vec{p_i}, \vec{p_j}) \wedge CC_b(\vec{p_i}, \vec{p_c}) \wedge CC_b(\vec{p_j}, \vec{p_j}) \tag{2.5}$$

After categorizing all the edge connection as convex or concave all valid convex connected edges from the same subgraph are merged by a region growing approach. The result of LCCP

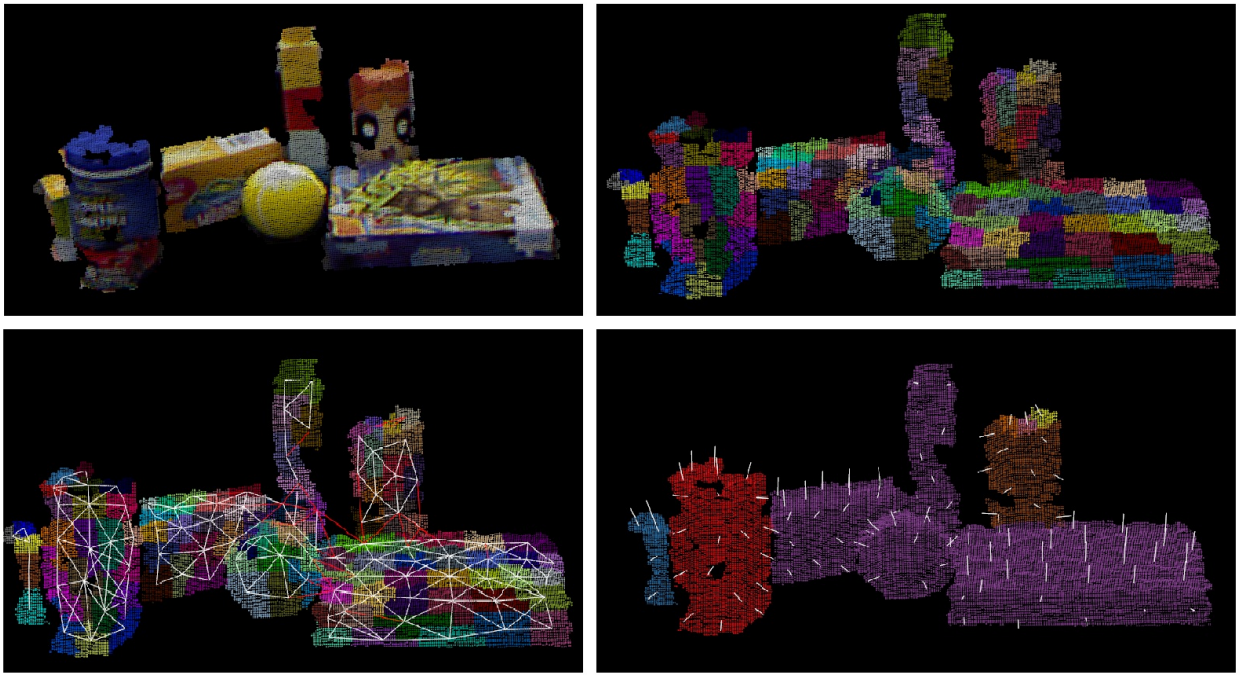


Figure 2.7.3: LCCP segmentation. From left to right, top to bottom a) Captured scene, b) Supervoxels, c) Supervoxels and adjacency graph, d) Segmented objects and normals

segmentation is presented in 2.7.3a and2.7.3d, where the supervoxels and the adjacency graphs are shown in 2.7.3b and 2.7.3c.

After segmenting point clouds in terms of clusters by locally convex connected patches, the objects can be segmented again fro dividing into multiple functional parts. This process can be performed by a Constrained Planar Cuts (CPC) [123] algorithm which produces greddy cuts for the locally concavity graph model. A cost function is implemented rewarding only the cuts orthogonal to concave edges. The adjacency graph is transformed into an Euclidean Edge Cloud representing each point as an edge in the graph. After implementing a directional weighted RANSAC, the cost function becomes:

$$S_m = \frac{1}{|P_m|} \sum_{i \in P_m} w_i t_i \tag{2.6}$$

$$t_i = \begin{cases} |\vec{d_i} \cdot \vec{s_m}| & i \text{ is concave,} \\ 1 & i \text{ is convex,} \end{cases} \tag{2.7}$$

Where $\vec{s_m}$ is the vector perpendicular to the surface model of m and $\vec{d_i}$ is the i^th edge direction calculated by $\vec{d} = \dfrac{\vec{x_1} - \vec{x_2}}{\|\vec{x_1} - \vec{x_2}\|}$

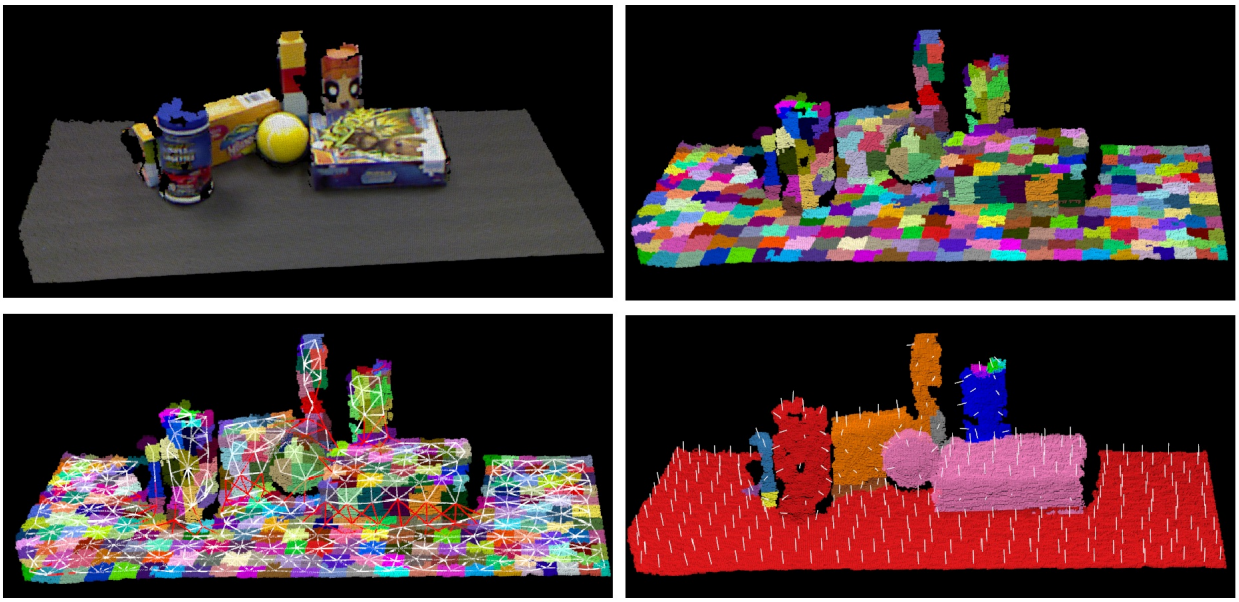


Figure 2.7.4: CPC segmentation. From left to right, top to bottom a) Captured scene, b) Supervoxels, c) Supervoxels and adjacency graph, d) Segmented objects and normals

The results from the constrained planar cuts are represented in 2.7.4a and 2.7.4b where the supervoxels and the adjacency graphs are shown in 2.7.4b and 2.7.4c.

2.8 Model Fitting

Instead of representing objects directly from sensor data, it is advantageous to represent objects in terms of mathematical models which can be fitted on the segmented sensor data. The advantages are such as:

1. By a model it is easy to estimate the quantitative behavior of the data.
2. The mathematical model discards the noise the sensor data
3. In terms of performing further manipulation of sensor data such as grasping, the operation can be performed directly on parameterized models.

The most basic model that can be fitted are implicit polynomial and algebric invariants [129] [22] mainly for object recognition and position estimation from noisy and partial data. The approach towards fitting the models is to minimize the distance function:

$$dist^2(S, f) = \sum_{(x_i,y_i,z_i)\in S} \frac{f^2(x_i, y_i, z_i)}{\|\nabla(f(x_i, y_i, z_i))\|^2} \tag{2.8}$$

Where $f(x_i, y_i, z_i)$ is the gradient and $\|\nabla(f(x_i, y_i, z_i))\|^2$ is the square of the norm of the gradient. The curve coefficients are considered as global descriptors. To fit the model, a comparison between curve coefficients and local region of interest is not feasible as the data for fitting the polynomials imposes constraints among the coefficients of fitted polynomials. The coefficients of the fitted polynomial are constrained to lie on a manifold of lower dimension than the number of coefficients.

In the work of [93], an algorithm to fit Spherical harmonics is presented which is widely used for the purpose of shape reconstruction, shape descriptor, filtering, and frequency based representation. The algorithm is computationally faster than other methods due to the monte carlo integration over the edges. First, it imposes an unfolding split condition to partition the object utilizing kd-tree. For each cell of the kd-tree, best fitting spheres are found and spherical harmonics computations are performed. Triangulating each local patch of the object spherical harmonics computations are performed again for local polar function and finally the representations are blend together for the reconstruction of the overall object. The spherical harmonics $\{Y_l^m(\theta, \varphi) : |m| \leq l \in \mathbb{N}\}$ are special functions defined on the unit sphere $\mathbb{S}^2$ as:

$$Y_l^m(\theta, \varphi) = k_{l,m} P_l^m(\cos\theta) e^{im\varphi} \tag{2.9}$$

where $\theta \in [0, \pi], \varphi \in [0, 2\pi]$, $k_{l,m}$ is a constant and P_l^m is the associated Legendre polynomial. In [68], a rotation invariant representation based on spherical harmonics is discussed. The core advantage of this descriptor lies in improved performance in matching for recognition and dimensionality reduction for compact representation.

Object representation in temrs of Geons, a qualitative model for symbolic object recognition is presented in [142]. The work in [28] proposed a method to model different types of real-world objects by a generalized Cylinders, which is the solid generated by a planar cross-section as it is moved and deformed along an axis [101]. In contrast of generalized cylinders, a symmetry seeking model [132] which is designed into internal forces that constrain the deformation of the model is proposed. The consistent deformation maintains a consistency with the 2D silhouette of the object. The model combines two separate models of a deformable spine model and a deformable tube model. The three forces does the following functionalities: coerces the spine into axial position with the shell, predisposes the tube to radial symmetry around the spine and controls expansion or contraction of the tube around the spine. In [89], an approach towards processing geometric models and efficiently recovering a compact representation of euclidean symmetry has been proposed. Union of balls representation is proposed in [107] where a medial axis is computed first, which is the center of the balls of maximum diameter inscribed inside the object.

In [73] and [72] 3D objects are modeled as blob models assuming 3D models are composed of a set of near-convex 3D simple parts and each part can be described as long and thin, flat and wide or truly 3-dimensional. The long thin parts are called sticks and the flat wide parts are called plates. The parts that are neither thin nor flat are considered blobs. Stick is a 4-tuple set $ST =< En, I, Cm, L >$ where En is the set of two ends of the stick, I is the interior of the stick, Cm is the center of mass and L is the length. Plate can characterized by $PL =< Eg, S, Cm, A >$ where Eg is the edge separating the two surfaces. $S = S1, S2$ is the set of surface points, where $S1$ is point on first surface and $S2$ is the points on second surface. The center of the mass of plate is Cm and A is the area. A blob is $BL =< S, Cm, V >$ where S is the points on the surface, Cm is the center of mass and V is the volume. The whole model is represented by the relational data structure between the sticks, plates and blobs.

Object representation by superquadric models is deeply described in next section.

2.9 Summary

In this chapter, object representation from sensor data is discussed especially in 3D. The processing of point data, segmentation and filtering procedures are analyzed. A brief explanation of different approaches of model fitting are also presented. The superellipsoid model used in this thesis is discussed in detail in next chapter.

Chapter 3: Superellipsoid Representation

3.1 Superquadric Representation

Superquadrics is a generalization or extension of the family of quadrics, or quadric surfaces first introduced by [15] and later extended in the work of [62]. The other members of ellipsoid family are superellipsoids, superhyperboloids of one piece, superhyperboloids of two pieces and supertoroids. The superellipsoid is one of the most popular geometric modeling tools in the computer graphics community because of its diverse property of representing several basic shapes such as cube, cylinder, octahedra, lozenges, spheres and spindles with rounded or sharp corners with a single mathematical equation. The most special property of a superquadric is that they represent closed surfaces on an object-centric coordinate system in an inside-outside function.

3.2 Mathematical Definition

Superquadric is a spherical product of 2-dimensional surfaces achieving a 3-dimensional surface. An unit sphere can be created as the spherical product of a half circle in the plane orthogonal to (x, y) and a full circle which is in the (x, y) plane:

$$m(\eta) = \begin{pmatrix} \cos\eta \\ \sin\eta \end{pmatrix}, \quad with : \eta \in [-\pi/2, \pi/2] \tag{3.1}$$

$$h(\omega) = \begin{pmatrix} \cos\omega \\ \sin\omega \end{pmatrix}, \quad with : \omega \in [-\pi, \pi] \tag{3.2}$$

The spherical product is :

$$r(\eta, \omega) = m(\eta) \oplus h(\omega) = \begin{pmatrix} \cos\eta\cos\omega \\ \cos\eta\sin\omega \\ \sin\omega \end{pmatrix} \tag{3.3}$$

Similarly, a superellipsoid can be obtained by the spherical product of two superellipses of the form:

$$\left(\frac{x}{a}\right)^{\frac{2}{\epsilon}} + \left(\frac{y}{b}\right)^{\frac{2}{\epsilon}} = 1 \tag{3.4}$$

$$r(\eta,\omega) = S_1(\eta) \oplus S_2(\omega) = \begin{Bmatrix} \cos^{\epsilon_1}\eta \\ a_3 \sin^{\epsilon_1}\eta \end{Bmatrix} \otimes \begin{Bmatrix} a_1 \cos^{\epsilon_2}\omega \\ a_2 \sin^{\epsilon_2}\omega \end{Bmatrix} = \begin{Bmatrix} a_1 \cos^{\epsilon_1}\eta \cos^{\epsilon_2}\omega \\ a_2 \cos^{\epsilon_1}\eta \sin^{\epsilon_2}\omega \\ a_3 \sin^{\epsilon_1}\eta \end{Bmatrix}, \quad (3.5)$$

with $\eta \in [\frac{-\pi}{2}, \frac{\pi}{2}]$ and $\omega \in [-\pi, \pi]$, where a_1, a_2, a_3 are the scaling factors of the three principal axes. The exponent ϵ_1 controls the shape of the superquadric's cross-section in the planes orthogonal to (x, y) plane, and ϵ_2 controls the shape of the superquadric's cross-section parallel to the (x, y) plane. The pose of the superquadric with respect to a world frame is specified by the six parameters that define a rigid motion, p_x, p_y, p_z for the position vector and ρ, ψ, θ for defining a rotation matrix, for instance using roll, pitch, and yaw angles. The total set of parameters that fully defines the superquadric consists of 11 variables, $\{a_1, a_2, a_3, \epsilon_1, \epsilon_2, p_x, p_y, p_z, \rho, \psi, \theta\}$

The superellipsoid can also be expressed using an implicit equation in normal form as

$$f(a,x,y,z): \quad \left(\left|\frac{x}{a_1}\right|^{\frac{2}{\epsilon_2}} + \left|\frac{y}{a_2}\right|^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left|\frac{z}{a_3}\right|^{\frac{2}{\epsilon_1}} = 1. \quad (3.6)$$

The superquadric is bounded by the planes given by $-a_1 \leq x \leq a_1$, $-a_2 \leq y \leq a_2$, and $-a_3 \leq z \leq a_3$.

3.3 Normals of Superquadrics

Depending on the value of the exponent, the intersections with the coordinate planes define curves where the surface may not be smooth, and computing their parameters separately is important. Working on the coordinate plane has some advantages regarding symmetry, however in some cases, it is necessary to work on other parts of the superellipsoid surface. The tangets to the superellipse in its differentiable regions can be computed along the parameters η and ω as:

$$t_{e\eta}(\eta,\omega) = \begin{Bmatrix} -a_1\epsilon_1 \cos^{\epsilon_1-1}\eta \sin\eta \cos^{\epsilon_2}\omega \\ -a_2\epsilon_1 \cos^{\epsilon_1-1}\eta \sin\eta \sin^{\epsilon_2}\omega \\ a_3\epsilon_1 \sin^{\epsilon_1-1}\eta \cos\eta \end{Bmatrix} \quad (3.7)$$

$$t_{e\omega}(\eta,\omega) = \begin{Bmatrix} -a_1\epsilon_2 \cos^{\epsilon_1}\eta \cos^{\epsilon_2-1}\omega \sin\omega \\ a_2\epsilon_2 \cos^{\epsilon_1}\eta \sin^{\epsilon_2-1}\omega \cos\omega \\ 0 \end{Bmatrix} \quad (3.8)$$

The normal vector can be defined as the cross product of the tangent vectors along the two coordinate curves,

$$n_\eta(\eta,\omega) = \left\{ \begin{array}{c} -a_2 a_3 \epsilon_1 \epsilon_2 \cos^{\epsilon_1+1}\eta \sin^{\epsilon_1-1}\eta \sin^{\epsilon_2-1}\omega \cos\omega \\ -a_1 a_3 \epsilon_1 \epsilon_2 \cos^{\epsilon_1+1}\eta \cos^{\epsilon_1-1}\eta \cos^{\epsilon_2-1}\omega \sin\omega \\ -a_1 a_2 \epsilon_1 \epsilon_2 \cos^{2\epsilon_1+1}\eta \sin\eta \cos^{\epsilon_2-1}\omega \sin^{\epsilon_2-1}\omega \end{array} \right\} \tag{3.9}$$

3.4 Differential Geometry of Superquadrics

The superellipsoid is convex for the values of the exponent $0 < \epsilon < 2$; for larger values, it becomes concave at the corresponding cross section, as shown in fig.3.4.1

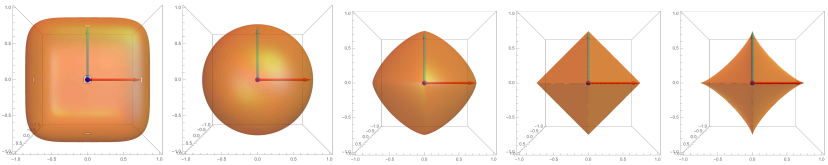


Figure 3.4.1: Convex superquadrics for $\epsilon = 0.4$, $\epsilon = 1$, $\epsilon = 1.4$ and flat for $\epsilon = 2$; concave superquadrics for $\epsilon = 2.4$.

Differential parameters of superconics can be easily calculated from either their implicit or their parametric expression. Considering the implicit equation in Eq.3.6, at the $x - z$ plane obtained when $\omega = 0$, the tangent vector to the curve is:

$$t_e(\eta) = \left\{ \begin{array}{c} -a_1 \epsilon_1 \cos^{\epsilon_1-1}\eta \sin\eta \\ a_3 \epsilon_1 \cos\eta \sin^{\epsilon_1-1}\eta \end{array} \right\} \frac{1}{\sqrt{(-a_1\epsilon_1 \cos^{\epsilon_1-1}\eta \sin\eta)^2 + (a_3\epsilon_1 \cos\eta \sin^{\epsilon_1-1}\eta)^2}} \tag{3.10}$$

and the normal vector,

$$n_e(\eta) = \left\{ \begin{array}{c} a_3 \sin^{\epsilon_1-2}\eta \\ a_1 \cos^{\epsilon_1-2}\eta \end{array} \right\} \frac{1}{\sqrt{(a_3 \sin^{\epsilon_1-2}\eta)^2 + (a_1 \cos^{\epsilon_1-2}\eta)^2}} \tag{3.11}$$

By using the position, tangent and normal vectors and components of parametric expression $r(\eta) = (x, y)$, we can calculate the curvature as:

$$k = \frac{\|t_e'(\eta)\|}{\|r_e'(\eta)\|} = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{3/2}} \tag{3.12}$$

For the superellipse, we can obtain:

$$k_e(\eta) = \frac{(\epsilon_1 - 2)a_1 a_3 \cos^{\epsilon_1 - 4}\eta \sin^{\epsilon_1 - 4}}{\epsilon_1 \sqrt{(a_1^2 \cos^{2\epsilon_1 - 4}\eta + a_3^2 \sin^{2\epsilon_1 - 4}\eta)^3}} \tag{3.13}$$

Similar expressions can be obtained for the other two planes passing through the origin of the superquadrics's frame. Depending on the value of the exponent ϵ_1, the minimum curvaute will be found at the intersection with the axes or at 45° from them. Fig. 3.4.2 shows the effect of the exponent on the location of the minimum curvature; the curvature is constant for $\epsilon_1 = 1$.

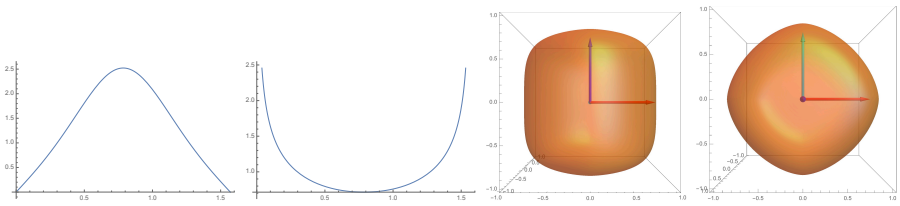


Figure 3.4.2: From top to bottm: a) curvature as a function of the angular parameter and the value of the exponent ϵ in curvature plot, b) cross-sectional cut of the superquadric for same semi-axes length and values $\epsilon = 0.5$ and $\epsilon = 1.2$.

3.5 Superquadric Sampling

In [102], a method is proposed for uniform spatial sampling of points on superquadric models. The arc length between two close points on $x(\theta)$ and $x(\theta + \Delta_\theta\theta)$ on a curve can be estimated by Euclidean distance,

$$D(\theta) = |x(\theta) - x(\theta + \Delta_\theta\theta))| \tag{3.14}$$

where we can use a first-order approximation for $\Delta_\theta\theta$,

$$\Delta_\theta\theta = \frac{D(\theta)}{\epsilon} * \sqrt{\frac{\cos(\theta)^2 \sin(\theta)^2}{a_1^2 \cos(\theta)^{2\epsilon} \sin(\theta)^4 + a_2^2 \sin(\theta)^{2\epsilon} \cos(\theta)^4}} \tag{3.15}$$

By setting $D(\theta)$ to a constant sampling rate, the incremental updates of the angular parameters are

$$\theta_i = \begin{cases} \theta_{i-1} + \Delta_\theta(\theta_i), & \theta_0 = 0, i \in \{1....N\}, \theta_N < \frac{\pi}{2}, \\ \theta_{i-1} - \Delta_\theta(\theta_i), & \theta_0 = \frac{\pi}{2}, i \in \{1....N\}, \theta_N > 0, \end{cases} \tag{3.16}$$

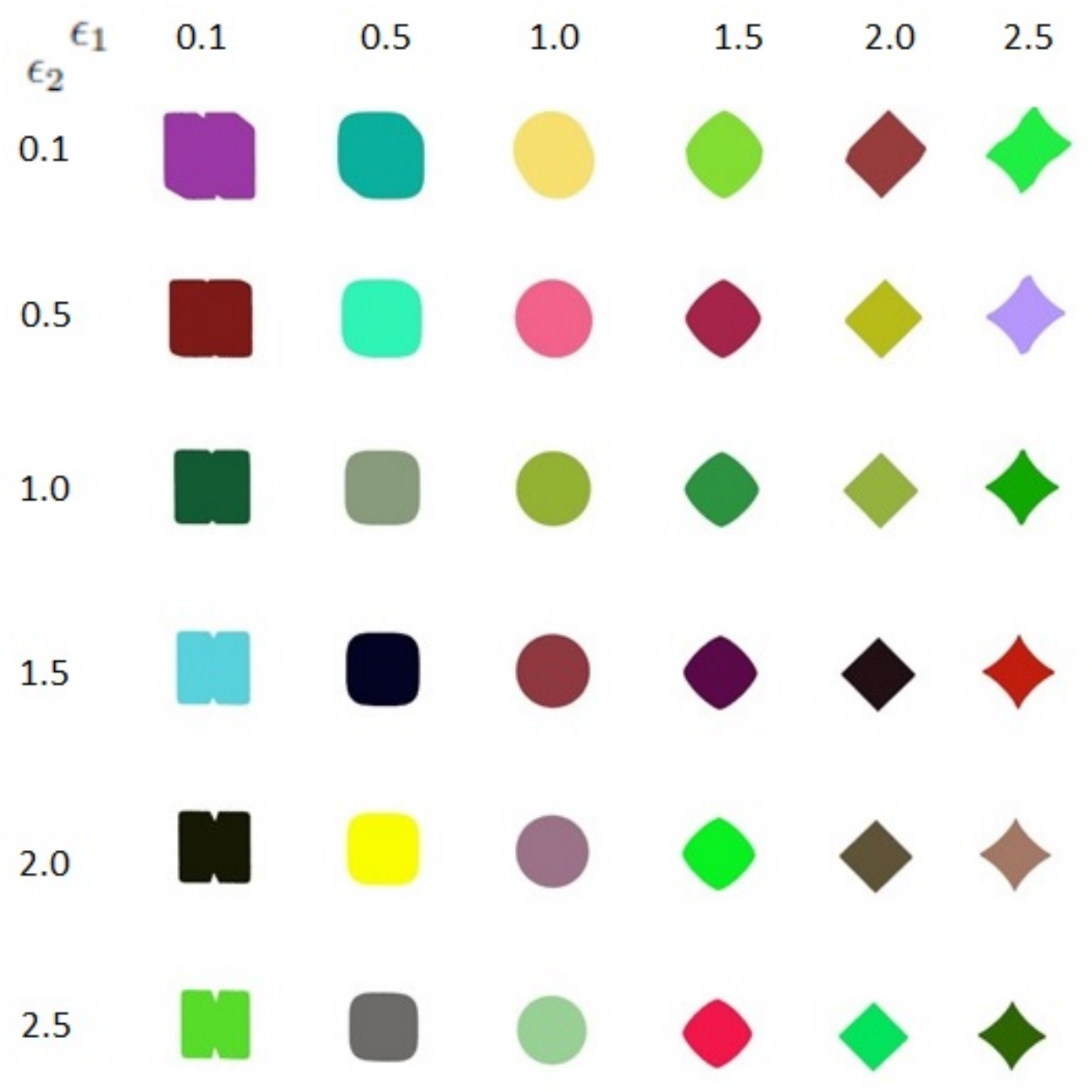


Figure 3.4.3: Superquadric sampling with varying values of ϵ_1 and ϵ_2

where for $\theta \to 0$,

$$\Delta_\theta \theta = \left(\frac{D(\theta)}{a_2} - \theta^\epsilon \right)^{\frac{1}{\epsilon}} - \theta \tag{3.17}$$

and for $\theta \to \frac{\pi}{2}$

$$\Delta_\theta \theta = \left(\frac{D(\theta)}{a_1} - (\frac{\pi}{2} - \theta) \right)^{\frac{1}{\epsilon}} - (\frac{\pi}{2} - \theta) \tag{3.18}$$

Sampling by the implicit equation [eq.3.6] results in loss of points on the surface and

clutter of points on the interim, while the [102] method generates a properly distributed sampling on the whole superquadric. The comparison is shown in fig.3.5.1, where the in fig(a), it is sampled by the implicit equation and (b) is constructed by Pilu-Fisher method from the same parameters of $a_1 = 0.0101381$, $a_2 = 0.0389216$, $a_3 = 0.0760457$, $\epsilon_1 = 0.1$ and $\epsilon_2 = 0.29569$.

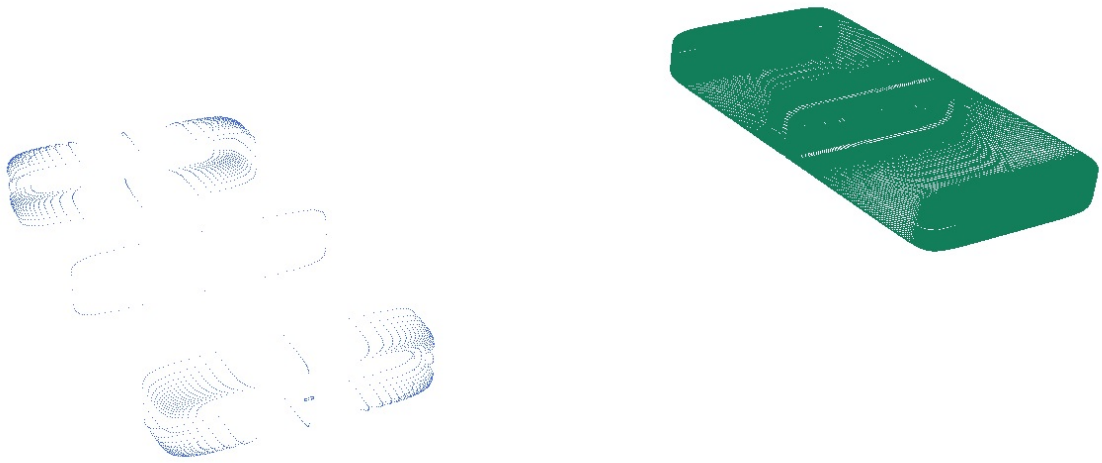


Figure 3.5.1: Superquadric sampling by a) implicit equation 3.6 and b) Pilu-Fisher Method [102]

3.6 Superquadric Deformation

A new kind of superquadric representation can be achieved by introducing deformations to the existing superquadric representations which can only represent simplistic convex and symmetric objects [126]. There are mainly two kinds of deformations can be introduced to superquadrics: a)Tapering and b)Bending.

3.6.1 Tapering

The most common tapering is linear tapering meaning only in z-axis which can achieved by multiplying x and y function with the function of z. The radius of the object progressively increase or decrease in x or y direction. With two new parameters t_x and t_y tapering factors,

the tapering function can be defined as:

$$T(x) = \left\{ \begin{array}{c} (t_x \frac{z}{a_3} + 1)x \\ (t_y \frac{z}{a_3} + 1)y \\ z \end{array} \right\} \tag{3.19}$$

The tapering parameters are restricted in the range of -1 to 1, where -1 and 1 meaning the ends are pointed and 0 means no deformations.

3.6.2 Bending

Bending can be achieved by rotating the xz plane around the z-axis by an angle of α. First the points of the undeformed superquadric are projected onto the bending plane and then bent along a circular section of the plane with a radius of $\frac{1}{k}$, and finally projected back to the original plane. The function for bending can simplified as:

$$B(x) = \left\{ \begin{array}{c} x + \cos\alpha(r_B - r) \\ y + \sin\alpha(r_B - r) \\ \sin(zk)(\frac{1}{k} - 1) \end{array} \right\} \tag{3.20}$$

where

$$r = \cos(\alpha - arc\tan\frac{y}{x})\sqrt{x^2 + y^2} \tag{3.21}$$

$$r_B = \frac{1}{k} - \cos(zk)(\frac{1}{k} - r) \tag{3.22}$$

While $\alpha \in (-\pi, \pi)$, a very small value of k should be provided as $k \neq 0$. The value of k should not be negative as deformation described by (α, k) is equivalent to $(\alpha\pi, k)$ [128].

3.7 Superquadric Fitting

The problem of fitting superquadric can be defined as the problem of finding a superellipsoid that best fits the points, by some quality -of-fit criteria. There are several approaches proposed to fit superellipsoid such as Genetic Algorithms [125], Least-square fitting [12][139], Simmulated Annealing[144] and combination of Genetic Algorithm and Simulated Annealing[125]. The most commonly used method is Least-square fitting method as in terms of computing time, it is less expensive. The goal is find the set of 11 parameters $a \ni \{a_1,$

a_2, a_3, ϵ_1, ϵ_2, p_x, p_y, p_z, ρ, ψ, θ } that minimizes the total error:

$$E(a) = \frac{1}{N}\sum_{i=1}^{N}\chi^2(a, x_i) \tag{3.23}$$

where N is the number of input points and i-th input point in the model space is x_i. By incorporating an error vector $d := d(a) := (\chi(a, x_i))_{i=1}^{N}$, it can be expressed in vector notation as:

$$E(a) = \frac{d^T d}{N} \tag{3.24}$$

The function χ can be minimized by iterative methods such as Gaussian-Newton, Lavenberg-Marquardt algorithms, or Taubin Method. The fitting process can be distinguished between two approaches:

1. Algebric Fitting: The implicit equation [Eq.3.6] to calculate the error of a point.

2. Geometric Fitting: Considering the euclidean distance of a point to the surface as the fitting error. It can be defined as the distance between x_i and its orthogonal projection x_i' on the surface.

As the interior points are always have distance < 1, the algebric fitting is not a very suitable choice. While geometric fitting is a better approach, it is computationally expensive. As a distance measurement Solina's distance [126], Taubin Distance [130], or Radial Euclidean Distance can be considered while Radial Euclidean Distance is the most suitable choice.

To fit the superquadric model to point cloud data by Radial Euclidean Distance, the distance from the points to the function $f(a, x, y, z)$ must be minimized. Radial distance $|OP|$, which is the distance between the points and the center of the superquadric [62], is used instead of the distance perpendicular to the surface(minimum distance) for each point. Given a point of coordinates $p_0 = (x_0, y_0, z_0)$ in the superellipsoid frame, considering the scalar β such that the tip of the scaled vector $p_s = \beta p_0$, falls on the surface of the superellipsoid. Given the inside-outside function, $F_e(p_s) = 1$, so that,

$$F_e(\beta x_0, \beta y_0, \beta z_0) = \beta^{\frac{2}{\epsilon_1}} F_e(x_0, y_0, z_0) = 1 \tag{3.25}$$

We can calculate the scalar β as:

$$\beta^{-\frac{2}{\epsilon_1}} = F_e(x_0, y_0, z_0) \tag{3.26}$$

The radial distance is then

$$d = |p_0 - p_s| = |p_0||1 - \beta| \tag{3.27}$$

which can represented in its full form as:

$$d(a, x, y, z) = \min \sqrt{a_1 a_2 a_3 \sum_{k=0}^{n} ||OP|| * (f(a, x, y, z)^{\epsilon} - 1)} \tag{3.28}$$

Where a is the superset of a_1, a_2, and a_3. The total number of points in the point cloud is n and d is the overall distance from the cloud to the superellipsoid. By Taubin Method [130], the distance measurement can be performed by gradients:

$$d(a, x, y, z) = \frac{|f(a, x, y, z) - 1|}{|| \bigtriangledown f(a, x, y, z)||}. \tag{3.29}$$

This method is more expensive as it needs the gradient at each step, while it provides more accuracy for points near the surface.

3.8 Supertoroid Representation

A supertoroid is a dough-nut like surface represented by a similar mathematical formulation of the superquadric. The supertoroid also has both implicit and parametric expression as the their quadratic counterparts, and it is included with them as superquadric as an extension. The parametric expression of the supertoroid is given by:

$$r_t(\eta, \omega) = \left\{ \begin{array}{c} a_4 + \cos^{\epsilon_1} \eta \\ a_3 \sin^{\epsilon_1} \eta \end{array} \right\} \otimes \left\{ \begin{array}{c} a_1 \cos^{\epsilon_2} \omega \\ a_2 \sin^{\epsilon_2} \omega \end{array} \right\} = \left\{ \begin{array}{c} a_1(a_4 + \cos^{\epsilon_1} \eta) \cos^{\epsilon_2} \omega \\ a_2(a_4 + \cos^{\epsilon_1} \eta) \sin^{\epsilon_2} \omega \\ a_3 \sin^{\epsilon_1} \eta \end{array} \right\}, \tag{3.30}$$

where in order to complete the four quadrants, we need to take $\cos^{\epsilon_1} \eta = sign(\cos \eta)|\cos \eta|^{\epsilon_1}$, and similarly for the rest of the trigonemetric variables. As in the case of superellipsoid, the real exponent ϵ_2 controls the shape in the canonical $x - y$ plane and the real, position exponent ϵ_1 controls the shape in the perpendicular planes. The coefficients a_i control the dimensions in the x, y, z directions and the size of the hole.

The supertoroid has the implicit equation, in its canonical coordinate system,

$$f_t(a,x,y,z) : \left| \left(\left|\frac{x}{a_1}\right|^{\frac{2}{\epsilon_2}} + \left|\frac{y}{a_2}\right|^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} - a_4 \right|^{\frac{2}{\epsilon_1}} + \left|\frac{z}{a_3}\right|^{\frac{2}{\epsilon_1}} = 1. \tag{3.31}$$

3.9 Formulation of Supertoroid Radius

Consdering only the $x-y$ plane ($z=0$) in the Eq. 3.30 we can obtain:

$$a_3 \sin^{\epsilon_1}\eta = 0, \;\; as \;\; a3 \neq 0 \;\; implying \;\; \eta = 0, \pi \tag{3.32}$$

In the $x-y$ plane,

$$r_t(0,\omega) = \left\{ \begin{array}{c} a_1(a_4+1)\cos^{\epsilon_2}\omega \\ a_2(a_4+1)\sin^{\epsilon_2}\omega \\ 0 \end{array} \right\} \tag{3.33}$$

The supertoid radius $R(\omega) = \sqrt{x^2+y^2}$ is:

$$\begin{array}{l} R^2(0,\omega) = (a_4+1)^2\left(a_1^2\cos^{2\epsilon_2}\omega + a_2^2\sin^{2\epsilon_2}\omega\right) \\ R^2(\pi,\omega) = (a_4-1)^2\left(a_1^2\cos^{2\epsilon_2}\omega + a_2^2\sin^{2\epsilon_2}\omega\right) \end{array} \tag{3.34}$$

The plot of the radius of supertoroid in the $x-y$ plane with $\eta = (0,\pi)$ and $\omega = (0,\pi/2)$ can be visualized in the Fig.3.9.1.

But the width of supertoroid is not constant as visualized in Fig.3.9.3a. Dependence of the width of the supertoroid with w:

$$\begin{aligned} w(\omega) = R(0,\omega) - R(\pi,\omega) &= \big((a_4+1)-(a_4-1)\big)\sqrt{a_1^2\cos^{2\epsilon_2}\omega + a_2^2\sin^{2\epsilon_2}\omega} \\ &= 2\sqrt{a_1^2\cos^{2\epsilon_2}\omega + a_2^2\sin^{2\epsilon_2}\omega} \end{aligned} \tag{3.35}$$

where $w(0) = 2a_1$ in the x-axis and $w(\frac{\pi}{2}) = 2a_2$ in the y-axis. A radius of the mean ellipse should be considered as shown in Fig.3.9.3b.

The radius of the mean ellipse is:

$$R^2(\frac{\pi}{2},\omega) = a_1^2a_4^2\cos^{2\epsilon_2}\omega + a_2^2a_4^2\sin^{2\epsilon_2}\omega + a_3^2 \tag{3.36}$$

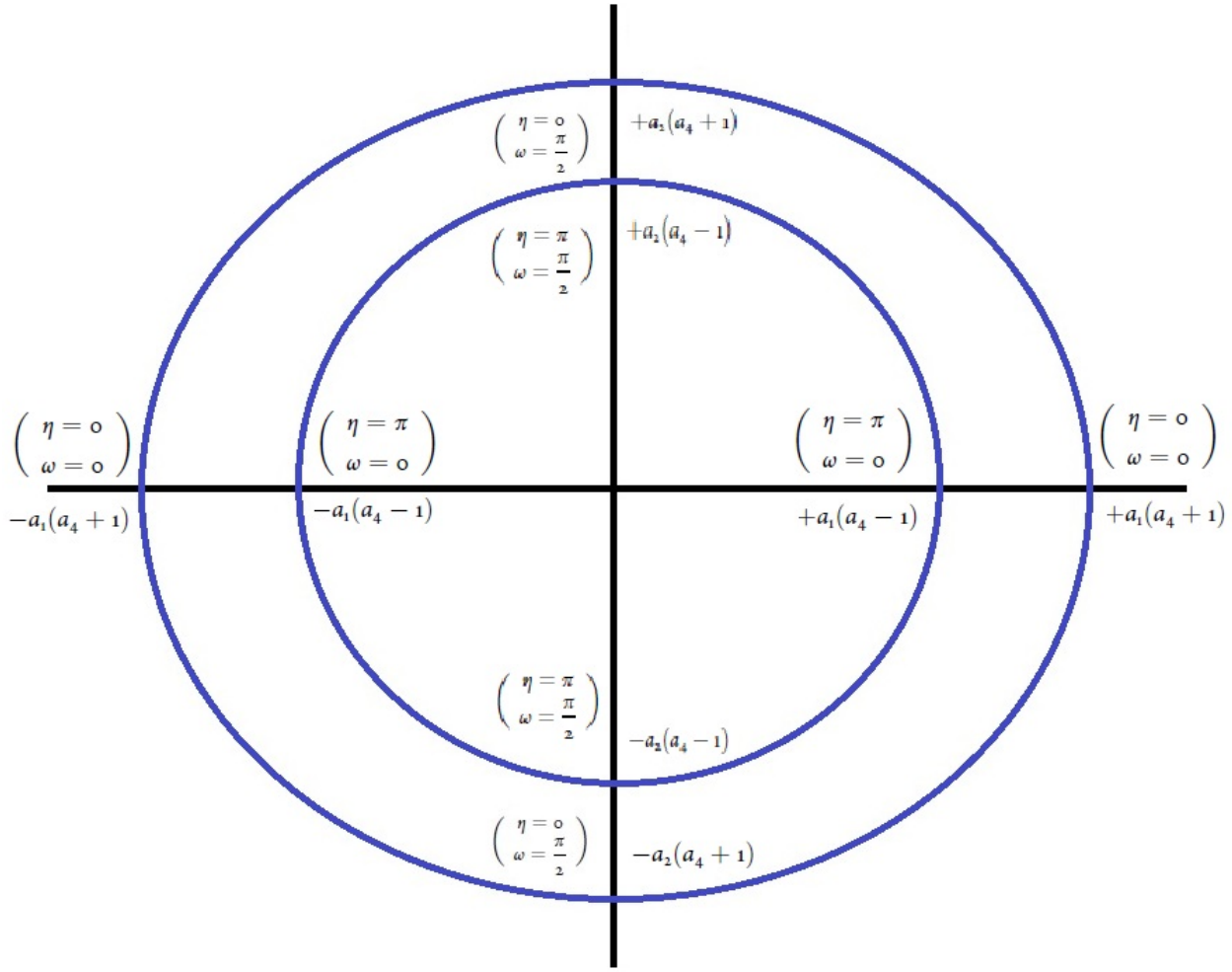


Figure 3.9.1: The radius plot of the projected supertoid ($z = 0$) for the range of $\eta = (0, \pi)$ and $\omega = (0, \pi/2)$

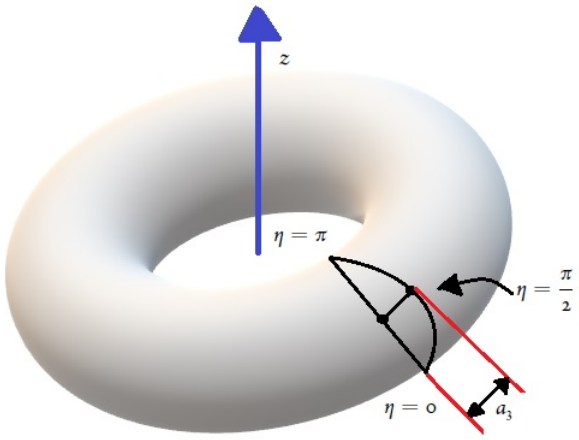


Figure 3.9.2: Radius of the supertoroid with varying η.

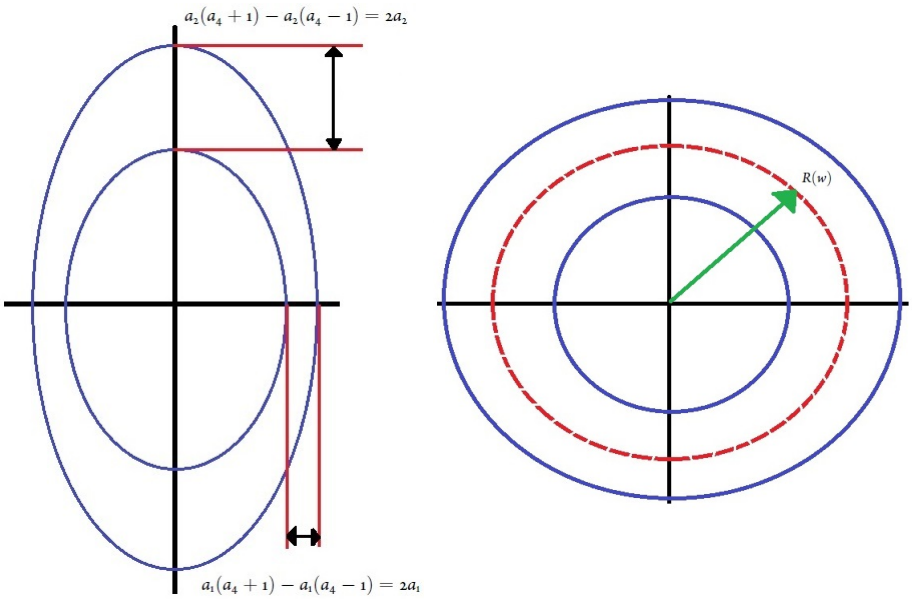


Figure 3.9.3: Radius of the supertoroid. a) Difference in x and y axis, b) Mean radius of the superellipse

$$R(w) = R(\frac{\pi}{2}, \omega)\Big|_{z=0} = a_4\sqrt{a_1^2\cos 2\epsilon_2\omega + a_2^2 \sin 2\epsilon_2\omega} \tag{3.37}$$

where $R(w)$ is the projection in $x-y$ plane.

3.10 Supertoroid Fitting

To fit the supertoroid model to point cloud data, the distance from the points to the function $f(a, x, y, z)$ must be minimized. Given a point of coordinates $p_0 = (x_0, y_0, z_0)$ in the superellipsoid frame, we have to consider two scalars β_1 and β_2, where the $R_\pi = \beta_2 P_\pi$ and $P_e = \beta_1 P_R$. Here R_π is the mean superellipse distance in the $x-y$ plane and P_π is the projection of the point P to be fitted on the $x-y$ plane. P_e is the closest point of the superellipse from R_π and P_R is distance from the closest point to the actual point to be fitted.

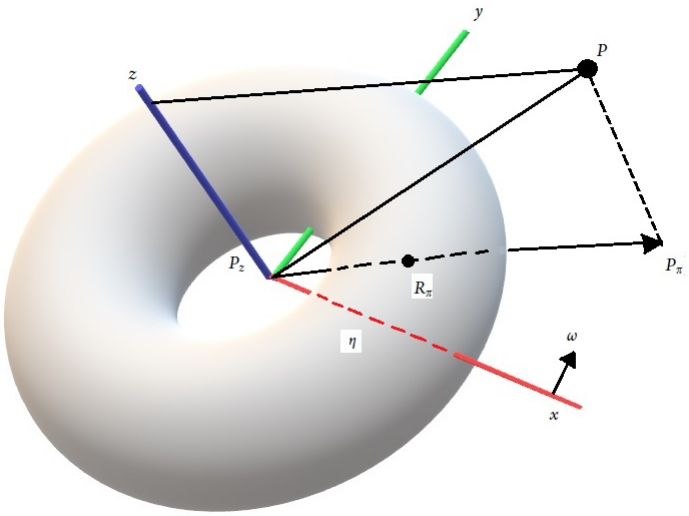


Figure 3.10.1: Calculation of distance β_2

3.10.1 Distance to supertoid β_2

Fig.3.10.1 Summarizes the calculation of β_2 The projection of point P can be calculated as $P_\pi = P - P_z$. The median superellipse defined by R_c:

$$R_c(\omega) = R(\frac{\pi}{2})\big|_{z=0} = \left\{ \begin{array}{c} a_1 a_4 \cos^{\epsilon_2} \omega \\ a_2 a_4 \sin^{\epsilon_2} \omega \end{array} \right\} \tag{3.38}$$

Assuming $a_4 > 0$ in the implicit equation of the supertorid Eq.3.31:

$$F_c\left(\left\{ \begin{array}{c} x \\ y \end{array} \right\} \right) = \left| \frac{x}{a_1 a_4} \right|^{\frac{2}{\epsilon_2}} + \left| \frac{y}{a_2 a_4} \right|^{\frac{2}{\epsilon_2}} = 1 \tag{3.39}$$

It is known that $F_c(R_c) = 1$ as R_c defines the superellipse. Then

$$F_c(R_c) = F_c\left(\beta_2 \left\{ \begin{array}{c} x_\pi \\ y_\pi \end{array} \right\} \right) = \left| \frac{\beta_2 x_\pi}{a_1 a_4} \right|^{\frac{2}{\epsilon_2}} + \left| \frac{\beta_2 y_\pi}{a_2 a_4} \right|^{\frac{2}{\epsilon_2}} = 1 \tag{3.40}$$

If $\beta_2 \geq 0$,

$$\beta_2^{\frac{2}{\epsilon_2}} \left(\left| \frac{x_\pi}{a_1 a_4} \right|^{\frac{2}{\epsilon_2}} + \left| \frac{y_\pi}{a_2 a_4} \right|^{\frac{2}{\epsilon_2}} \right) = \beta_2^{\frac{2}{\epsilon_2}} F_c(P_\pi) = 1 \tag{3.41}$$

So, β_2 can be defined as:

$$\beta_2^{-\frac{2}{\epsilon_2}} = F_c(P_\pi) \tag{3.42}$$

3.10.2 Distance to supertoid β_1

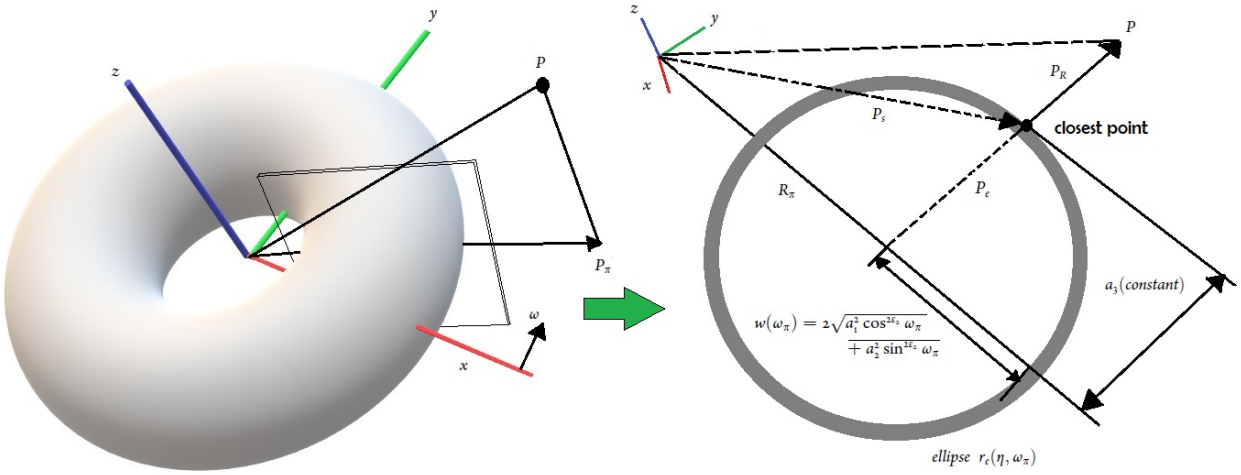


Figure 3.10.2: Calculation of distance β_1

The closest point to the surface is P_s where

$$P_s = R_\pi + P_e = \beta_2 P_\pi + \beta_1 (P - \beta_2 P_\pi) \tag{3.43}$$

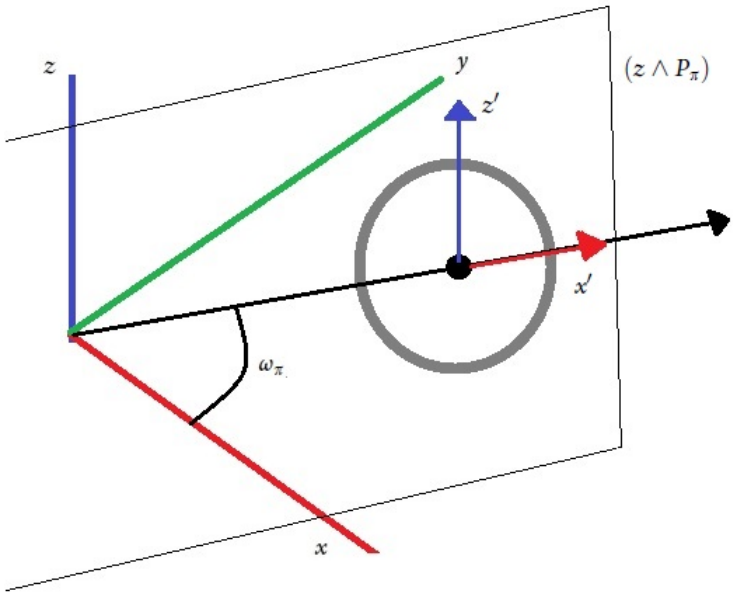


Figure 3.10.3: superellipse in $x' - z'$ plane

In the $x' - z'$ plane ($z \wedge P_\pi$) as shown in Fig.3.10.3 :

$$r'_e(\eta, \omega_\pi) = \left\{ \begin{array}{c} 2\sqrt{a_1^2 \cos^{2\epsilon_2} \omega_\pi + a_2^2 \sin^{2\epsilon_2} \omega_\pi} \cos^{\epsilon_1} \eta \\ 0 \\ a_3 \sin^{\epsilon_1} \eta \end{array} \right\} = \left\{ \begin{array}{c} a_\pi \cos^{\epsilon_1} \eta \\ 0 \\ a_3 \sin^{\epsilon_1} \eta \end{array} \right\} \tag{3.44}$$

considering $a_\pi = 2\sqrt{a_1^2 \cos^{2\epsilon_2} \omega_\pi + a_2^2 \sin^{2\epsilon_2} \omega_\pi}$. Transforming to the original $x - y - z$ axis, it is just a rotation in $z - axis$:

$$r_e(\eta, \omega_\pi) = \begin{bmatrix} \cos\omega_\pi & -\sin\omega_\pi & 0 \\ \sin\omega_\pi & \cos\omega_\pi & 0 \\ 0 & 0 & 1 \end{bmatrix} r'_e(\eta, \omega_\pi) \tag{3.45}$$

Using the superellipse in $x' - z'$:

$$F_e\left(\left\{ \begin{array}{c} x' \\ 0 \\ z' \end{array} \right\} \right) = \left| \frac{x'}{a_\pi} \right|^{\frac{2}{\epsilon_1}} + \left| \frac{z'}{a_3} \right|^{\frac{2}{\epsilon_1}} = 1 \tag{3.46}$$

where

$$\left\{ \begin{array}{c} x' \\ y' \\ z' \end{array} \right\} = \begin{bmatrix} \cos\omega_\pi & \sin\omega_\pi & 0 \\ -\sin\omega_\pi & \cos\omega_\pi & 0 \\ 0 & 0 & 1 \end{bmatrix} \left\{ \begin{array}{c} x \\ y \\ z \end{array} \right\} = \left\{ \begin{array}{c} x\cos\omega_\pi + y\sin\omega_\pi \\ -x\cos\omega_\pi + y\sin\omega_\pi \\ z \end{array} \right\} \tag{3.47}$$

Then

$$F_e\left(\left\{ \begin{array}{c} x \\ y \\ z \end{array} \right\} \right) = \left| \frac{x\cos\omega_\pi + y\sin\omega_\pi}{a_\pi} \right|^{\frac{2}{\epsilon_1}} + \left| \frac{z}{a_3} \right|^{\frac{2}{\epsilon_1}} = 1 \tag{3.48}$$

$F_e(P_e) = 1$, as in the superellipse-centric frame, P_e belongs to the quadric, hence:

$$F_e(\beta_1, P_R) = F_e\left(\left\{ \begin{array}{c} \beta_1(1-\beta_2)P_x \\ \beta_1(1-\beta_2)P_y \\ \beta_1 P_z \end{array} \right\} \right) \tag{3.49}$$

$$P_R = P - R_\pi = P - \beta_2 P_\pi = \left\{ \begin{array}{c} \beta_1(1-\beta_2)P_x \\ \beta_1(1-\beta_2)P_y \\ \beta_1 P_z \end{array} \right\} \tag{3.50}$$

$$\left|\frac{\beta_1(1-\beta_2)P_x\cos\omega_\pi + \beta_1(1-\beta_2)P_y\sin\omega_\pi}{a_\pi}\right|^{\frac{2}{\epsilon_1}} + \left|\frac{\beta_1 P_z}{a_3}\right|^{\frac{2}{\epsilon_1}} = \beta_1^{\frac{2}{\epsilon_1}} F_e(P_R) = 1 \tag{3.51}$$

So β_1 can be defined as:

$$\beta_1^{-\frac{2}{\epsilon_1}} = F_e(P_R) \tag{3.52}$$

The final inside-outside function $F_e(P_e)$ is:

$$F_e(P_e) = \beta_1^{-\frac{2}{\epsilon_1}} \left|\frac{(1-\beta_2)P_x^2}{\sqrt{P_x^2+P_y^2}} + \frac{(1-\beta_2)P_y^2}{\sqrt{P_x^2+P_y^2}}\right|^{\frac{2}{\epsilon_1}} + \left|\frac{P_z}{a_3}\right|^{\frac{2}{\epsilon_1}} = 1 \tag{3.53}$$

$$= \beta_1^{-\frac{2}{\epsilon_1}} \left((1-\beta_2)^{\frac{2}{\epsilon_1}} \left| \frac{\frac{P_x^2+P_y^2}{\sqrt{P_x^2+P_y^2}}}{a_\pi} \right|^{\frac{2}{\epsilon_1}} + \left|\frac{P_z}{a_3}\right|^{\frac{2}{\epsilon_1}} \right) = 1 \tag{3.54}$$

3.11 Pose Estimation

To estimate the pose of the superquadric in the sensor frame the most common method is Principal Component Analysis(PCA). For determining the axes of the supertoroid, initial value of $\epsilon_1 = \epsilon_2 = 1$ of an superellipsoid can be considered. The position parameters (p_x, p_y, p_z) can be estimated so that X_0 is the barycenter of the input points:

$$\mathrm{X}_0 = \bar{\mathrm{X}} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{X}_i \tag{3.55}$$

For a set of input points X, we can define

$$C = \frac{1}{N}\sum_{i=1}^{N} (\mathrm{X}_i - \bar{\mathrm{X}})(\mathrm{X}_i - \bar{\mathrm{X}})^T \tag{3.56}$$

C is the covariance matrix which is symmetric. The eigenvectors of C can be defined as the three axes of the inertia of the points. To obtain the solution, Jacobian Transformation or Singular Value Decomposition [140]can be used. The goal is to align the main axis of inertia with the z-axis of the superellipsoid. Sorting according to increasing eigenvalues, the e_1, e_2, e_3 are the axes of the superellipsoid in the x, y and z axes, moreover in the coordinate system of the superellipsoid. The matrix of the eigenvectors represents the rotation matrix

of the superellipsoid. The rotation angles ρ, ψ, θ can be obtained as:

$$\rho = arc\tan\frac{e_{2,3}}{e_{1,3}} \tag{3.57}$$

$$\psi = arc\cos e_{3,3} \tag{3.58}$$

$$\theta = arc\tan\frac{e_{3,2}}{-e_{3,1}} \tag{3.59}$$

where $e_{i,j}$ is the j-th component of the i-th eigenvector.

Another method of estimating the pose in [5] where it is estimated the objects are standing on the support surface, e.g table. The poses are only altered in terms of rotation in only one axis. Instead of finding all the 6 parameters (p_x, p_y, p_z,ρ, ψ, θ), it only searches for 4 parameters (p_x, p_y, p_z,θ).

The center of the object or the position of the object (p_x, p_y, p_z) is estimated by taking the mean of the points in all three dimensions. For completely unknown objects, the ground truth orientation is also unknown, so an assumption is asserted that the orientation of all the objects perceived are z-axis upwards. The problem of obtaining pose information (ρ, ψ, θ) is now limited to finding only θ which controls the rotation in z-axis. The volume V_{cloud} is calculated by maximum distances in each dimension(d_x, d_y, d_z), where the maximum distance is the difference between the maximum value and minimum value in each plane. All the points in the point cloud are projected onto the table plane and a 2D convex hull is created around the projected cloud. The volume of a bounding box can also be estimated from length l, width w of the convex hull and h, height as the value of p_{Zmax}; the maximum height. By rotating the convex hull in small iterations and minimizing the error (ϵ) between the volume of the bounding box and volume of the point cloud, the rotation θ in z-axis can be obtained.

$$\epsilon = min(V_{cloud} - [l * d * h]) \tag{3.60}$$

Figure 3.11.1, illustrates the procedure of finding pose of the superquadric from the point cloud data by iteration method. After finding the dimensions of the bounding box, the box is projected to 2D plane and rotated by small iterations.

3.12 Summary

In this chapter, the superellispdoid representations are formulated. The implicit and parametric equations for the superquaric are presented with the notion of superquadric sampling.

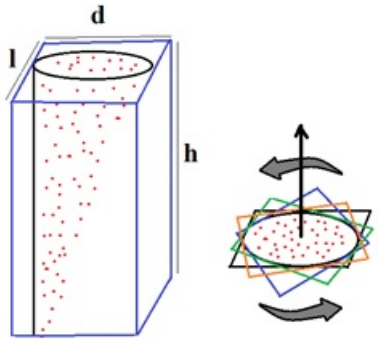


Figure 3.11.1: Pose Estimation by Iteration

The process of fitting superquadric to point cloud data is presented. Additionally the equations for supertoroid models are formulated. Finally a method of pose estimation by iteration is presented.

The robotic industry's ongoing advancements depend upon state-of-the-art equipments- with unique specifications and capabilities. As the hardware of the robotic systems is developed, so as the softwares are metamorphosed. From the perspective of an industry or household user, however, the main goal of deploying a robotic system remains the same: to successfully perform the desired task. Tasks can range from intense industrial work such as welding, packaging, and manipulation to relatively smaller scale of picking, placing or grasping. For the users to accept and employ a robotic system, it must, (1) precisely execute the task, and (2) integrate well into the user's workspace. For both features, the user and the robot should be aware of information about the robot's reach and the workspace. Without this knowledge, deployment of the robot depends solely on user intuition. An incorrect intuition can lead to human casualty or catastrophic damage to the robotic system and workplace.

Non-expert users who do not have an in-depth understanding of robot kinematics and the reachability challenges of robots might hold misconceptions that the workspace of a robotic manipulator is sphere-shaped when the radius of the arm is fully extended and in this region, the robot's end-effector can move freely. On the contrary, the robotic arm's workspace depends entirely on the constraints posed on its arm joints. In the workspace, there are few sections where the arm can reach freely; in some parts of the workspace, the arm faces singularity. While this reachability information is advantageous in extended applications such as finding a suitable base position for a robotic task, it can be very helpful in filtering grasp poses and searching for most feasible grasp in the object's grasp space.

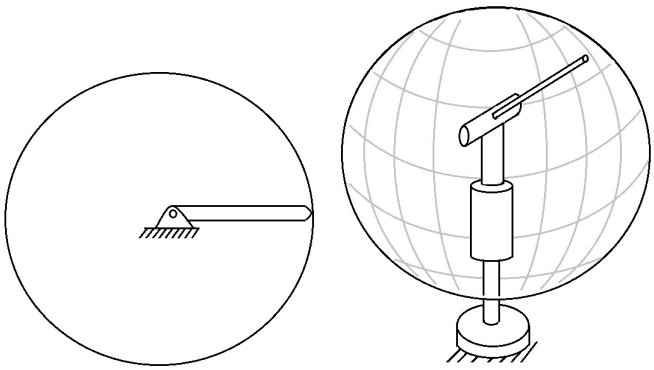


Figure 4.0.1: From top to bottom, left to right. a) Cartesian workspace of 1DOF robot, b)Workspace of a 2DOF robot where the joint axes are perpendicular

4.1 Workspace of a Robot

The workspace is a set of configurations that the end-effector of the robot can achieve depending on the structure of the robot. By definition of the workspace, every point in the workspace of the robot should be reachable by at least one configuration or multiple configurations of the robot[80]. Adding more joints than the DOF of the workspace point poses redundancy to the system. Generally, the points of the reachable workspace are points on cartesian space which the robot's end-effector can reach. Combination of these points generates the structure of the workspace. As for example, if the robot has a single revolute joint or the robot is a 1DOF(Degree of Freedom) robot, the workspace of the robot will be a planar disk or a surface as shown in Fig.4.0.1a. Adding another joint perpendicular to axis transforms the robot to 2DOF robot and the workspace will be a sphere as demonstrated in Fig.4.0.1b.

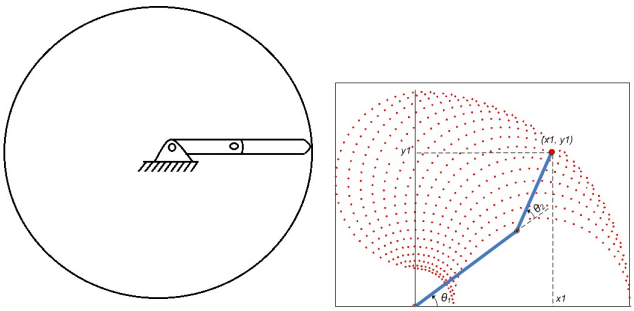


Figure 4.1.1: From top to bottom, left to right. a) Cartesian workspace of 2DOF robot where the joint axes are parallel, b)Workspace plot of the 2DOF robot with joint limits and self-collision

Adding another revolute joint parallel to the axis of the first joint should maintain the planar disk workspace of the robot, as displayed in Fig.4.1.1a. But in reality, due to joint limits and self-collision, the structure is limited as shown in 4.1.1b for the 2R robot. The limitation in the structure starts increasing with increase in joints. The increased complexity of workspace with the inclusion of joints is demonstrated in Fig. 4.1.2 for a 7DOF robot and a 6DOF robot.

4.1.1 From configurations to Workspace points

Workspace points from robot configurations can be achieved by Forward Kinematics (FK) which finds the position (p_x, p_y, p_z) and orientation (ρ, ψ, θ) of end-effector provided a configuration (q) or value of individual joints $(q_1, q_2, \ldots q_n)$ of the manipulator, where the robot has a n-DOF manipulator [82]. The simple assumption is that all joints have a single degree

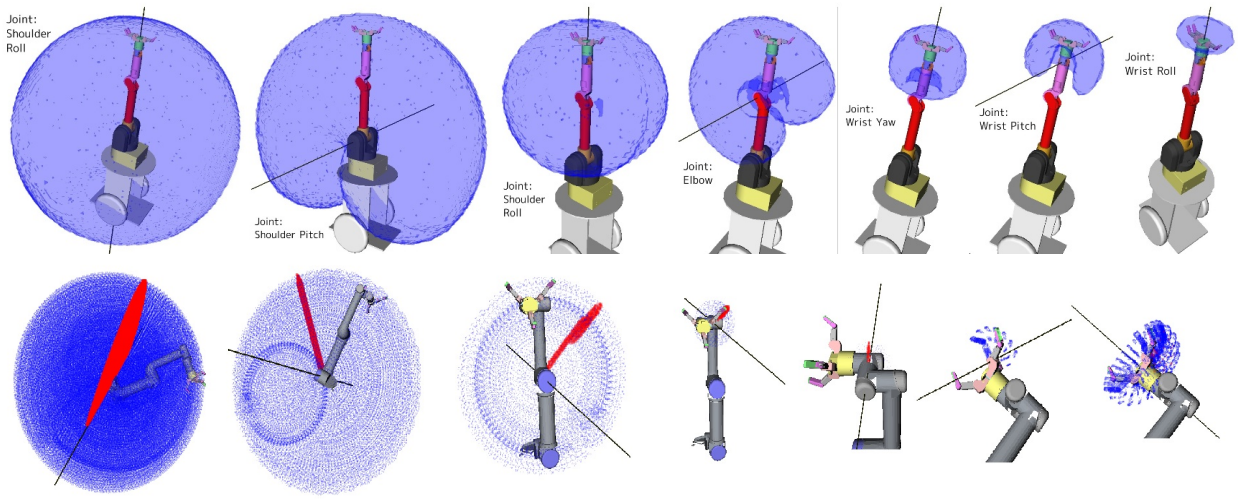


Figure 4.1.2: From top to bottom. a) Workspace of individual joints for 7DOF WAM arm, b) Workspace of individual joints for 6DOF UR5 arm with Barrett gripper

of freedom and the actions of each joint can be represented by a single number; angle of rotation for revolute joint and displacement for prismatic joints. We can associate a joint variable depending on the type of joint, such as for a joint i, the if it is revolute, it can be denoted by q_i and if it is prismatic, it can be defined as d_i. A coordinate frame is assigned to each link of the robot as shown in Fig.4.1.3, where blue represents z-axis, green is y-axis and x-axis is represented by red. Frame $O_0x_0y_0z_0$ is assigned to the first joint considered as the base frame. When a joint is actuated, the links after the joints and the coordinate frames obtain a resulting motion. Suppose A_i is the homogeneous transformation that provides the position and orientation of frame $O_ix_iy_iz_i$ with respect to the frame $O_{i-1}x_{i-1}y_{i-1}z_{i-1}$. We can say that

$$A_i = A_i(q_i) \tag{4.1}$$

The homogenerous transformation that expresses the position and orientation of $O_jx_jy_jz_j$ with respect to $O_ix_iy_iz_i$

$$T_j^i = \begin{cases} A_{i+1}A_{i+2}\cdots A_{j-1}A_j, & if \;\; i < j \\ I, & if \;\; i = j \\ (T_i^j)^{-1}, & if \;\; i > j \end{cases} \tag{4.2}$$

where T_j^i is called the transformation matrix.

If the position and orientation of the end-effector with respect to the base frame are d_n^O

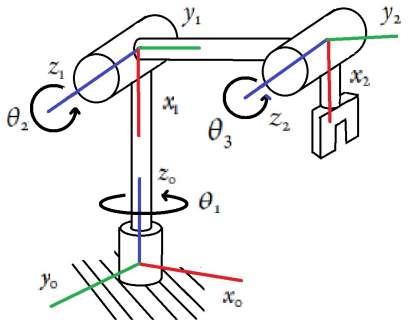


Figure 4.1.3: Coordinate frames assigned to a 3R robot.

and R_n^O, the it can be represented in terms of homogeneous transformation as

$$T_n^O = A_1(q)A_2(q)\cdots A_{n-1}(q_{n-1})A_n(q_n) = \begin{bmatrix} R_n^O & d_n^O \\ 0 & 1 \end{bmatrix} \tag{4.3}$$

Where

$$A_i(q_i) = \begin{bmatrix} R_i^{i-1} & d_i^{i-1} \\ 0 & 1 \end{bmatrix} \tag{4.4}$$

The final transformation matrix is

$$T_j^i = A_{i+1}A_{i+2}\cdots A_{j-1}A_j = \begin{bmatrix} R_j^i & d_j^i \\ 0 & 1 \end{bmatrix} \tag{4.5}$$

with

$$R_j^i = R_{i+1}^i \cdots R_j^{j-1} \;\; and \;\; d_j^i = d_{j-1}^i + R_{j-1}^i d_{j-1}^j \tag{4.6}$$

Expressing forward kinematics eqaution in the product of exponential form:

$$T(\theta) = e^{|S_1|\theta_1} e^{|S_2|\theta_2} \ldots e^{|S_n|\theta_n} M \tag{4.7}$$

where the robot is spatial n-DOF open chain. The forward kinematics equation can also be represented in terms of Denavit-Hertenberg parameters [82].

4.1.2 From Workspace points to Configurations

Searching for a joint solution in terms of end-effector poses is the problem of finding inverse kinematics (IK) solution [82]; the inverse problem of forward kinematics problem. The general problem can be stated as:

Given a $4x4$ homogeneous transformation

$$H = \begin{bmatrix} R_{3\times3} & d_{3\times1} \\ 0 & 1 \end{bmatrix} \in SE(3) \tag{4.8}$$

with $R \in SO(3)$, we have to find one or all the solution of equations

$$T_n^O(q_1, \cdots q_n) = H \tag{4.9}$$

where

$$T_n^O(q_1, \cdots q_n) = A_1(q_1) \cdots A_n(q_n) \tag{4.10}$$

The inverse kinematics problem can be solved in two ways. a) Numerical Inverse Kinematics and, b) Analytic Inverse Kinematics

Numerical Inverse Kinematics

There are several attempts to find inverse kinematics solutions by Numerical methods which revolve around gradient descent using $\frac{\partial T}{\partial q}$. The goal is to find the roots of the nonlinear equation. An approach towards solving the non-linear equation can be Newton-Raphson method. Still, different optimization methods may be needed where exact solutions do not exist, and a closest approximate solution should be incorporated. To find solutions by Newton-Raphson method for an equation $g(\theta) = 0$ with a differentiable function $g : \mathbb{R} \to \mathbb{R}$, an initial guess θ_0 should be considered. The Taylor series expansion for $g(\theta)$:

$$g(\theta) = g(\theta_0) + \frac{\partial g}{\partial \theta}[(\theta_0)(\theta - \theta_0)] + \frac{\partial^2 g}{\partial \theta^2}[(\theta_0)(\theta - \theta_0)] + \cdots \tag{4.11}$$

Taking only the first-order terms, setting $g(\theta) = 0$, and solving for θ:

$$\theta = \theta_0 - \left(\frac{\partial g}{\partial \theta}(\theta_0)\right)^{-1} g(\theta_0) \tag{4.12}$$

Taking this solution as the initial guess for the solution and repeating finding the solution for θ, we can obtain the iteration:

$$\theta^{k+1} = \theta^k - \left(\frac{\partial g}{\partial \theta}(\theta^k)\right)^{-1} g(\theta^k) \tag{4.13}$$

Until some stopping criteria or some user specified threshold such as $|g(\theta^k) - g(\theta^{k+1})|/|g(\theta^k)| \leq \epsilon$, the iteration process is repeated, where ϵ is an user-defined value . The process is similar

for multidimensional equation, where $g : \mathbb{R}^n \to \mathbb{R}^n$

$$\frac{\partial g}{\partial \theta}(\theta) = \begin{bmatrix} \frac{\partial g_1}{\partial \theta_1}(\theta) & \dots & \frac{\partial g_1}{\partial \theta_n}(\theta) \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial \theta_1}(\theta) & \dots & \frac{\partial g_n}{\partial \theta_n}(\theta) \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{4.14}$$

Expressing the end-effector frame using the coordinate vector x by the forward kinematics equation $x = f(\theta)$ by a multidimensional non-linear equation as $f : \mathbb{R}^n \to \mathbb{R}^m$ which maps the n dimensional joint coordinates to m dimensional end-effector coordinates, we can solve for $g(\theta)$ by Newton-Raphson method. The equation can be defined as

$$g(\theta_d) = x_d - f(\theta_d) = 0 \tag{4.15}$$

Taking θ_0 as the initial guess, the equation can be expressed as Taylor series expansion to the first-order terms as:

$$x_d = f(\theta_d) = f(\theta_0) + \left.\frac{\partial f}{\partial \theta}\right|_{\theta_0} \triangle\theta \tag{4.16}$$

Where $\triangle\theta = (\theta_d - \theta_0)$ and the differentaible part is called Jacobian $J(\theta) \in \mathbb{R}^{m \times n}$. The equation can be further approximated as:

$$J(\theta_0)\triangle\theta = x_d - f(\theta_0) \tag{4.17}$$

Assuming $J(\theta^0)$ is invertible and square, the $\triangle\theta$ can be solved as

$$\triangle\theta = J^{-1}(\theta_0)(x_d - f(\theta_0)) \tag{4.18}$$

If the J is not square or singular or invertible, a close solution can still be obtained by Moore-Penros pseudoInverse $J^\dagger$.

Analytical Inverse Kinematics

Analytic Inverse Kinematics solvers are faster than numerical solvers and present almost exact solutions. Also, the analytic solution is completely dependent on the structure of the robot. There are many approaches towards solving inverse kinematics by analytical methods. The most general one is to convert the problem into a higher-dimensional univariate polynomial. Another approach is to analyze the structure of solution sets of polynomial solutions [138]. The most effective method is to formulate the problem as eignedecomposition problem [112] [81]. In [36], a minimal, numerically stable analytical inverse kinematics solver

is presented which proposes an algorithmic-search based approach.

The final transformation of the end-effector can be further simplified as:

$$T_{ee} = T_0 J_0 T_1 J_1 T_2 J_2 \dots T_n = \begin{bmatrix} r_0 0 & r_0 1 & r_0 2 & p_x \\ r_1 0 & r_1 1 & r_1 2 & p_y \\ r_2 0 & r_2 1 & r_2 2 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.19}$$

where T_i is the constant affine transformations and J_i is the transformation depending on the joint value j_i, which may be a rotation around an arbitary axis v_i:

$$J_i(j_i) = I_{4\times4} + \sin j_i \begin{bmatrix} 0 & -v_{i,z} & v_{i,y} & 0 \\ v_{i,z} & 0 & -v_{i,x} & 0 \\ -v_{i,y} & v_{i,x} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + (1 - \cos j_i) \begin{bmatrix} 0 & -v_{i,z} & v_{i,y} & 0 \\ v_{i,z} & 0 & -v_{i,x} & 0 \\ -v_{i,y} & v_{i,x} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}^2 \tag{4.20}$$

or a translation along an axis:

$$J_i(j_i) = \begin{bmatrix} I_{3\times3} & j_i v_i \\ 0 & 1 \end{bmatrix} \tag{4.21}$$

All constraints for solving the inverse kinematics can be derived by changing frame of reference:

$$\begin{aligned} T_0^{-1} T_{ee} T_n^{-1} &= J_0 T_1 J_1 T_2 J_2 \dots J_{n-1} && (i = 0) \\ J_0^{-1} T_0^{-1} T_{ee} T_n^{-1} &= T_1 J_1 T_2 J_2 T3 \dots J_{n-1} && (i = 1) \\ T_1^{-1} J_0^{-1} T_0^{-1} T_{ee} T_n^{-1} &= J_1 T_2 J_2 T3 \dots J_{n-1} && (i = 2) \\ \dots T_{n-1}^{-1} J_{n-2}^{-1} \dots T_0^{-1} T_{ee} T_n^{-1} &= J_{n-1} && (i = 2n-2) \end{aligned} \tag{4.22}$$

For each of the equations $T_{ee,i} = \begin{bmatrix} R_{ee,i} & t_{ee,i} \\ 0 & 1 \end{bmatrix}$ and $T_i = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix}$ such that $T_{ee,i} = T_i$. Including the constraints, the full equation becomes:

$$\sum_i a_i \prod_j c_j^{p_{i,j}} s_j^{q_{i,j}} j_i^{r_{i,j}} = 0 \tag{4.23}$$

where $c_j = \cos(j)$ and $s_j = \sin(j)$ and $p_{i,j} + q_{i,j} + r_{i,j} \leq 1$. Putting $c_j^2 + s_j^2 = 1$ and other trigonomtric variables, the system can consists upto $2n$ variables [36].

4.1.3 Workspace in terms of Reachability Map

While the workspace fully defines all the Cartesian positions and orientations the robot can reach, without parameterization, it is not thoroughly useful. In reality, it can be infinite. Therefore to quantify the workspace, a mapping is needed which can maintain a specialized structure to store all the task poses the robot can reach and the robot configurations associated with it. The structure should also present a notion of areas where the manipulator is most reachable and where it is less reachable or unreachable. It should be capable of providing quick results of simple queries such as which poses are in the most reachable area from a pool of multiple poses or which grasps have the most feasibility in terms of reachability. Reachability map is a parameterization of the workspace of the robot.

4.2 Previous Work

One of the challenges in developing a useful robotic system is designing a platform or workspace that lets the robot to effectively complete the desired task. Initial work on path planning and reachability were typically performed for simple grasp points on objects [90], which involves creating a map representing the areas of high dexterity for the manipulators. This work was then extended to the use of reachability maps to solve the inverse reachability task [23] [36], where the optimal base placement was found to perform the desired grasp on an object. Work done by [146] examined ways to simplify the reachability map by generating a capability map, a simplified structure that permits faster and more efficient searches on the map in order to solve the inverse reachability problem.

These methods are limited. They solve only the general problem of whether or not an inverse kinematics solution was found. A given grasp location could mean that some or all of the solutions found could be non-optimal (near singularities or joint limits). The work of [145] presents the concept of manipulability ellipsoid measure, which can be derived from a Jacobian matrix of manipulators. The size of the ellipsoid and the principal axes represent the manipulation ability in a certain configuration and are thus used to determine the effectiveness of a grasp at a given location. This work was extended regrading grasping and manipulation in [135].

To overcome the single grasp location issue, various methods have been used to search the reachability map in order to find the location where the desired trajectory can be executed with the highest level of dexterity. Work proposed in [134] uses sampling of the trajectory to find and overlay multiple base placements, while [147] uses a pattern search to fit the trajectory into the area representing the field of high dexterity. Proposed method in [148] uses a cross-correlation technique to fit the desired trajectory to the model of the robot

reachability map. Further improvements to the reachability map method were developed by [37] to include the ability to add a transform offset from the original end-effector location, which is useful if the robot is grasping a tool with a non-zero length. Method of [106]explores the case where a specific trajectory is not given; rather the task has been simplified into a generalized workspace environment where the robot must work.

Algorithm 1 Generate Reachability Map

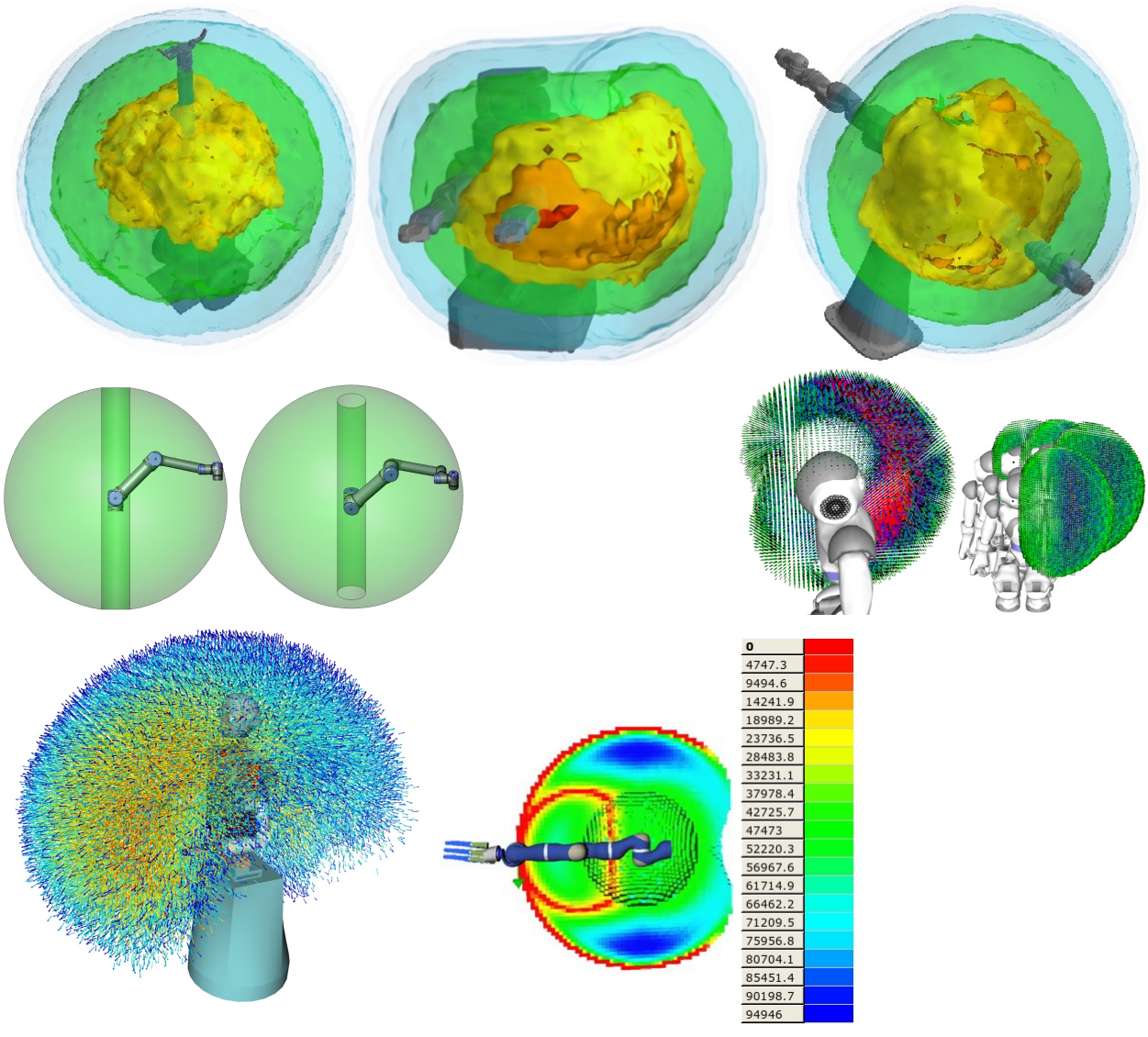
input (URDF of robot)

```
procedure GenerateReachMap
create VoxelStructure V
    for each voxel v_i in V  do
        create sphere s
        checkforSelf Collision(s)
        Store s_filtered in S
    end for
    for each s_i in S  do
        Sample surface and generate poses P
        for each j in P  do
            findIKSolution (P_j)
            if solution  then
                Store (s_i, P_j)
            end if
            d_i = FindReachabilityMeasure(s_i)
            Store d_i with (s_i, P_j, q_j) in map
        end for
    end for
end procedure
```

4.3 Reachability Map Generation

A reachability map is a collection of all poses that the robot's end effector can reach. To accommodate the abundant number of reachable poses, similar poses are clustered into a sphere structure suitable for visualization and easy to access. The structure also captures

Figure 4.2.1 *(following page)*: Different reachability map representations. From top to bottom: a)Reachability Space in OpenRave [36], b)Default workspace visualization in UR5 manual, c) Grasping objects by reachability map [90], d)Reachability map in[135], e)Reachability map representation in [146]


0
4747.3
9494.6
14241.9
18989.2
23736.5
28483.8
33231.1
37978.4
42725.7
47473
52220.3
56967.6
61714.9
66462.2
71209.5
75956.8
80704.1
85451.4
90198.7
94946

the directional information of reachable poses, because several positions can be reached only from a certain direction. An individual sphere is represented as a multimap data structure, which holds information about the sphere's position, all reachable poses belonging to the sphere, the robot configurations leading to the poses, and the reachability measure of that sphere. The robot's workspace is first discretized, sampled for reachable poses and the information of reachable poses are stored in a structured manner [6]. The algorithm for constructing reachability map is presented in Algorithm. 1. The reachability map structure of four different robots created with the Reuleaux library is shown in Fig.4.3.5.

4.3.1 Workspace voxelization

The input to the creation of a robot reachability map procedure is the Unified Robot Description Format (URDF) model of the robot, from which a detailed robot description can be obtained. The robot's hypothesized workspace is first voxelized to create a square structure around the robot. The voxelization process is performed by octree [54], a hierarchical data structure enabling spatial partitioning and searching between voxels. The root node of the octree is at the base of the manipulator, and every node is connected to its eight children. The tree is extended to the overestimation of the diameter of the robot arm in an extended state. The size of the voxels required is task dependent and thus user-defined. Tasks requiring a high degree of accuracy will need a smaller voxel size to provide more accurate results in the final base placement location. The centers of voxels are determined, and a sphere with a radius of the voxel's resolution is placed in every voxel. The workspace voxelization is presented in Fig.4.3.1. For ease of visualization, very low-resolution voxels are presented. In real work, the voxel resolution should be determined based on the task.

4.3.2 Self-Collision Checking

All the spheres collected by the voxelization procedure do not indicate reachable workspace to be considered. The workspace regions where the robot body is present should not be included in the workspace because end-effectors cannot reach those sections due to collisions. For filtering out such sections, as a preprocessing step, the robot body is modeled as a collection of triangles and checked for collisions with the voxelized sphere center. The sphere centers in collisions with robot body (excluding the arm considered for reachability analysis) are opted out of the workspace structure; they are not further discretized for reachability analysis. The robot links of the body that are filtered from the voxelized workspace are shown in red in Fig.4.3.2.

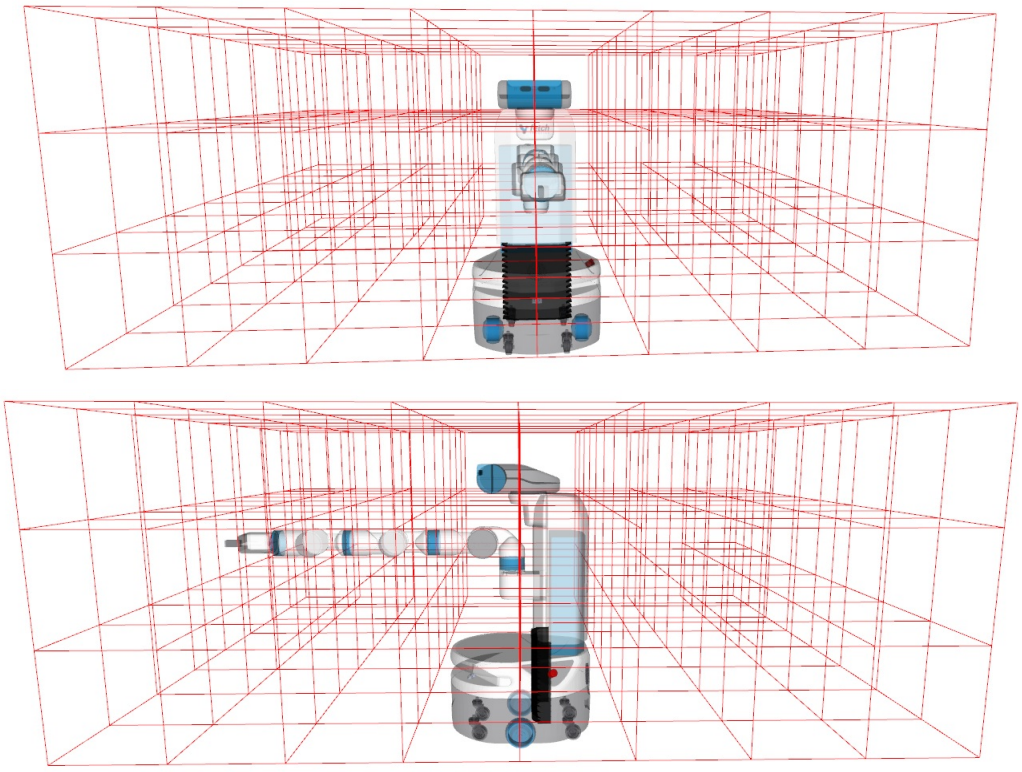


Figure 4.3.1: Voxelization of the workspace of the robot from two point of views a)front and b)side.

A collision checking library FCL [97] is used for fast collision checking. This method provides an advantage in terms of time efficiency since most other methods such as [146] and [36], check for collision only when obtaining solutions from poses. This step drastically reduces the spheres to be searched.

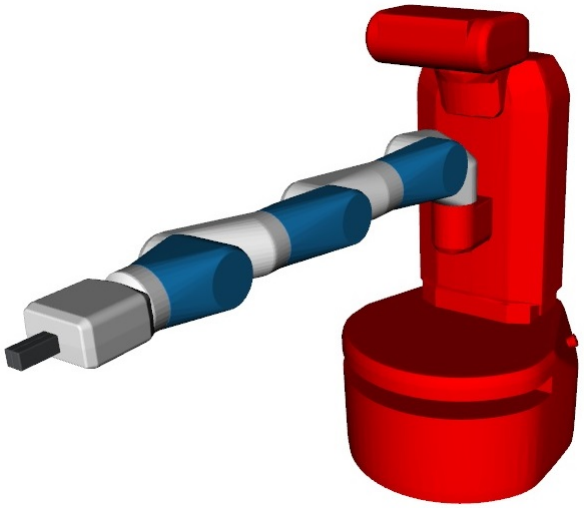


Figure 4.3.2: The voxels on the robot body are filtered. Robot body links are shown in red color.

In the method proposed in [6], self-collision is also included while checking for reachability of the poses. Certain poses may not be inside the robot body, but to reach the pose, some part of the robot may be in collision. Such poses can be considered unreachable. A condition where the robot is in self-collision is represented in Fig.4.3.3, where the links in collision are shown in different colors. It is worth noticing in Fig.4.3.5 that reachability maps created without self-collision represent overall symmetric structures that consider higher reachability at the robot's base, e.g., the reachability map of JACO arm (Fig.4.3.5b). Due to joint limits and self-collision, the area of the robot base should be less reachable, which can be identified in Fig.4.3.5a, the reachability map of the LWR arm. The reachability map of the Fetch and PR2 robot in Fig.4.3.5c and Fig.4.3.5d shows an inconsistent structure since the sphere centers within the robot's body are filtered out in this step.

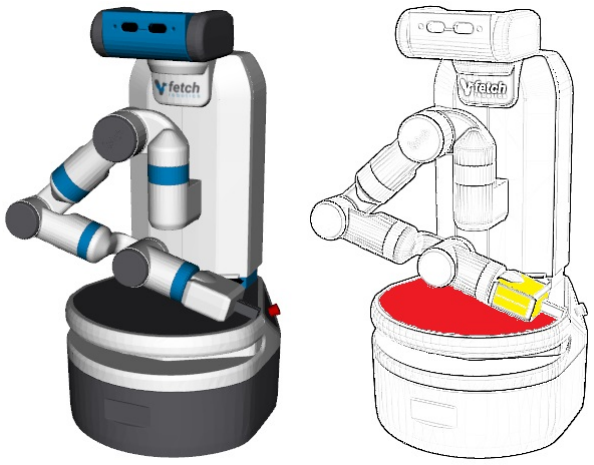


Figure 4.3.3: Self collision between two links of the robot. Colliding links are shown in red and yellow color

4.3.3 Workspace Sampling

One of the most influential ways to determine robot's possible reach position is by forward kinematics (FK), where the joints of the manipulator can be uniformly sampled and the configuration of the tool center point (TCP) can be stored in an efficient structure to represent the reachability workspace, similar to 4.1.1b. The number of TCP positions in a voxel gathered from FK was considered to be a suitable representation of efficient reachability in certain voxelized sections. In [146], it is proven invalid for most cases: If the positions in the voxel represent a singular configuration, any large step in the configuration space will cause small steps in the Cartesian space, leading to an abundance of poses in a single region. Also, uniform sampling of the configuration space does not guarantee uniform sampling in

the Cartesian space.

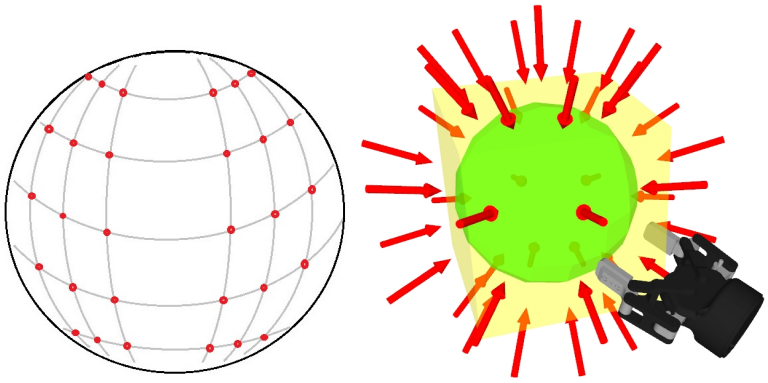


Figure 4.3.4: a)Sampling of points on the sphere, b) Discretization of a single voxel in the workspace. The cube(yellow) is the voxel to be searched. A sphere (green) is fitted inside the cube. The sphere is discretized with probable poses (red arrow)

For uniformly sampling the task space, the spheres representing the voxels are sampled for uniform point distribution on the sphere by the method presented in [118]. From every point of the distribution, a direction is assumed towards the center of the sphere. As depicted in Fig.4.3.4, an individual voxel of the discretized workspace is fitted with a sphere, and the sphere is sampled for points on its surface. From an individual point, a direction is considered towards the center of the sphere represented by a red arrow. Considering a frame, the direction towards the center of the sphere is the z-axis, while the x and y axes are tangential to the sphere surface.

All the frames created in the previous step are collected and searched for inverse kinematics(IK) solutions. A closed-form analytical inverse kinematics solver IKFast[36] is employed to find inverse kinematics solutions. For manipulators where analytical solutions are not available, a numerical solver KDL[1] is used which is less time-efficient due to the nature of the solver. All reachable poses that return a valid joint solution are stored in a multimap data structure with their corresponding sphere centers and inverse kinematics solutions.

4.3.4 Measure of Reachability

The reachability map generation procedure is simplified in Algorithm 1. A reachability metric is proposed for a given voxel based on the fraction of discrete poses in the sphere centered at that voxel that are reachable to the robot. In equation 4.24, the reachability measure is termed D, where the number of discretized frames of a sphere is represented by

N and the number of reachable frames calculated is R.

$$D = (R/N) * 100 \tag{4.24}$$

Based on the calculated D, spheres are assigned to different colors on the color spectrum, where blue represents higher reachability (i.e., the spheres with higher reachable poses) and red represents lower reachability. In terms of decreasing reachability, the colors are in the sequence of blue, sky blue, green, yellow and red.

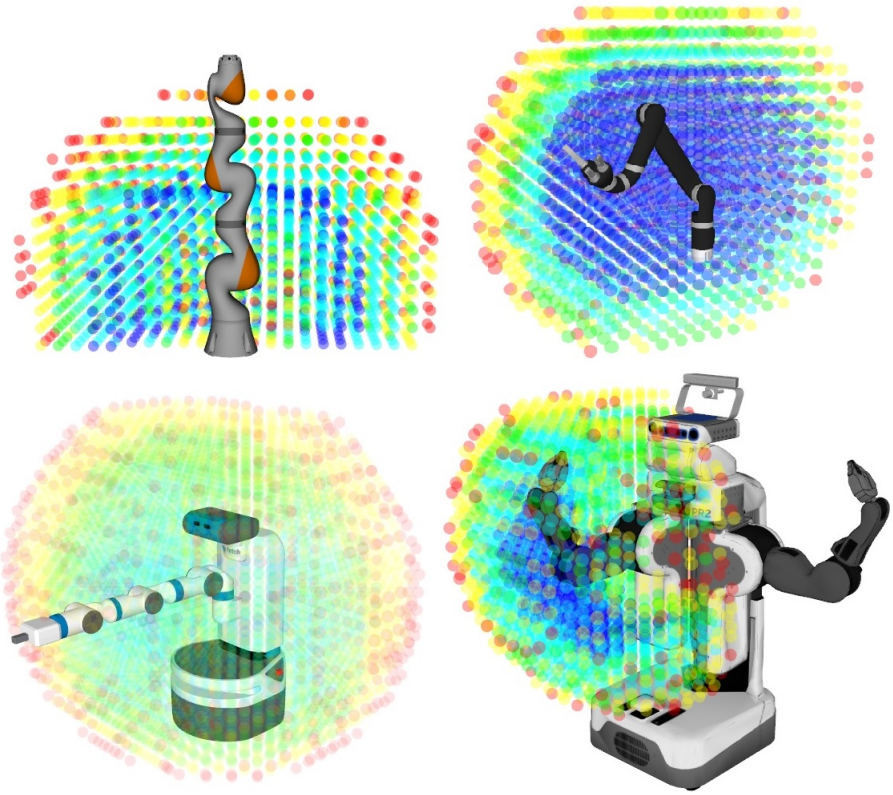


Figure 4.3.5: Reachability map representations in Reuleaux. From top to bottm, left to right: Reachability map of a) LWR 7DOF arm b) Jaco 6DOF arm c) Fetch 7DOF arm, d) PR2 7DOF arm

4.4 Queries from Reachability Map

As reachability map is a parameterization of the workspace of the robot, from the reachability map, some vital information of the workspace can be easily queried. e.g. some cartesian pose is reachable or not, which poses are most feasible to reach from a pool of poses and many more. The initial solution to a numerical IK solver can also be obtained from the reachability map, which provides fast convergence. As the voxels are already associated with a numerical

value of the reachability measure, it is quite easy to search for a pose associated with the voxel. To search for a reachability measure of a pose, first which voxel it belongs to should be determined. First, the position part (p_x, p_y, p_z) of the query pose should be searched in multimap data structure of the reachability map. It can be searched in $\mathcal{O}(n)$ time which provides the closest position of the voxel in the reachability map. After obtaining the position information of the voxel, the orientation (r_x, r_y, r_z) should be searched, which can also be done in $\mathcal{O}(n)$ time. It outputs the closest orientation of the pose from the reachability map. The associated reachability index value of the voxel can be assigned to pose and closest IK solution can be considered to the initial solution to the IK solver. The returned numerical value for the pose can be vital for the following sections, as the reachability index can be associated with the grasp hypotheses obtained by grasping algorithm for better searching in the grasp space.

Algorithm 2 Search for reachability measure

input (query pose $\mathbb{P}$)
output(reachability index, initial IK solution)

1: procedure *SearchReachability*
2: Search for position $(p_x, p_y, p_z) \in \mathbb{P}$of closest voxel
3: Search for orientation $(r_x, r_y, r_z) \in \mathbb{P}$ of closest pose in the voxel
4: Return RI of the voxel and initial IK solution
5: end procedure

4.5 Summary

In this chapter, a reachability map representation is presented which is a parameterization of the workspace of the robot. The workspace of the robot, the mapping between robot configuration and workspace points and the procedure to obtain the workspace are introduced. Existing work on reachability mapping and the implementation of the Reuleaux reachability mapping package is discussed. Finally, the importance of the reachability map in the context of grasping and query procedure for obtaining a numerical value for a task pose is presented.

Chapter 5: Grasp Generation

Robotic Grasping is one of the most needed qualities for manipulating objects. The robots to be placed in home or unstructured environment such as assistive services, healthcare, household applications, they should have the capability of grasping objects. In the recent years, the area of robotic grasping has gained significant attention and improvement. But still, there is no grasping algorithm available that can be considered as an "ideal" grasping algorithm. The main cause behind that is the vast quantity of objects availability and the uncertainty in the environment. Also, it is quite complex to present a grasping algorithm without the previous knowledge about the shape and pose of the object. The problem becomes more complex as the generated sensor data is noisy and incomplete.

Accurate grasping of possibly unknown objects is one of the main needs for robotic systems in unstructured environments. In order to do so, robotic systems have to go through multiple costly calculations. First, the object of interest needs to be identified, which includes a segmentation step resulting in a suitable surface representation of the object. In the next stage, the precise position of the object in the world should be determined. Then, grasps need to be synthesized by calculating the set of points on the object where the fingers can be placed followed by determining preferred approach directions. Finally, the system needs to plan a safe collision-free path of the manipulator towards the object. Typical robotic grasping methods focus on optimization of stable grasp metrics. One common approach is to maintain a database of objects with their corresponding preferred grasps. When the system encounters an object that can be identified with an entry from the database, the corresponding suitable referred grasp is pulled out. In this case, the system has to explore and evaluate an increasingly large number of objects and grasps for a given scenario. One of the limitations of such kind of systems is that the database has to be quite large to contain all of the common objects that a robot may encounter. If the system has to work in less structured environments, observing a novel object may be common. A methodology for grasping unknown objects is preferred here, rather than relying on the database.

5.1 Basic Concepts

Some of the most important concepts and terminologies related to robotic grasping are discussed in this section.

5.1.1 Definition of Grasp

According to Wikipedia, "A grasp is an act of taking, holding or seizing firmly with (or as if with) the hand." The implication of the definition is that by a grasp someone holds

something firmly by its hand. In the context of robotic grasping, by a grasp, a robotic hand provides force and torque balance and constraints the motion of the object and resists forces and torques in certain directions. Grasp synthesis associates with the problem of finding a suitable grasp among an infinite set of hypothesized grasps which can satisfy a certain set of criteria.

As most of the robotic hands try to mimic the human hand, understanding the grasping system of the human hand is necessary. In the earlier work in [94], a human grasp taxonomy is presented. The taxonomy is a systematic arrangement of the space of human grasps, and the organization of the taxonomy reveals some of the factors influencing grasp choice. Grasps can be placed on a continuum according to object size and power requirements.The taxonomy shows how task requirements (forces and motions) and object geometry combine to dictate grasp choice. The taxonomy is based on observations of single-handed operations by machinists working with metal parts and hand tools. The machinists were observed and interviewed and their grasp choices were recorded as they worked. In addition, their perceptions of tactile sensitivity, grasp strength, and dexterity were recorded. The Grasp Taxonomy is presented in Fig. 5.1.1.

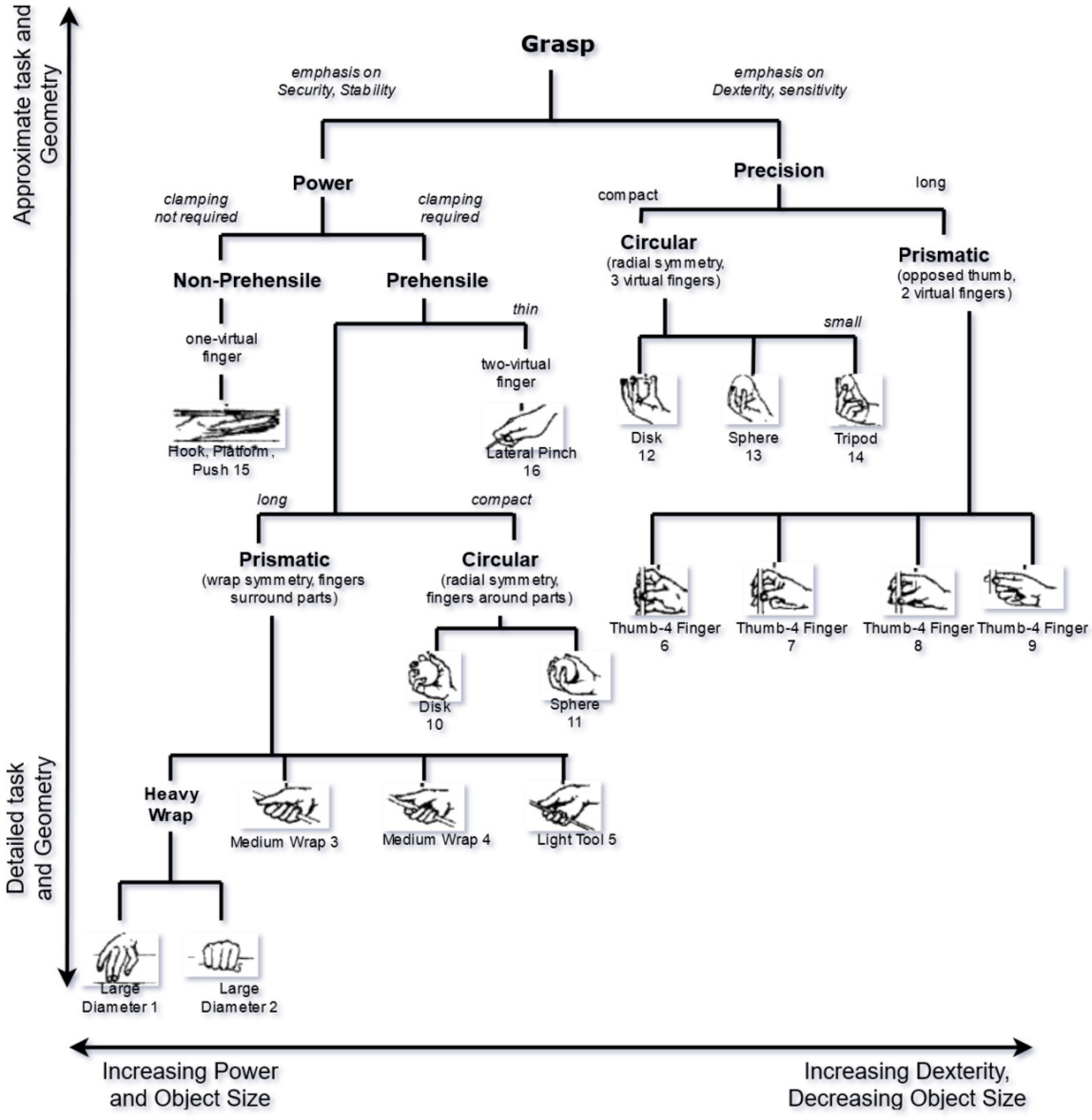
Several types of grasps can be performed. They are mainly:

- Force-closure Grasp: These types of grasps completely restrain the object.
- Torque-closure Grasp: In these type of grasps, the fingers completely restrain any external moment of a force.
- Equilibrium: The contact forces can balance the object weight and external forces. The origin is contained within grasp wrench space.
- Manipulable Grasp: These type of grasps can impart arbitrary velocities on the object without breaking contact.

5.1.2 Friction cone and Wrench Space

Friction at a contact point allows forces in the directions other than the contact normal. Considering a set of points C_i where the robot finger touches the object, the contact normal n_i at C_i points from the object to the finger. Let us assume the contact force as f_i^c that is exerted by the object to the finger. The normal component $p_i = (n_i \cdot f_i^c)$ is called pressure

Figure 5.1.1 *(following page)*: Grasp taxonomoy reimagined from[94]


Approximate task and Geometry
Detailed task and Geometry
Grasp
emphasis on Security, Stability
emphasis on Dexterity, sensitivity
Power
Precision
clamping not required
clamping required
compact
long
Non-Prehensile
Prehensile
Circular (radial symmetry, 3 virtual fingers)
Prismatic (opposed thumb, 2 virtual fingers)
thin
small
one-virtual finger
two-virtual finger
Hook, Platform, Push 15
Lateral Pinch 16
Disk 12
Sphere 13
Tripod 14
long
compact
Prismatic (wrap symmetry, fingers surround parts)
Circular (radial symmetry, fingers around parts)
Thumb-4 Finger 6
Thumb-4 Finger 7
Thumb-4 Finger 8
Thumb-4 Finger 9
Disk 10
Sphere 11
Heavy Wrap
Medium Wrap 3
Medium Wrap 4
Light Tool 5
Large Diameter 1
Large Diameter 2
Increasing Power and Object Size
Increasing Dexterity, Decreasing Object Size

and the tangential component $f_i^t = f_i^c - (n_i \cdot f_i^c)$ is the friction force at C_i. The point contact remains the fixed contact mode until the contact force f_i^c lies inside the friction cone directed by n_i.

$$f_i^c \cdot n_i > 0 \ \ and \ \ \|f_i^t\|_2 \leq \mu_i (f_i^c \cdot n_i) \tag{5.1}$$

where μ_i is the static friction. The friction cone can be estimated as a convex sum of a force vector on the boundary assuming an unit normal vector.

$$f \approx \sum_{j=1}^{m} \alpha_i f_j \tag{5.2}$$

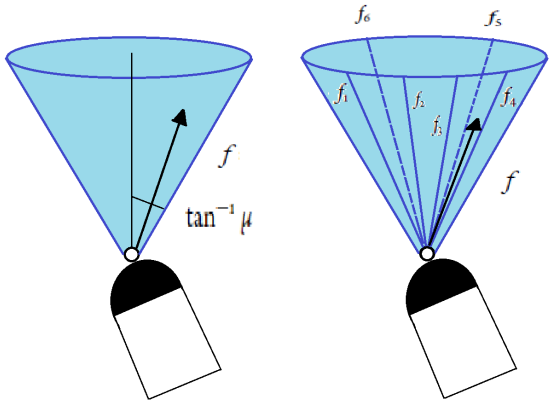


Figure 5.1.2: friction cone

A contact wrench is a set of forces and torques applied through a contact point. For a 2D point, a wrench is 3- dimensional vector with 2 force components and 1 torque component, while for 3d case, a wrench is a 6-dimensional vector with 3 force and 3 torque components. The set of all possible wrenches are considered as wrench space. A wrench is a point on the wrench space. The space of wrenches that can be applied by a grasp is a grasp wrench space(GWS) and the space of wrenches that can be applied by a task is called a task wrench space (TWS).

$$W(i,j) = \begin{vmatrix} f(i,j) \\ \lambda(d \times f(i,j)) \end{vmatrix} \tag{5.3}$$

where $W(i,j)$ is the wrench. The force is defined by $f(i,j)$ and the torque is defined by $d \times f(i,j)$. λ is the torque scale factor and $\lambda = \dfrac{1}{d_{max}}$.

5.1.3 Force Closure and Form Closure

For a grasp to be force-closed, the fingers have to apply arbitrary wrenches on the object through some contact points and the motion of the objects is restricted by the contact forces [95]. To be a force closure grasp, the contact wrenches should span the whole wrench space. In a force closure grasp, generally friction forces are modeled by friction cones and the grasp is able to resist all forces and torques from arbitrary directions. Force closure can be possibly achieved within minimal contact points and it is mainly used for precision grasps. But it is much difficult because of the nonlinear property of the Coulomb-friction cones.

On the other hand, in the case of form-closure, the contact points of the grasp immobilizes the object without depending on the contact surface friction [17]. Form closure is a better suitable condition than force-closure and mostly used for power grasps.

5.1.4 Grasp Quality

For analyzing a grasp, a grasp quality measure is needed which can also serve a criteria for grasp synthesis. A detailed review of grasp quality measures is presented in [116]. Some of the most popular grasp quality measures are:

1. Grasp quality can be measured in terms of force closure property where a specific line is considered between two grasping points [119] [74]. The magnitude of angle between the applied force direction and the line can be considered for quality measure.

2. Maximum finger force and total finger forces can also be considered as grasp quality measure which should be optimized for searching for stable grasp [88].

3. In a grasp, the position of the gripper fingers is a vital property which can be considered a grasp quality measure. In [30], a method is proposed where maximum singular value and the combination of all singular values are used as grasp quality measure. It is worth mentioning that, the gripper singularity should always be avoided.

4. Humans tend to grasp objects on the principal axis of an object. Utilizing this property, a grasp quality is proposed in [14] and [13] which in combination with other measure provides improved results.

5. Representing contact wrenches as points and forming a polygon by these points (grasp polygon) [88] proposes a method where the area of this polygon is used as maximization criterion. On the other hand, the distance of the center of mass of the object and the center of the polygon should be minimized.

6. In [117], the proposed method utilizes the property of the surface curvature of the contact surface and used it as grasp quality measure.

5.2 Previous Work

As grasping is one of the most fundamental tasks in robotics, grasping objects by robotic hands have a significant amount of history. There are several articles present that tried to accumulate surveys on grasp synthesis [16] [20] [124]. In this section, a comprehensive discussion on grasping known, familiar and unknown objects is presented. Additionally, discussion on grasping objects based on superellipsoid modeling is also presented, as it is most closely related to our work.

5.2.1 Grasping Known Objects

For grasping a known object, the problem of grasp synthesis reduces to the problem of searching in the database. In this type of methods, the shape model of the object is previously known and the grasp hypotheses are stored in the database. Utilizing a set of criteria, the search for optimum grasp in the database of grasp hypothesis can be performed. A detailed analysis of these types of methods is presented in [16] .

Earlier work on grasping objects were mostly based on extracting edges of polygons from a 2D vision sensors. By the method proposed in [95], precision grasps can be synthesized, where force closure grasps are performed for 2-finger grippers and form closure grasps are performed for 4-finger grippers. Another method for 2-finger force closure grasps for curved objects are introduced in [44], where each part of the object are modeled by parametric curves and each part are considered for grasp synthesis. Force closure grasps for 2 finger grippers by optimizing maximum finger force is discussed in [45]. In the work of [63] and [27], the local curvature is classified as convex and concave regions and grasps are searched in the regions. Modeling object boundary is also presented in [117], where the authors tried to model the boundary by Elliptic Fourier Descriptors(EFD). An algorithm for grasping considering the holes of the object is presented in [91] and using wrenches for optimization criteria is presented in [88].

The complexity increases significantly for 3D objects due to added dimension and the pose of the object should also be identified relative to the robot body. Most initial efforts in grasping known objects in 3D is presented by extending the procedures from 2D grasping in 3D. Most substantial and used work in grasping is presented in [86], where a simulator GraspIt is presented. In the simulator, grasp heuristics are calculated offline and in the online scenario, best grasp is chosen. Another notable simulator for generating grasps in

OpenGrasp [76]. The basis of the work presented in[87] is model simplification, where object models are simplified for basic shape primitives such as boxes, cones, and cylinders. Another method that utilizes model simplification is presented in [59]. Methods presented in [40] and [79] deals directly with the inaccuracies in the model and pose estimation. The gripper preshapes are learned from a human demonstrator using a glove which significantly reduces the possible grasp space. In terms of grasping known objects from a previously created database, there are several works present which focuses on constructing the grasp database [51], [67], [75], [104], [120].

There are also several works which incorporate learning in terms of grasping known objects. In the proposed work in [49], the shape properties are mapped with approach vector by neural networks and data is obtained by human volunteers. A framework is presented in [26] which maps object representation to task representation by modeling discrete views of the object by simple shapes and utilizes a neural network to map them to task representations. The work of [41] the sub-components of the grasp are learned from manually labeled synthetic data. Another work which uses human experience for grasping is the work in [34] which builds the database around object-specific empirical grasp density. A method for handling object uncertainty using partially observable Markov Decision Process(POMDPs) is presented in [55]. In [143], the sum of finger forces is optimized by a genetic algorithm.

Grasping known object by estimating pose from a monocular image and further grasp selection from an offline database is presented in [92] and later modified for stereo images in [10]. The demonstration of the method on a whole robotic system is shown in [8] and [59]. In [98], an approach is proposed for grasping unknown object in clutter environment by 3D object recognition and pose estimation based on object geometry. Using object categorization to find grasps for similar objects is presented in [141] which overcomes the issue of incomplete data perceived by a partial view. Grasping familiar objects also explored in the work of [19] and [21] where a global shape representation is introduced and the system is trained by a supervised learning on synthetic images. In the work of [53] template-based modeling is done on objects and grasps are created by programming by demonstration technique. Another similar method where prototype grasps are taught to the system is presented in [35]. Incorporating object models and enabling the use of force closure grasps are incorporated in several other works such as [85], [78] and [103].

5.2.2 Grasping Unknown Objects

The methods which try to grasp unknown objects does not rely on any database for object models or previously calculated grasp hypotheses. Most generally, the system has to identify the objects from partial sensor data and generate grasps in an online scenario.

The work in [69] aims to grasp objects by a two finger gripper based on raw depth data by searching a pattern that fits in the interior of the end-effector by maximizing the contact area between perceived point cloud and the gripper. Without utilizing any prior object model, the work in [77] also utilizes the shape matching approach. In the work of [53], a heightmaps inspired template matching algorithm is presented which generalizes grasps for novel objects. [115] considers a certain set of grasps and grabs the object from the table top surface. The grasps are considered to be closest to the center of mass of the objects. This method is later improved in the work of [114] where opening of top surfaces and handles can also be grasped. Contact reactive grasping by tactile information is presented in [56] where depth images are used for calculating the principal axis and the grasp is executed by heuristics from a defined grasp set. Another work that uses contact-reactive grasping is presented in [122], where the system approaches to grasp around the principal axis and while execution, if an unwanted contact is detected, the gripper reacts and corrects its pose. In [33], a classifier is trained for stabilizing grasps based on tactile information. Another variation of grasping by contact reaction is presented in [57] where proximity sensors are placed on the fingertips and finger poses are tuned to achieve stable grasps. Grasping by knowledge base where the knowledge base is created by an offline simulator is presented in [31].

Some algorithms tend to fill the gap of missing information of the partial sensor data utilizing symmetry. In [109], the shape of the object is completed from the partial view and a mesh of the completed model is generated for grasping. Generating a full model from a partial model can be approached in several ways, mainly symmetry detection [89] and symmetry plane [133] [83] [84]. Extrusion-based object completion has also been proposed in [71]. [100] presents a different strategy of changing the viewpoint in a controlled manner and registering all the partial views to create a complete model. This technique provides good results, yet it is not very suitable for real- time systems and is prone to errors due to the registration of several views. In addition, results are not satisfactory when the working environment becomes densely cluttered.

Grasping unknown objects by 2D image features are propose in [9] [120]. Here supervised learning on local patch-based image and depth features for grasping unknown objects is proposed. The work is improved in [64] by including the capability of learning optimal gripper width. Another improvement of the work is presented in [74], where the authors accommodated multiple contacts in the grasps. The algorithm is fully implemented in a task of lifting object and unloading a dishwasher is presented in [121]. A grasping algorithm based on edge and texture based features to build a hierarchical representation is proposed in [70]. An approach to combine 2D and 3D features for grasping is presented in [105]. The approach further extended in [104] to include surface and edge-based grasps. Reinforcement

leaarning is proposed for a solution in grasping unknown object in [11].

Some of the systems deal directly with clearing the whole table. The work of [32] proposes an end-to-end approach of manipulating unknown objects in a pile and placing them into a bin. The method of clearing a box by a Height Accumulated Feature (HAF) is presented in [46]. The system is trained by an SVM to search for good grasps with the HAF features. [25] proposes an algorithm to grasp unknown objects by curvature information from silhouette images. In [2], the proposed work models the object by Gaussian and pick objects in the domestic environment by optimizing a grasp criterion of minimizing distance between robot palm and object. Another method for that grasps unknown objects for clearing a table is presented in [131], where obtaining a point cloud and geometric parameters of the hand, the system outputs a set of hand configurations that are expected to be good grasps. The work is extended in [52] where the robot grasp configurations are detected using a convolutional neural network.

5.2.3 Grasping by Superquadric Modelling

Another notable method to deal with the challenges of grasping known and unknown objects tries to approximate the shape of the object models and calculates grasp on the approximated models. Different shape primitives such as boxes, cones, cylinder, and spheres are implemented in the work of [87], [58] and [108]. In the work of [50] superquadric models are used to approximate object models. This work approaches towards decomposing an object model by decomposition tree and fit multiple superquadrics and later calculate grasps by GraspIt simulator [86]. Another work that uses decomposition trees and fit superquadric models on each part is presented in [149]. The proposed work in [136] segments object parts from range images based on semantic cues and then fits superquadrics on each part for classification of different generic shapes. Another work that utilizes range image data for object manipulation by superquadrics from one view in cluttered scene is presented in [18]. In the method, the detection is based on hierarchical RANSAC search for obtaining fast detection results and voted by quality-of-fit criteria. Using superquadrics a distinction between graspable and non-graspable objects is presented in [42]. After segmenting the object parts, each part is fitted with a superquadric and trained by an artificial neural network to classify between prehensile or not. Incorporating superquadric as a higher level representation to approximate object model in the scene and grasping with multi-fingered hands is proposed in [128]. In the work of [29], the region of interest(ROI) is decomposed into meaningful ROIs by decomposition and superquadrics are fitted to them. Finally dynamically adjusting voxel shapes are merged for estimating surfaces. A method for multiple superquadric fitting for efficient pose and shape recovery is presented in [38]. The method introduces multi-scale

voxelization for superquadric fitting and evaluated on multiple datasets. Superquadrics are also used for modeling the shape hierarchy for visually guided grasping in [113]. The work in [48] proposes a method to improve local symmetry estimation by superquadric fitting. A probabilistic framework to suitable grasping regions on objects by superquadric modeling is proposed in [43]. The object model is acquired by occupancy grid representation that deals with uncertainty and later decomposes object global shape into components. Superquadric model based approach to find substitute tools in 3D vision data is proposed in [3] where simple hand-coded models of known tools are presented by superquadrics and a relationship among them is maintained. Later the models are fitted to point clouds of unknown tools and identified for good fit. A method to localize objects using superquadric model by Kinect sensor is proposed in [7].

Using symmetry information and extrusion for grasping by superquadric modeling is presented in [109]. The work is improved in [110] where an approach for efficient grasp selection for manipulation of unknown objects utilizing superquadric modeling is presented in. A grasp manipulability measure is proposed to prioritize grasps. Recent work in grasping by superquadric is approached in [137], where the object model is computed in real-time using stereo vision and the pose of the object is formulated as a nonlinear constrained optimization problem.

5.3 Input to Grasping System

In this section grasping known objects directly on superquadric models is been discussed following the work in [5]. The approach towards grasping unknown object by the method is different from other works such as [50] [137] and [43] is that the grasping phase is an online phase, where no simulator is used for grasp calculation or any offline database is not incorporated. The grasp configurations are achieved from the estimated geometry of the superquadrics, instead of directly searching in the point cloud data.

Reiterating the superquadrics equations, a superquadric can be represented by the spherical product of two superellipses:

$$r(\eta,\omega) = S_1(\eta) \oplus S_2(\omega) = \begin{Bmatrix} \cos^{\epsilon_1}\eta \\ a_3 \sin^{\epsilon_1}\eta \end{Bmatrix} \otimes \begin{Bmatrix} a_1 \cos^{\epsilon_2}\omega \\ a_2 \sin^{\epsilon_2}\omega \end{Bmatrix} = \begin{Bmatrix} a_1 \cos^{\epsilon_1}\eta \cos^{\epsilon_2}\omega \\ a_2 \cos^{\epsilon_1}\eta \sin^{\epsilon_2}\omega \\ a_3 \sin^{\epsilon_1}\eta \end{Bmatrix}, \quad (5.4)$$

with $\eta \in [\frac{-\pi}{2}, \frac{\pi}{2}]$ and $\omega \in [-\pi, \pi]$,where a_1, a_2, a_3 are the scaling factors of the three principal axes. The exponent ϵ_1 controls the shape of the superquadric's cross-section in the planes orthogonal to (x, y) plane, and ϵ_2 controls the shape of the superquadric's cross-section

parallel to the (x, y) plane. It can also be expressed by an implicit equation as:

$$f(a, x, y, z): \quad \left(\left| \frac{x}{a_1} \right|^{\frac{2}{\epsilon_2}} + \left| \frac{y}{a_2} \right|^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left| \frac{z}{a_3} \right|^{\frac{2}{\epsilon_1}} = 1. \tag{5.5}$$

where it is bounded by the planes given by $-a_1 \leq x \leq a_1$, $-a_2 \leq y \leq a_2$, and $-a_3 \leq z \leq a_3$. The total set of parameters that fully defines the superquadric consists of 11 variables, $\{a_1$, a_2, a_3, ϵ_1, ϵ_2, p_x, p_y, p_z, ρ, ψ, θ $\}$

The input to the system is a point cloud data obtained from a 3D sensor such as Kinect. This type of time-of-flight sensors can only provide a partial view model of the environment. A typical scene captured by the sensor is presented in Fig. 5.3.1 where the box and cylinder objects are partially visible by the sensor. The other parts of the objects are occluded. The output of the grasping system is a grasp defined by a) Gripper opening angle, b) grasp pose and c) approach direction.

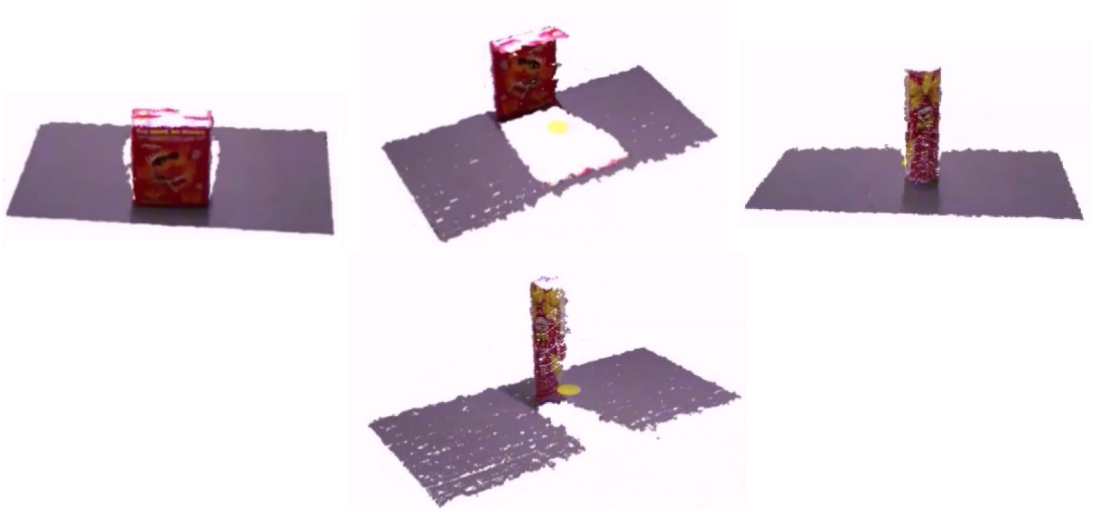


Figure 5.3.1: Scene captured by a 3D sensor. The point cloud represents the partial model of the object. The other sides of the objects are occluded

5.3.1 Preprocessing Point Cloud

From a typical 3D scene of an environment, the system deals with only the objects on the table. So the rest of points can be discarded by specifying a workspace composed of the table and the objects on the table. The point cloud resting on the table can be segmented out by the methods described in Section.2.7. If there are multiple objects on the table, the objects can be clustered by the clustering algorithms defined in Section.2.8. After the object cluster is defined, the approximated object model can be run through a mirroring stage, by

which the points on the object are mirrored and an approximated shape of the object can be obtained which can surpass the limitations of the occluded region of the object.

Creating a mirrored surface

Algorithm 3 Generate Mirrored Cloud

input (organized point cloud *Cloud*)

```
procedure CompleteCloud
Segment out the Table
Segment Objects
    for each i in Objects  do
        Find pose (c_x, c_y, c_z, ρ, ψ, θ)
        Generate local frame at c_x, c_y, c_z
        for each point p in i do
            Calculate Euclidean distance (c, p)
            Negate distance and find point m
            Add m to i
        end for
    end for
end procedure
```

In this section, the procedure for completion of object models from the partial view point cloud is described. A typical point cloud view from a point cloud capturing device is shown in Fig.5.3.3a , where only a part of the object is captured. The other parts of the object are on the occluded side, so they can only be captured by a device situated on the other side of the table. Reconstruction of the full object model can then be performed by registering the two views from two individual devices. As most of the objects in our daily lives are symmetric by nature, by the law of symmetry it can be asserted that the occluded side of the object is merely a reflection of the visible part. By using only this assumption, the mirroring algorithm presented here creates the occluded part of the point cloud and reconstructs the object model online. To obtain a complete approximated model of the perceived object, the points perceived are mirrored based on the pose (p_x, p_y, p_z,θ) estimated in the Section.3.7. A local reference frame is assumed at the center of the object and the Euclidean distance of all the points from the centroid is calculated. All the points are reflected on the occluded side by scaling. So for every single point $P_i(p_x, p_y, p_z)$, a mirrored point $M_i(-p_x, -p_y, -p_z)$ is created where p_x, p_y, p_z are distances in the x, y, z directions from the center of the object. Though the pose does not contain any ρ and ψ values, this mirroring method can work on any arbitrary orientation of the object. To deal with noisy data, all the point clouds can be

processed through a series of outlier removal algorithms.

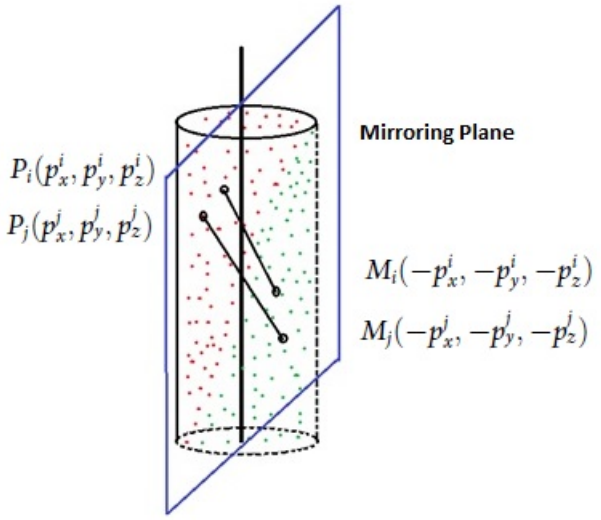


Figure 5.3.2: Illustration of the mirroring procedure

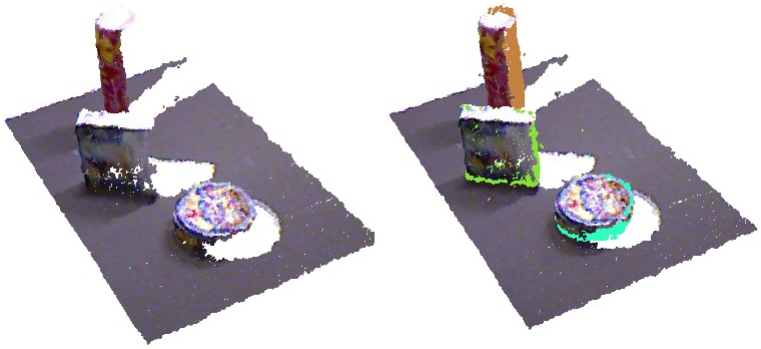


Figure 5.3.3: Mirroring: From left to right, a) Point cloud Scene and b) each object with its own mirrored points

An illustration for the mirroring procedure is presented in Fig.5.3.2, where the red points are original points and the green points are in-correspondence mirrored points which are mirrored by the plane. The mirroring algorithm is presented in Algorithm.3. The output of the mirroring procedure is shown in Fig.5.3.3b, where the mirrored points are the colored points and original points are shown in RGB. The approximated object model is advanced to the superquadric fitting pipeline to obtain the 11 parameters $\{a_1, a_2, a_3, \epsilon_1, \epsilon_2, p_x, p_y, p_z, \rho, \psi, \theta\}$ of the superquadric. The grasps can be directly searched on the superquadrics.

5.3.2 Gripper parameters

The gripper of the robotic system is 2-finger parallel jaw gripper. The gripper width(w) and the depth(d) are illustrated in the Fig.5.3.4 for 2 different grippers; PR2 and Robotiq. The gripper frames are also shown. A notion of solvable multi-fingered hands is presented in [4].

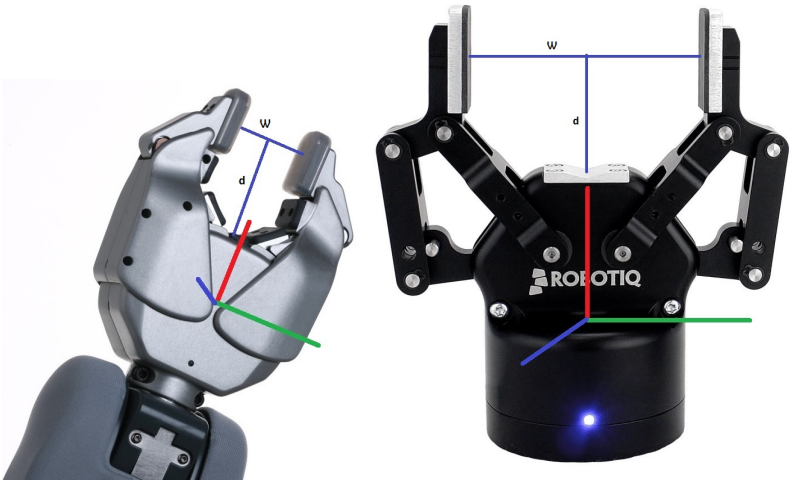


Figure 5.3.4: Gripper parameters for 2 parallel jaw grippers. a) PR2 gripper and b) Robotiq 2-finger gripper.

5.4 Grasp Hypothesis Generation

This work focuses on the use of two-fingered grippers for grasping the objects. It is well known that antipodal grasps can be successful in both convex and concave objects in the presence of friction. Finding antipodal points on generally-shaped object models usually returns many candidates, which are ranked according to some metric. In order to make the grasp more stable and balanced, we impose two conditions: a) grasping closer to the centroid of the object and b) at the points of minimum curvature. In addition, we have constraints given by the depth and width of the robotic gripper and the task-based direction of approach.

5.4.1 Gripper Openning angle

The gripper opening angle is dependent on the placement of the gripper on the object. While finger placements are determined, the gripper opening angle can be easily determined by the mapping between the gripper width (w) and the opening angle(α) declared by the gripper kinematics. The gripper contact points for antipodal grasping with minimum curvature can

be calculated from the superquadric parameters to be

$$p_W = [T_{WK}][T_{KS_i}](\pm \begin{Bmatrix} p_a \\ 1 \end{Bmatrix}), \tag{5.6}$$

where $[T_{WK}]$ is the 4×4 homogeneous transformation from the world frame to the camera frame, $[T_{KS_i}]$ is the 4×4 homogeneous transformation from the camera frame to the principal frame of superquadric i, and p_a is the position vector of the contact points in the superquadric frame, to be selected according to the criteria explained below and summarized in Table 5.4.1.

Table 5.4.1: Position vector to contact points as a function of the length of the semi-axis and the exponent ϵ.

Exponent	Dimensions	Position vector
$\epsilon < 1$	$a_1 > a_2$	$p_a = (0, a_2, 0, 1)$
	$a_1 = a_2$	$p_a = (a_1, 0, 0, 1)$ or $p_a = (0, a_2, 0, 1)$
	$a_1 < a_2$	$p_a = (a_1, 0, 0, 1)$
$\epsilon = 1$	$a_1 > a_2$	$p_a = (0, a_2, 0, 1)$
	$a_1 = a_2$	$p_a = (a_1 \cos\omega, a_1 \sin\omega, 0, 1)$ for all ω
	$a_1 < a_2$	$p_a = (a_1, 0, 0, 1)$
$\epsilon > 1$	$a_1 = a_2$	$p_a = (a_1/\sqrt{2}, a_1/\sqrt{2}, 0, 1)$
	$a_1 \neq a_2$	$p_a(\omega)$ as per Eq.(5.9)

The superquadric is convex for values of the exponent $0 < \epsilon < 2$; for larger values, it becomes concave at the corresponding cross section. Curvature in superquadric can be easily calculated from either their implicit or their parametric expression. Considering the implicit expression in eq (5.5) at the $x - z$ plane obtained when $\omega = 0$, the curvature is

$$k(\eta) = \frac{(\epsilon_1 - 2) a_1 a_3 \cos^{\epsilon_1 - 4}\eta \sin^{\epsilon_1 - 4}\eta}{\epsilon_1 \sqrt{(a_1^2 \cos^{2\epsilon_1 - 4}\eta + a_3^2 \sin^{2\epsilon_1 - 4}\eta)^3}}, \tag{5.7}$$

and similarly for the other two planes passing through the origin of the superquadric frame. Depending on the value of the exponent ϵ_1, the minimum curvature will be found at the intersection with the axes or at 45^o from them.

For substantially different lengths of the semi-axes of the convex superquadrics, the points of minimum curvature at $\epsilon > 1$ are not antipodal. Here, normal directions to the superquadrics are found with minimum angular deviation. Disregarding its sign, the normal to the superquadrics can be calculated for a coordinate plane passing through the origin as

the second derivative of the parametrized curve. For $x-z$ plane at $\omega = 0$ normal n is

$$n(\eta) = \left\{ \begin{array}{c} \frac{a_3 \cos^{\epsilon_1+2}\eta \sin^{2\epsilon_1}\eta}{\sqrt{a_3^2 \cos^{2\epsilon_1+4}\eta \sin^{4\epsilon_1}\eta + a_1^2 \cos^{4\epsilon_1}\eta \sin^{2\epsilon_1+4}\eta}} \\ \frac{a_1 \cos^{2\epsilon_1}\eta \sin^{\epsilon_1+2}\eta}{\sqrt{a_3^2 \cos^{2\epsilon_1+4}\eta \sin^{4\epsilon_1}\eta + a_1^2 \cos^{4\epsilon_1}\eta \sin^{2\epsilon_1+4}\eta}} \end{array} \right\} \tag{5.8}$$

Fig. 5.4.1 shows an example of the antipodal points with minimum curvature on the $x-z$ plane; if a solution exists, then there will always be another symmetric solution. In addition, the contact points at the major axes are antipodal, but depending on the value of the exponent, they may present higher curvature or a cusp for extreme cases.

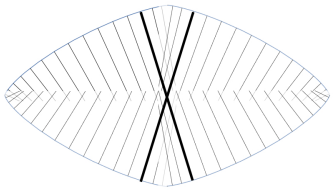


Figure 5.4.1: Normals to the surface of the superquadrics for $a_1 = 0.5$, $a_3 = 0.9$ and $\epsilon_1 = 1.5$. Normal lines of approximate antipodal points marked with thicker lines

The angular parameter of the approximate antipodal points in the plane can be found by imposing that the normal line passes close enough to the origin,

$$p_a = \min_{\eta}\left(\frac{a_1 a_3 \cos^{2\epsilon_1}\eta \sin^{2\epsilon_1}\eta}{\sqrt{a_3^2 \cos^{2\epsilon_1+4}\eta \sin^{4\epsilon_1}\eta + a_1^2 \cos^{4\epsilon_1}\eta \sin^{2\epsilon_1+4}\eta}} - 1\right). \tag{5.9}$$

For concave superquadrics, the curvature helps make the grasp closed. Antipodal points are sought using eq (5.9).

5.4.2 Grasp Pose

While the gripper opening angle is formed, the grasp pose of the gripper can be determined by the normal of the antipodal points. Utilizing the gripper depth parameter (d) the distance of the gripper frame from the normal line to the antipodal points are calculated. Let us define the plane of grasping as the plane containing the contact points and normal to the fingers of the gripper. The initial gripper pose should be in this plane. Because of the condition of entered grasp, at least one of the dimensions of the superellipsoid at the plane of grasping, a_i, needs to be smaller than the width of the gripper, w. In addition, the dimension of the major axis in the direction perpendicular to that plane needs to be smaller than the depth

of the gripper, d. The grasping plane can be defined as:

$$s_W = [T_{WK}][T_{KS_i}](\pm \begin{Bmatrix} s_a \\ 0 \end{Bmatrix}), \tag{5.10}$$

where $s_a = x, y$, or z or a coordinate rotation about one of those axes.

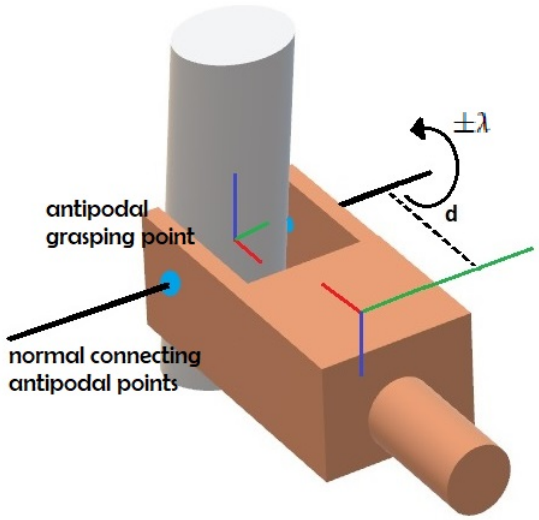


Figure 5.4.2: Grasp pose

As one of the criteria is to grasp objects by the major axes, the initial gripper pose should be parallel but inverse to the major axes of the superquadric by which the grasp is determined and also perpendicular to other two axes. In reality, grasps determined only by major axes is not a sufficient condition, as the determined grasp pose may not be reachable by the robot. The robot poses can be sampled by iterating through a tilting angle ($\pm\lambda$) to accommodate multiple grasp poses from a single normal line to the antipodal points. An illustration of determining the grasp pose is presented in Fig.5.4.2.

5.4.3 Direction of approach

After Sampling multiple grasp poses, the approach direction of the gripper can be calculated by sampling another pose in a fixed distance (t) from the grasp pose and applying a fixed transformation in the x-axis of the gripper. The transformation has an identity (I) rotation and a positive fixed translation. The line that passes through the poses is the approach direction of the gripper and finally, it is transformed into the robot frame.

Multiple grasp hypotheses are sampled by above-mentioned criteria for opening angle and grasp pose. For each sampled antipodal point, multiple grasps are defined by small iterations

of λ on the normal axis of the antipodal points. All the grasp hypothesis are collected and forwarded to the grasp filtering stage.

5.5 Grasp Filtering

The above-mentioned criteria are sufficient to find antipodal grasps on the superquadric, but they may not be efficient for executing on a real robot system due to constraints and type of the environment. Suppose the objects are situated on the table, so it can be stated obviously that the grasps in the -z principal axis are invalid, as the robot cannot grasp through the table surface. There are also other constraints on the system, by which all the grasps can be quantified and based on the assigned value, they can be filtered. Each grasp is assigned a numerical value G_i by:

$$G_i = (\frac{RI_i}{100.} + ct_i - d_i)s_i \tag{5.11}$$

Where RI_i measures the reachability of the grasp pose. The distance of the grasp pose from the center of the superquadric is d_i. The distance the arm have to travel to reach the grasp pose from its current position is assigned as t_i. c is a normalized factor for the voxelspace of the environment and s_i defines the distance of the grasp pose from the table surface defined by $s_i = h_{G_i} - h$, where h_{G_i} is the height of the grasp pose and h is the height of the table. s_i takes care of the fact that, the robot cannot execute the grasps which are through the table or very close to the table. The contribution of the d_i is that it makes sure, that the final grasp is closest to the center of the superquadric, so the moment on the object can be dissipated while picking up. The t_i makes sure that the robot tries not to grasp from any negative approach direction while RI_i measures the reachability of the grasp.

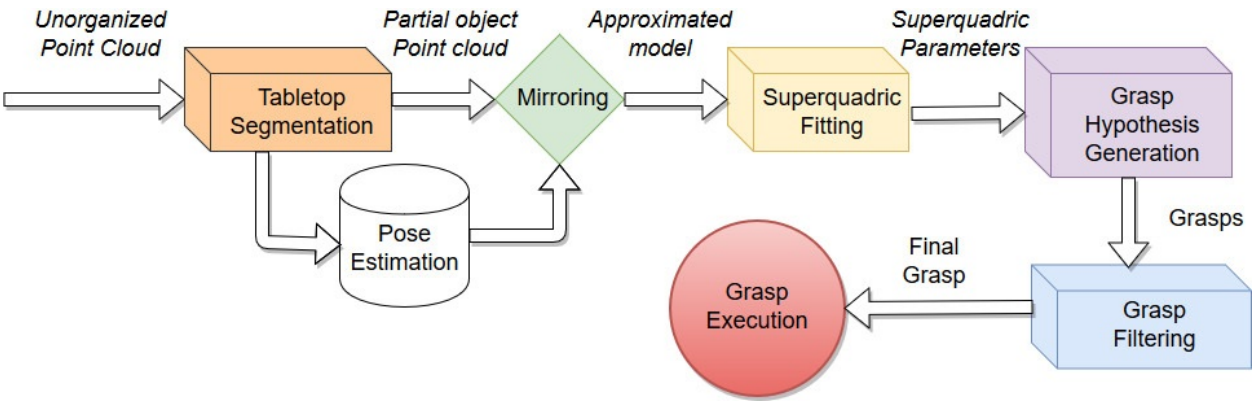


Figure 5.5.1: Grasping pipeline

Algorithm 4 shows the summary of the process to select the plane of grasping and the

Algorithm 4 Grasp synthesis

input (superquadrics parameters, depth of the gripper d, width of the gripper w)

```
procedure SelectContactPlanes
    for s_a superquadrics direction do Calculate s_W
        if angle(s_W,Z_W) > α or length(s_a) > d then
            Discard s_a
        end if
    end for
    for remaining s_a do
        if 2 * a_i > w for perpendicular directions then
            Discard s_a
        end if
    end for
end procedure
procedure CalculateContactPoints
    Select p_a (Table 5.4.1)
    Calculate p_W
end procedure
```

contact points. It is intuitive that the algorithm always tries to approach object from the minimum $\pm a_i$ direction.

It is shown in Fig. 5.5.2, where the first and last image shows grasping of the box from the minimum $\pm a_i$ direction. The red are those which cannot be executed. The yellow grasps are sampled by iterating by λ and the final green grasp has the highest grasp value. The final approach direction is shown in cyan. The whole grasping pipeline is illustrated in Fig.5.5.1, where the input to the system is unorganized point cloud representing the environment and the output is a final grasp defined by a grasp pose, gripper opening angle and approach of direction which can be executed on the real robotic system.

5.6 Summary

In this chapter, state-of-the-art in robotic grasping for known and unknown objects are discussed preceding by a discussion on common and most used terminologies in robot grasping. The other works that utilized superquadric modeling as an ingredient of robotic grasping are also presented. The grasping algorithm developed in the thesis is examined in detail.

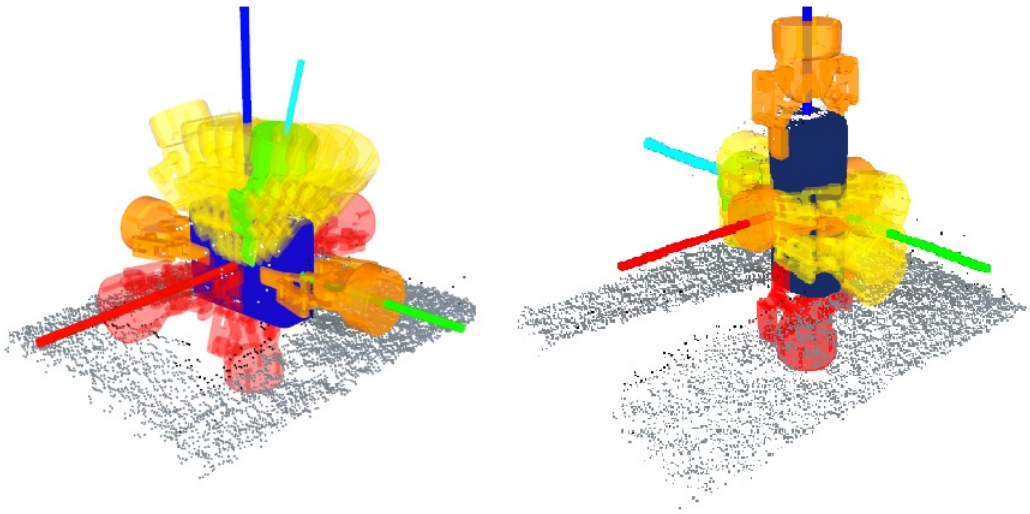


Figure 5.5.2: Grasps on the superquadrics. From left to right, a) on a box, b) on a cylinder. The red grasps are invalid, yellow grasps are potential grasp on different axes, light yellow grasps are potential grasp on current axes and green grasp in the best grasp. The final best approach direction is shown by cyan bar.

The focus of the work is to present a concrete grasping algorithm enabling robotic systems to generate antipodal grasping points for unknown objects and grasp them in real time. To achieve this, a superquadric model approach has been incorporated, which can fit on object models and the grasps can be directly performed on them. The main contribution here is 1) a library by which a reachability map of the robot can be constructed and 2) a novel grasping algorithm that fits grasping points directly on superquadric models and executes them.

The experiments with the reachability mapping library and grasping systems are presented here. To compare the efficiency of the reachability mapping, the results are compared with two other works that construct reachability map presented in [36] and [146]. The superquadric fitting results are compared with the work proposed in [109]. For comparing the results of grasping unknown objects, two recent works presented in [131] and [46] are recreated. The objects used for grasping are mostly symmetric household objects of various shape and sizes. A curated list of objects used in different experiments is shown in Table.6.0.1. The objects mentioned in the experiments are shown in Fig.6.0.1. The experiments are performed on 3 different types of robot, a) WAM arm with custom hook gripper, b) UR5 robot with Robotiq 2-finger gripper and c) PR2 robot with parallel-jaw gripper shown in Fig.6.0.2.

Figure 6.0.1: Objects used for the grasping experiments

The superquadric fitting experiments are performed in two conditions, where objects are in isolated scene (a single object in the scene) and cluttered scene (multiple objects are present while they are touching each other). The grasping experiments are performed also in isolated and cluttered scenario for all three robots. The reachability maps are generated for 3 different robots, a) PR2, b) UR5 and c) LWR arm for comparison with other methods.

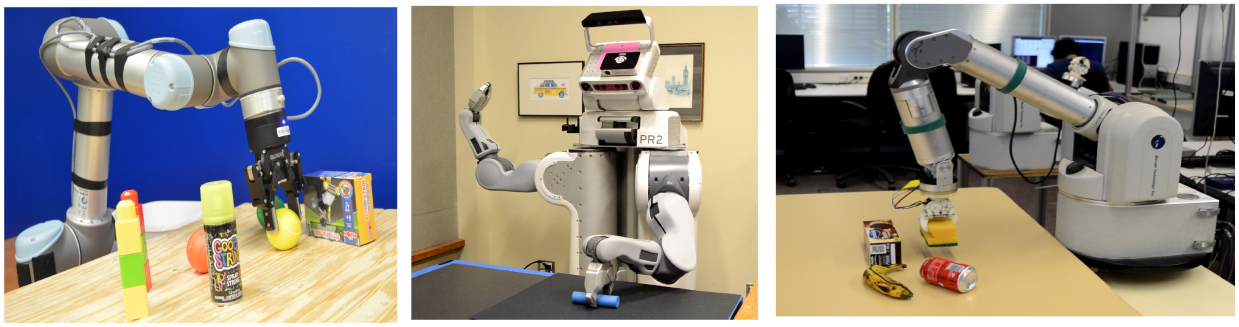


Figure 6.0.2: Robots used for the grasping experiments

6.1 Reachability Mapping Results

The Reuleaux map generation library was applied on several robots with different specifications and sizes. Without the collision checking facility, the average reachability map generation time was around 156 sec. However, without self-collision, the generated map was just a collection of reachable poses, which is disadvantageous. Since there is no standard metric to measure the efficiency of a reachability map, we can consider time efficiency and generalizability as the metrics. As the reachability map can be generated offline, its utility lies in using the map in other tasks.

In Table 6.1.1 we represent an analysis of Reuleaux's reachability map generation compared to two methods presented in [36] and [146] on three different robots. PR2 right arm is a 7DOF manipulator, rigidly connected to the body. The LWR and UR5 are considered on the ground. While the DOF of LRW is 7, the UR5 has only 6ODF. In method [36], available as open-source library, reachability is not represented as sphere; instead, it is represented as area. The opportunity to set the desired options for creating maps is only limited to saving joint solutions and setting a maximum radius. On the other hand, [146] represents reachability as a sphere and in extension with different shape representations. Because this method is most similar to our approach, we also represent their work as spheres.

In Reuleux, the creation of a reachability map is based on the desired resolution of the voxels and maximum radius. Resolution is a very useful parameter as it determines the size of the voxel, which later creates the size of the spheres. For all our experiments, we set the resolution of voxels at $0.08m$ and maximum radius at $1m$. For convenience, our implementation of [146] also uses the same resolution. For all the experiments, the IKFast [36] library was used to obtain ikfast solutions.

In Table 6.1.1, the significant difference in the number of poses processed stems from the fact that [36] obtained the poses by default parameters and in [146] all poses are rotated by

Table 6.0.1: Objects list

object	Superquadric Fitting		Grasping Experiment					
			WAM		UR5		PR2	
	single	clutter	single	clutter	single	clutter	single	clutter
Toy Cylinder	✓	✓			✓	✓	✓	✓
Big Ball	✓	✓					✓	✓
Cheese Box	✓	✓					✓	✓
large Dice	✓	✓			✓	✓		
Pringles Can	✓	✓	✓	✓			✓	✓
Coke Can	✓	✓	✓	✓	✓	✓		
Banana			✓	✓				
Apple			✓	✓				
Cleenex Box					✓	✓	✓	✓
Puzzle Box	✓	✓	✓	✓	✓	✓	✓	✓
Lego 1			✓	✓	✓	✓	✓	✓
Lego 2	✓	✓			✓	✓	✓	✓
Green Ball	✓	✓			✓	✓	✓	✓
Yellow Ball					✓	✓	✓	✓
Red Ball	✓	✓			✓			
Box1	✓	✓			✓	✓	✓	✓
Gooft String Cylinder	✓	✓			✓	✓	✓	✓
Chocolate Box	✓	✓	✓	✓				
ScrewDriver					✓	✓		
Eraser	✓	✓					✓	✓

Table 6.1.1: Rechability map results

Method	Robot	(x100000) Poses processed	(x1000) spheres created	Time(min)
Reuleaux	PR2	20.938	2552	124.31
	LWR	13.718	5127	160.41
	UR5	7.636	5017	143.27
Diankov et.al[36]	PR2	205.48	-	490.07
	LWR	145.727	-	427.11
	UR5	123.513	-	371.38
Zacharias et al [146]	PR2	104.69	2680	405.47
	LWR	68.59	5213	542.18
	UR5	38.18	4883	413.23

5 degrees in z-direction to obtain extra poses. Regarding the time difference, in our system, we have filtered out the spheres on the robot body in a preprocessing step.

6.2 Superquadric Fitting Results

Our object dataset mostly consists of small symmetric objects, such as small boxes, cylinders, and balls with varied parameters in x, y, z directions. For calculating the errors, in an offline scenario, we obtained the ground truth (a_1, a_2, a_3) parameters of all objects. ϵ_1 and ϵ_2 are obtained by iterating over the values between 0.1 and 1.9 and choosing the volume that best fits the object. The error is the radial distance error between the point cloud generated by the ground truth information and the point cloud generated by estimated superquadric parameters. For evaluation of single objects, 5 different poses are defined on the table and objects are perceived from 5 different poses. The results in 6.2.1 are a comparison of recovered parameters between our method, the method presented in [109] and a method where pose estimation is done by PCA and superquadric fitting without mirroring. As the result suggests, our method shows improved results in superquadric fitting in terms of time-efficiency and accuracy, as out method determines the poses on the approximated center of the object. Also, instead of optimizing 11 parameters of the superquadric, we only optimize 5 parameters related to object shape. While the size of the object increases, the time duration of the optimization process also increases.

Figure 6.2.1: The complete superquadrics fitting pipeline. From top to bottom, the mirroring process with a box and a cylinder object (The capturing device can be assumed at the right side of the image): a) RGB side view of the object, b) captured point cloud of the scene, c) segmented point cloud of the object, d) mirrored point cloud with the segmented cloud, e) superquadric representation of the object.

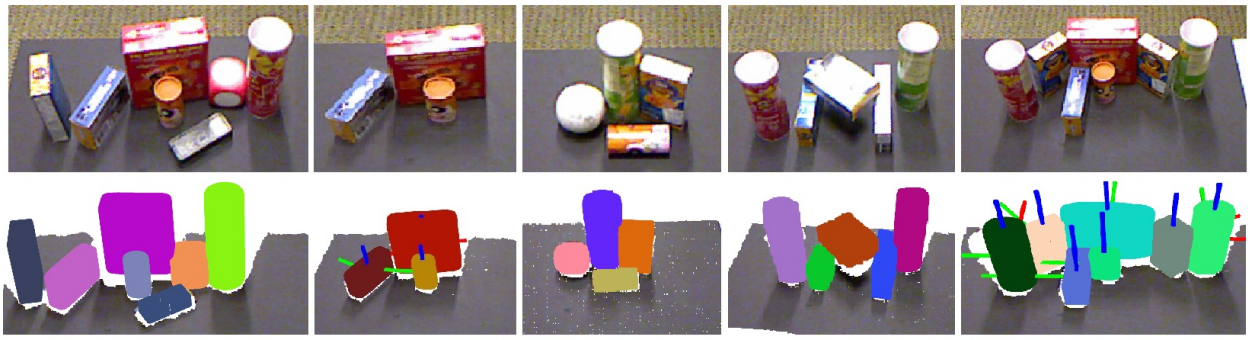


Figure 6.2.2: Superquadrics fitting in 5 different cluttered scenes. The detected poses of the objects by iteration methods are shown in b and e

Table 6.2.1: Single object parameters. Method 1 is Mirroring and SQ Fitting, method 2 by Quispe et al. [109]. Method 3 is PCA and superquadric fitting Average time in seconds.

Object	Method	a_1	a_2	a_3	e_1	e_2	Avg. time	Avg. error
toy cylinder	1	0.023	0.024	0.15	0.389	1.031	0.535	9.2%
	2	0.022	0.022	0.15	0.561	0.928	1.788	9.78%
	3	0.025	0.017	0.14	0.52	0.88	2.121	10.24%
ball	1	0.069	0.09	0.045	1.003	0.95	0.78	5.23%
	2	0.43	0.04	0.043	0.986	0.972	1.23	6.24%
	3	0.62	0.07	0.051	1.01	0.91	2.41	8.44%
Cheese Box	1	0.07	0.026	0.13	0.1557	0.319	0.82	8.41%
	2	0.06	0.023	0.15	0.1329	0.417	1.18	9.6%
	3	0.047	0.025	0.12	0.12	0.35	2.33	11.18%
Large Dice	1	0.05	0.05	0.479	0.603	0.606	0.56	6.77%
	2	0.054	0.054	0.052	0.629	0.613	0.92	6.28%
	3	0.06	0.064	0.058	0.57	0.62	2.02	9.77%
Pringles Box	1	0.012	0.012	0.168	0.355	1.28	0.95	7.48%
	2	0.016	0.016	0.162	0.372	1.315	1.24	8.21%
	3	0.011	0.014	0.151	0.36	1.22	2.24	10.31s%
Coke Can	1	0.041	0.043	0.172	0.41	1.12	0.61	9.5%
	2	0.044	0.044	0.15	0.55	1.08	1.28	9.8%
	3	0.021	0.047	0.14	0.51	1.14	2.07	11.12%
Puzzle Box	1	0.05	0.016	0.19	0.144	0.323	0.65	7.1%
	2	0.047	0.014	0.16	0.121	0.37	0.92	7.4%
	3	0.061	0.017	0.21	0.141	0.42	1.13	9.1%
Lego Set1	1	0.031	0.04	0.18	0.12	0.55	0.52	6.82%
	2	0.044	0.042	0.163	0.08	0.667	0.88	8.13%
	3	0.037	0.045	0.168	0.15	0.428	1.33	11.41%
Green Ball	1	0.033	0.031	0.045	1.01	0.97	0.44	7.43%
	2	0.012	0.035	0.025	0.99	0.91	0.81	9.71%
	3	0.037	0.032	0.03	1.13	0.82	1.31	11.37%
Chocolate Box	1	0.09	0.07	0.068	0.16	0.526	0.72	7.3%
	2	0.11	0.062	0.05	0.18	0.435	1.44	9.1%
	3	0.094	0.06	0.062	0.164	0.552	2.17	9.4%

To fit superquadrics on multiple objects present in cluttered scenario, different objects are chosen randomly for 5 different scenes. The 5 scenes are presented in the Fig.6.2.2. The scenes cannot be considered as densely cluttered as some parts of the objects can be seen from the camera point-of-view. The scenes can be considered as semi-cluttered environment. It is noticeable, for a couple of scenes such as $Scene2$ the part of the cheesebox is not visible as the toy cylinder is situated before the cheesebox and is occluding the front surface. Still, the superquaric model is fitted properly on the cheesebox as the mirroring of the cheesebox model generated enough points to approximate a box-alike structure. The fitting of superquadrics in semi-cluttered scenario is highly dependent on the object clustering algorithm. If the system cannot classify different objects by the segmentation, the fitting process leads to erroneous fitting.

6.3 Experiments with WAM Arm

For the experiments, a WAM arm and a custom hook gripper is used. The mechanism of the gripper is similar to parallel-jaw grippers while the closing plates are hooks instead of flat plates. The gripper used for this experiments are shown in Fig.6.3.2a. WAM is a 7DOF arm while the gripper has 1 DOF. The frames of the arm, Kinect and gripper are shown in Fig.6.3.2b.

The sensor in the experiments is a Kinect1 attached upward to the system pointing to the table where the objects are located. The goal of the robot is to detect the objects by the Kinect device, fit superquadric models to the object, find grasps on them, pick up the objects and place them on the left of the robot. The motion planner plans a motion to carry the gripper to the approach pose. A cartesian controller is implemented which follows the approach direction and reaches the gripper to the final pose. After picking up, the robot follows a joint controller to place the objects. If the robot successfully picks and places the object in the designated place, the trial is a success, otherwise, it is a failure.

The objects used in the single object grasping experiments are mainly apple, banana, coke can, chocolate box, pringles can, puzzle box. The typical scenario where the object is picking up the apple is shown in Fig.6.3.1. Three different locations are designated at the objects are kept on the location for each trial. The average time and success rate for each trial is shown in Table.6.3.1. The success rate for the banana is low, due to the fact that the superquadric fitting on the banana was not proper, which led to incorrect grasping in all of the locations.

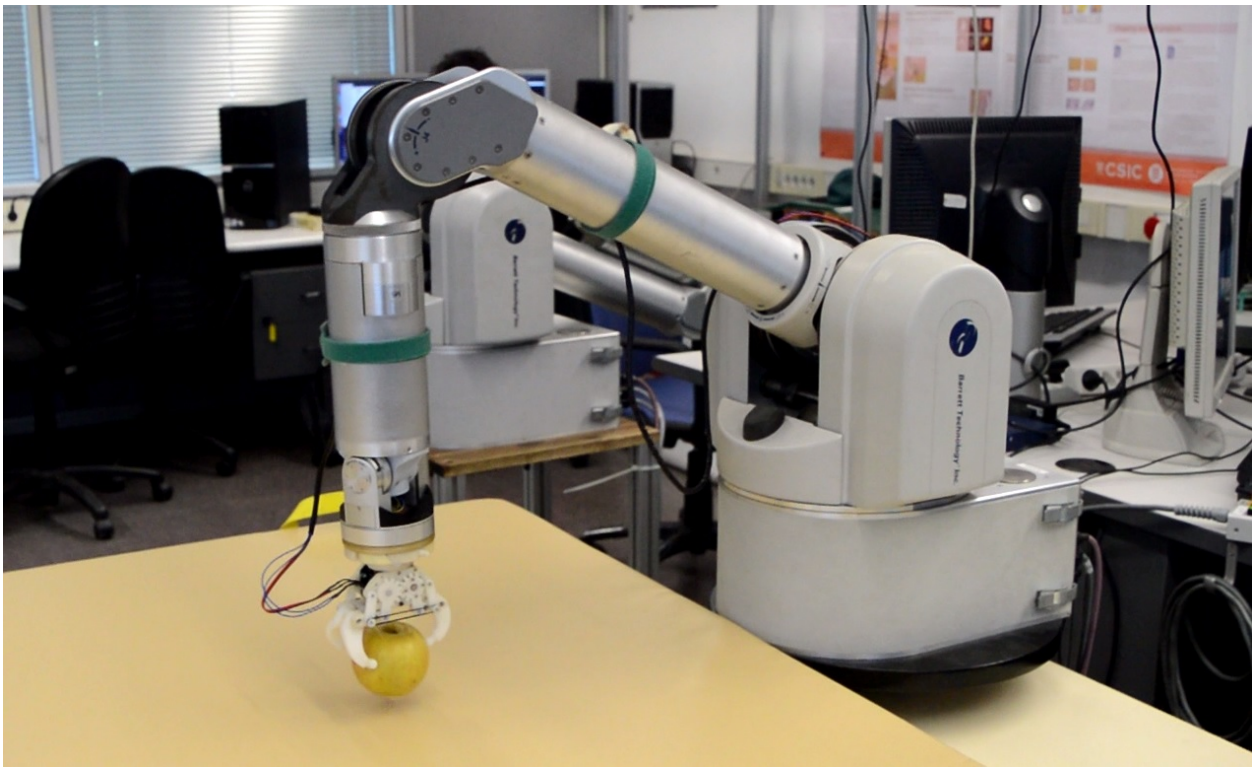


Figure 6.3.1: WAM robot single object

Table 6.3.1: Grasping single object WAM

Object	Loc. 1	Loc. 2	Loc. 3	Avg. Time	succeess Rate (%)
Apple	3/3	2/3	3/3	7.13s	88.89
Banana	3/3	1/3	2/3	7.05s	66.67
Coke Can	2/3	3/3	3/3	7.21s	88.89
Chocolate Box	3/3	3/3	3/3	7.23s	100.
Pringles Can	3/3	3/3	2/3	8.17s	88.89
Puzzle Box	2/3	3/3	2/3	7.71s	77.78

Table 6.3.2: Table clearing WAM

Test case	Number of objects	Success	Total time(seconds)
TC1	7	5/7	103s.
TC2	3	2/3	53s.
TC3	5	3/5	72s.
TC4	5	2/3	87s.
TC5	3	1/3	45s.

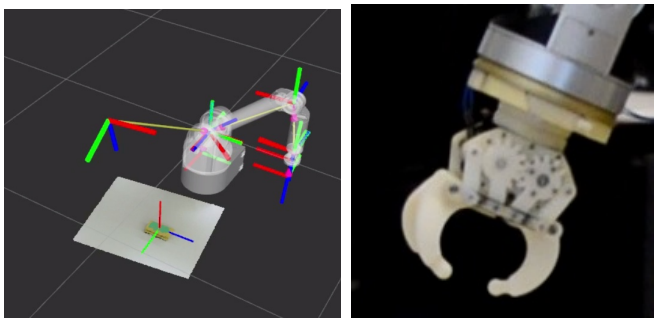


Figure 6.3.2: The WAM robot setup. a) The frames of the WAM arm and the pose of detected object, b) The custom hook gripper

Figure 6.3.3: WAM robot multiple objects

The chocolate box has been grasped properly from each location.

Grasping multiple objects is purely a table clearing task, where randomly chosen multiple objects are placed on the table and the task of the robot is to clear the table off all objects. A full trial consists of the task of robot picking and placing each object in a designated place one by one. The trail ends when either all of the objects are placed or the system stops between the experiment. Each object can be tried only one. If the robot cannot grasp the

object or fails to deliver to place it, it can be considered as a failure. A grasping multiple object scenario is presented in Fig.6.3.3. The results of grasping multiple objects with the success rate are presented in Table.6.3.2. The time shown is the full time for the trial. It is easily visible the time taken for the trial increases with the number of objects present in the scene. It is due to the fact that after grasping and placing one object, the system again starts fitting for rest of all the objects present in the table. Because after moving one object, it is not guaranteed that the pose of the rest of the objects remains same.

6.4 Experiments with UR5 Arm

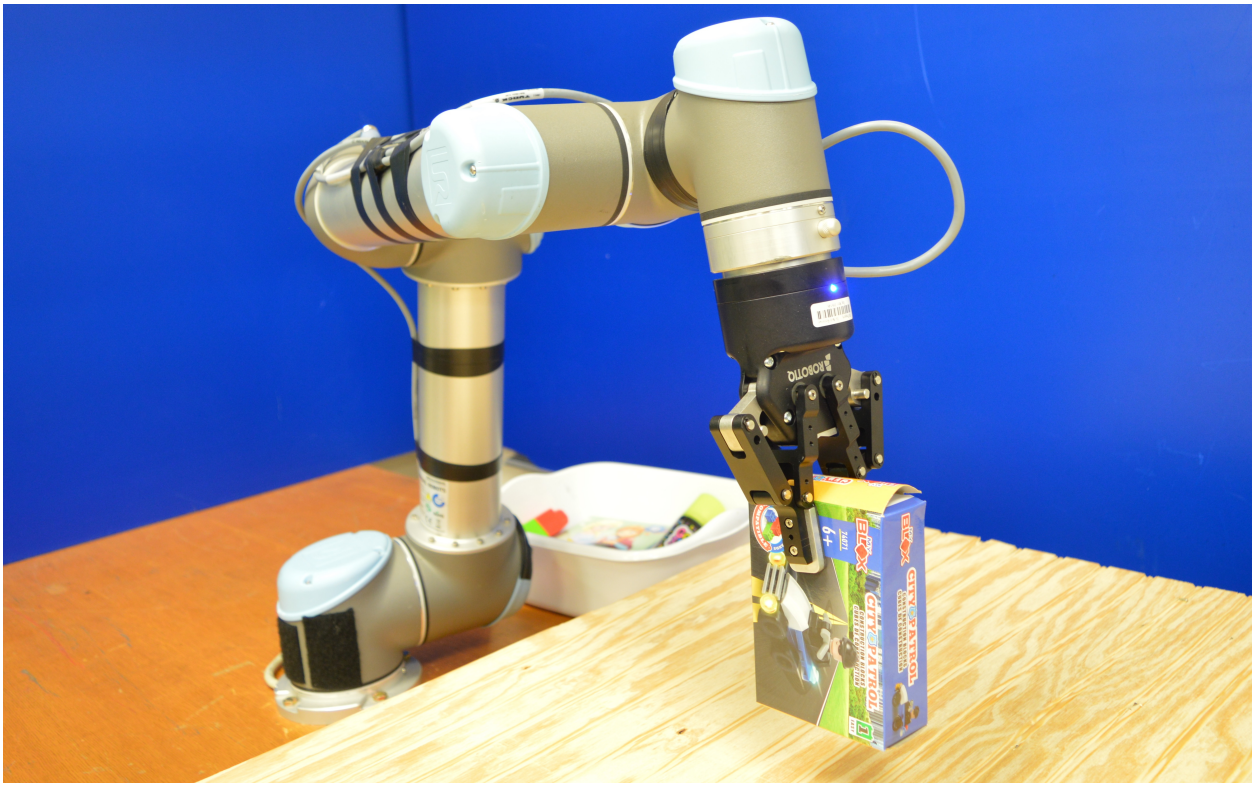


Figure 6.4.1: UR5 robot and Robotiq 2-finger gripper in the grasping single object scenerio

The setup for the experiment with the UR5 arm is shown in Fig.6.4.1 where the robot is picking up object *Box1*. The robot in the experiments is a UR5 arm with 6DOF and the gripper is a robotiq 2-finger parallel-jaw gripper with 1DOF. This robot is specifically chosen as it has 1 less DOF than other robot and the reachability of the robot is completely different from other two robots used in our other experiments. The sensor in the experiments is Kinect2 which has a much higher resolution that Kinect1. Due to calibration issues, the depth data from Kinect2 device is much error-prone. So background of the environment is

colored blue. The Kinect is placed in front of the table instead of putting it on the top of the table like the previous experiment. In the previous experiment, the sensor could obtain a top view of the scene. For the multiple object scenes, the workspace is chosen compact enough to only include the table and the objects upon the table.

Table 6.4.1: Grasping single object UR5

Object	Loc. 1	Loc. 2	Loc. 3	Avg. Time	succeess Rate (%)
Box1	3/3	3/3	3/3	6.14s	100.
Goofy String Cylinder	2/3	2/3	3/3	6.08s	77.78
Toy cylinder	3/3	3/3	3/3	6.01s	100.
Puzzle Box	3/3	2/3	2/3	6.52s	77.78
Lego1	3/3	2/3	3/3	6.13s	88.89
Yellow Ball	3/3	3/3	3/3	6.07s	100.
Red Ball	3/3	2/3	3/3	6.11s	88.89
Kleenex Box	2/3	2/3	2/3	6.55s	66.68

Similar to the previous experiment for single object experiment, the objects are placed in isolation to three different places on the table. For every trial, the objects are grasped three times. The success rate and average time for grasping each object is shown in Table.6.4.1. The objects used in the previous experiment is not used here. The object list for this experiment is completely different. A box is placed at the left of the robot. The task of the robot is to pick the objects and place them in the box.

Table 6.4.2: Table clearing UR5

Test case	Number of objects	Success	Total time(seconds)
TC1	5	5/5	78s.
TC2	8	4/8	113s.
TC3	5	4/5	81s.
TC4	4	3/4	65s.
TC5	5	4/5	87s.

For grasping multiple approach experiments similar to the previous experiment with the WAM arm is performed. The objects are chosen randomly and place on the table in a semi-cluttered way. An example of the scene ($TC2$) with 8 objects is presented in the Fig.6.4.2. The results from the table clearing experiment are shown in Table.6.4.2.

Figure 6.4.2: UR5 robot and Robotiq 2-finger gripper in the table clearing scenerio

6.5 Experiments with PR2 Robot

In this section, the grasping experiments with PR2 robot are presented. PR2 robot is a mobile manipulator from Willow Garage with two 7DOF arms. For our experiments, we have used the left hand of the PR2. The sensor used in the experiments is a Kinect1 attached to the head of the robot. The experimental setup is presented in Fig.6.5.1. Similar to the previous experiments, for single object grasping scenario, the objects are placed to three different locations on the table and for every trial, the robot had to pick up the objects from the table and place them in a box on the left of the robot. The motion planner takes care of carrying the gripper to the approach pose and a custom cartesian planner is incorporated to travel the approach distance towards the grasp pose. After picking up the object, the robot had to drop the object by a joint motion planner.

The table is considered as a collision object, which is shown in the Fig.6.5.2. The grasps identified for three objects are also shown. The single object grasping results are compared with two recent methods HaF (Height Accumulate Features)[131] and Agile[46] shown in Ta-

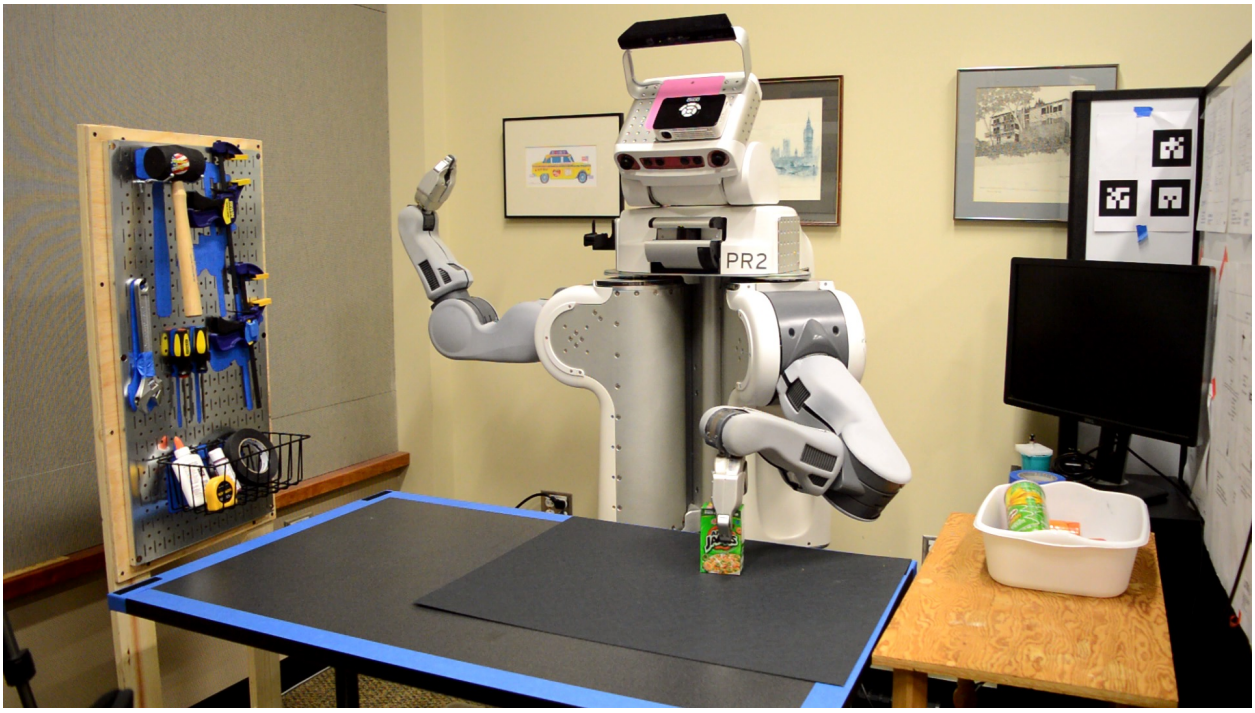


Figure 6.5.1: PR2 robot and environment setup

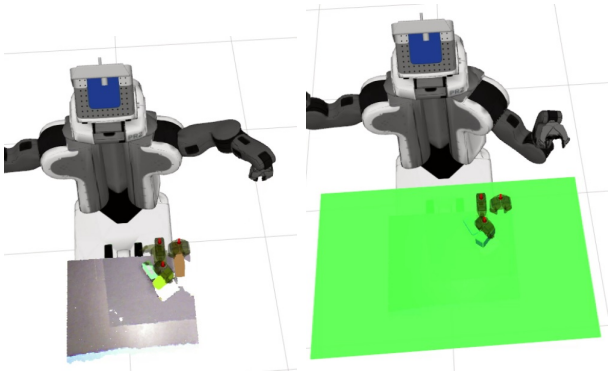


Figure 6.5.2: The tabletop setup for the experiments with the PR2. a) The typical experiment scenerio in visualization, b) The collision table shown in Green and the Grasps identified by the system. The

ble.6.5.1. Grasping different objects for the single object experiments are shown in Fig.6.5.3.

For dense cluttered scenes, 5 experiments are performed and for all the experiments and the objects are kept the same for individual experiments. The results of grasping in clutter comparing with the other two methods are presented in Fig.6.5.4. As our system does not rely on iteration for finding grasps to execute, it provides better results in time efficiency than other methods. For single objects, location 2 is the place where every method struggled. The location is chosen deliberately far away from the camera frame, where the segmentation process is prone to failure. As the Agile method is not dependent on segmentation, it

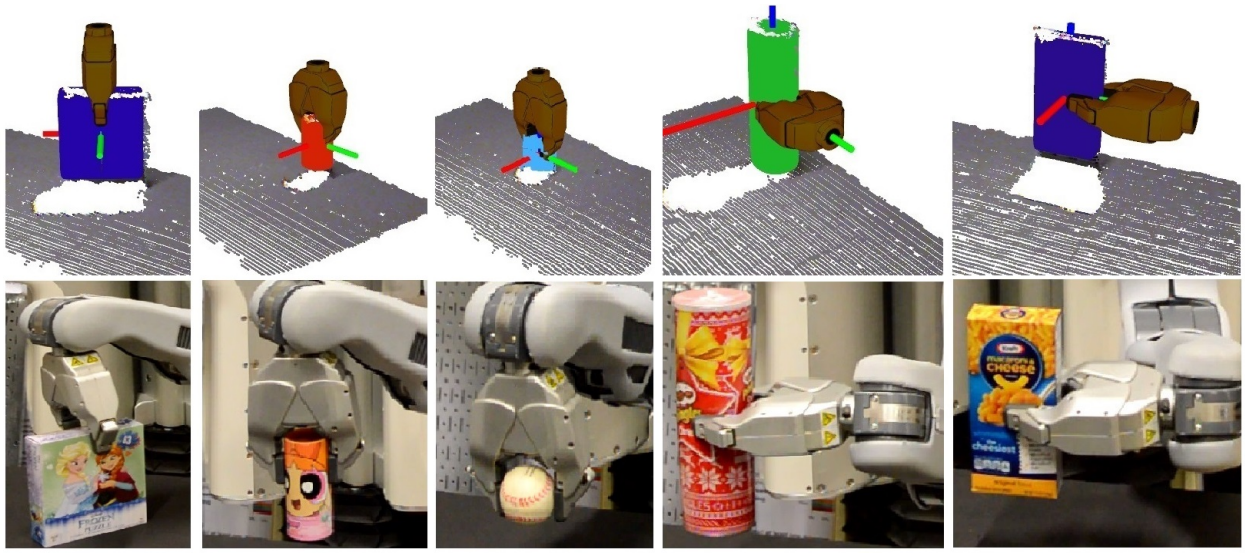


Figure 6.5.3: PR2 grasping single objects

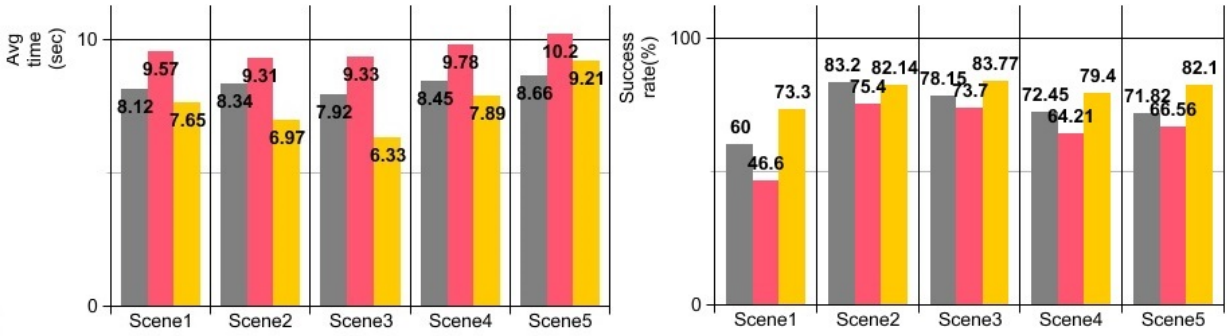


Figure 6.5.4: Grasping result compared with methods presented in [131] and [46]. From left to right: a) avg. time and b)success rate for individual objects. Gray bars are [131], Red bars are [46] and while yellow bars are by Mirroring and Superquadric fitting.

provides a better result for only one object (ball). It failed significantly for the rest of the objects. For cluttered scenes, our system consumes much more time for some scenarios than the other scenarios, as those contain the highest number of objects in a given scene. In experiment 1, though our method calculated all the grasp poses, the dice slipped from the gripper, affecting the accuracy. The vertical placement of the camera where it perceives the environment from the top explains the reason for the significantly poor performance of the Haf grasping method. As for the Agile grasping, the execution system of agile grasping is not based on most vertical grasps, so though in most of the cases, the objects are perceived properly for good grasps, it still failed to execute them.

Table 6.5.1: PR2 grasping single object

Object	Method	Loc. 1	Loc. 2	Loc. 3	Avg. Time	succeess Rate (%)
Ball	Agile	3/3	2/3	3/3	4.24s	88.89
	Haf	3/3	1/3	2/3	6.71s	66.67
	Superquadric	3/3	2/3	3/3	3.02s	88.89
Puzzle Box	Agile	2/3	1/3	2/3	4.57s	55.56
	Haf	2/3	1/3	1/3	6.32s	44.45
	Superquadric	2/3	1/3	2/3	3.21s	66.67
Toy Cylinder	Agile	2/3	1/3	3/3	3.78s	66.67
	Haf	2/3	0/3	2/3	6.45s	44.45
	Superquadric	3/3	2/3	3/3	3.252s	88.89
Marker Eraser	Agile	1/3	1/3	2/3	4.11s	44.45
	Haf	2/3	0/3	1/3	6.23s	33.34
	Superquadric	3/3	1/3	3/3	3.16s	77.78

6.6 Summary

In this chapter, the experimental setup and results are presented. This chapter discusses the results from the reachability library (Reuleaux) with two other reachability mapping approaches. The superquadric fitting results on objects in isolation and in clutter are discussed. For the grasping experiments, three different scenarios with three different robots are presented. For all the scenarios, grasping in isolation and clutter is discussed. For grasping with the PR2 robot, the results are compared with two well-recognized grasping methods for unknown objects.

Chapter 7: Conclusion and Outlook

Robots are stepping inside daily-life human environment, leaving the days behind when they were only meant to be in the industries doing repetitive tasks. There are still many aspects that need to be resolved, specifically in the field of grasping and manipulation, because contradictory to the industrial scenario, the environment where humans live is very uncertain. The objects that need to be manipulated are the same objects used daily by the human, coming in different varieties, types, shapes, and sizes. So it is impractical to maintain a huge database which can contain all the objects in a household environment. There will always be some objects that are unknown. So the most feasible choice is to develop a grasping system that can directly grasp novel objects instead of searching them in the database. The key contribution of the thesis approaches that problem of grasping novel objects by low-cost sensor data quickly and efficiently.

7.1 Summary

This thesis presents an approach towards grasping unknown objects in isolated as well as in cluttered scenario. The method proposed does not rely on the construction of databases. The main contribution of the work is a superellipsoid fitting algorithm that fits models on point cloud data in real time. The variation in the fitting algorithm creates a mirrored surface as a pre-processing step which approximates the occluded side. The pose estimation is performed in a separate stage which reduces the load to the model parameter searching phase. Later the grasps are directly searched on the parameterized superellipsoid model instead of exploring the feature space of the point clouds. Fast and efficient antipodal grasp synthesis is incorporated which ensures grasp execution by filtering the grasp hypotheses with reachability maps. Experiments are performed with objects used daily in the human environment, comparing the efficiency of the method in terms of success and time with state-of-the-art methods.

The benefits of modeling by superellipsoid representation are explored in such a way where a single mathematical model can represent multiple shapes and sizes, instead of fitting different representation in different scenes. The parameters of the model serve as the ingredient of the grasping phase which plays a vital role in determining the curvature of the surface and finally leads to an optimized gripper opening, grasp pose and approach direction.

The approach to the grasping problem is robust to dynamically changing environment, not limited only to theoretical practices or simulations. The extension of grasp hypothesis filtering by reachability maps ensures the execution of grasps based on robot kinematic

structure. The utilization of reachability maps plays a vital role in determining which object to grasp based on the reachability of the object in the scene.

Several tests are performed on three different robotic platforms with distinct kinematic structure of the robot and gripper designs ensuring hardware scalability. The tests are mainly categorized into grasping objects in isolation and clearing table with multiple objects.

A complete grasping system is presented which takes point cloud of the scene and gripper parameters as inputs and provides outputs in terms of grasps that can be directly executed on the robot. The works presented in the thesis is openly available to the open-source community.

7.2 Contributions to the Research Problem

The overall goal of the research is to investigate the problem of grasping unknown objects, by computing a gripper pose, the opening angle of the gripper and an approach towards the object which ensures safe and reliable grasping. Though grasping in household environments poses different problems based on different scenarios, the thesis tries to contribute to four different problems:

1. Modeling point cloud data to object model
2. Grasping unknown object in isolation
3. Grasping unknown object in clutter
4. Ensuring safe and reliable grasping

There are various approaches that try to model point cloud sensor data by mathematical models. The approach presented in the thesis enables robotic systems to:

- fit a single model in all scenarios: The mathematical model utilized here is a superellipsoid model, that has the capability of representing all the basic symmetric shapes such as spheres, cylinders, boxes and many more enabling a generalization in the modeling phase.
- fit model in real time: The approach tackles the problems of finding pose and fitting model in different threads where each thread takes care of their functionalities and merge them together towards a parameterized model in a quick and efficient way.
- approximate the occluded side: A mirroring algorithm is presented which constructs the occluded side of the object model which cannot be perceived by a sensor capable of capturing a partial-view model.

The problem of grasping unknown object is still an unresolved problem. The thesis can be a stepping stone to solving the problem enabling following services:

- discard the process of constructing a huge database: The approach estimates the object model from the point cloud data before grasping. So there is no obligation towards creating a database of objects beforehand and the grasps can be found directly from the scene.
- find grasps by mathematical model: The grasping synthesis is directly performed on the superellipsoid instead of the point cloud data. The geometry and shape information extracted by the superellipsoids enables defining the antipodal points for grasping.

Grasping unknown objects in clutter or clearing a surface with objects that are close to each other is a significantly hard problem as the system has to perform multiple grasps to complete the task and removing a single object, changes the location and features of the rest of the objects. This thesis can help in performing the tasks of:

- segmenting multiple objects and find grasps on each of them: After the objects are segmented in different clusters, all the clusters are fitted to individual models and all the models are searched for grasps in a parallelized process.
- finding grasps multiple times: After one object is removed, the process resets itself and start from the beginning of searching for grasps on rest of the objects. So the grasps defined in the previous iteration are discarded, as they may become invalid in the current iteration.

It is not enough just to find grasp hypotheses that can be a representation of grasps for certain object. The system should find reliable grasps that can be executed on real systems. The thesis provides approaches to the execution of grasps enabling:

- incorporation of reachability information: The reachability map presented in this thesis combines the kinematic information of the robot with the grasping phase. This ensures that the grasps found by the system are reachable to the robot.
- safe and reliable grasp execution: It is the duty of the motion planner of the manipulator for a safe grasp execution. While searching for an optimized grasp, the system includes constraints such as distance to table for collision avoidance and grasping close to the center of the object to care for moment of inertia, by which the object does not wobble in the hand. This constraints posed by the grasping system reduces the load on the motion planner and leads to safe execution.

7.3 Outlook

Robotic grasping in unstructured environment is an interesting and vast field of research. Grasping known objects is solved to some limits with various constraints. But robotic grasping for unknown objects is far away from a concrete solution. The author believes that this thesis may work as a stepping stone towards the solution. There are some attainable aspects of the research that are not discussed in this thesis.

A further potential extension to the work is interactive grasping, where the user can point to the object to be grasped instead of grasping every object from a pile of objects. The model fitting and searching for antipodal grasping points are limited to symmetric simplistic objects. For objects with asymmetry, multiple models should be fitted and merged to construct a complete model which can fit those objects. These models should not be only a superquadric. A combination of a superquadric and a supertoroid may be incorporated e.g. for fitting a cup; while the handle of the cup can be fitted with supertoroid and the cup is fitted with a superquadric. Segmentation is still one of the bottlenecks in the problem. Incorrect segmentation of the object may lead to incorrect fitting finally leading to erroneous grasping, causing damage to the system and even to the human co-worker.

References

[1] KDL kinematics and dynamics library (kdl). http://wiki.ros.org/kdl. Accessed: 2017-07-11.

[2] U. Klank A. Maldonado and M. Beetz. Robotic grasping of unmodeled objects using time-of-flight range data and finnger torque information. In *International Conference on Intelligent Robots and Systems (IROS), 2010*, pages 995–1001, 2010. doi: 0.1109/IROS.2010.5649185.

[3] P. Abelha, F. Guerin, and M. Schoeler. A model-based approach to finding substitute tools in 3d vision data. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2471–2478, May 2016. doi: 10.1109/ICRA.2016.7487400.

[4] Alba Perez Gracia Abhijit Makhal. Solvable multi-fingered hands for exact kinematic synthesis. In *Advances in Robot Kinematics*, 2014. URL https://doi.org/10.1007/978-3-319-06698-1_16.

[5] Alba Perez Gracia Abhijit Makhal, Federico Thomas. Grasping unknown objects in clutter by superquadric representation. In *Proceedings. 2nd IEEE International Conference on Robotic Computing*, pages 292–299, January 2018.

[6] Alex K.Goins Abhijit Makhal. Reuleaux: Robot base placement by reachability analysis. In *Proceedings. 2nd IEEE International Conference on Robotic Computing*, pages 137–142, January 2018.

[7] Ilya Afanasyev, Luca Baglivo, and Mariolino De Cecco. 3d object localization using superquadric models with a kinect sensor. 2011.

[8] T. Asfour, P. Azad, N. Vahrenkamp, K. Regenstein, A. Bierbaum, K. Welke, J. Schröder, and R. Dillmann. Toward humanoid manipulation in human-centred environments. *Robotics and Autonomous Systems*, 56(1):54 – 65, 2008. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2007.09.013. URL http://www.sciencedirect.com/science/article/pii/S0921889007001339. Human Technologies: "Know-how".

[9] Justin Kearns Chioma Osondu Ashutosh Saxena, Justin Driemeyer and Andrew Y. Ng. Learning to grasp novel objects using vision. In *10th International Symposium of Experimental Robotics,*, 2006.

[10] Pedram Azad, Tamim Asfour, and R. Dillmann. Stereo-based 6d object localization for grasping with humanoid robot systems. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 919–924, Oct 2007. doi: 10.1109/IROS.2007.4399135.

[11] T. Baier-Lowenstein and Jianwei Zhang. Learning to grasp everyday objects using reinforcement-learning with automatic value cut-off. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1551–1556, Oct 2007. doi: 10.1109/IROS.2007.4399053.

[12] Ruzena Bajcsy and Franc Solina. Three dimensional object representation revisited. In *Proceedings. First International Conference on Computer Vision*, June 1987.

[13] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka. Human-guided grasp measures improve grasp robustness on physical robot. In *2010 IEEE International Conference on Robotics and Automation*, pages 2294–2301, May 2010. doi: 10.1109/ROBOT.2010.5509855.

[14] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka. Physical human interactive guidance: Identifying grasping principles from human-planned grasps. *IEEE Transactions on Robotics*, 28(4):899–910, Aug 2012. ISSN 1552-3098. doi: 10.1109/TRO.2012.2189498.

[15] A. H. Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, Jan 1981. ISSN 0272-1716. doi: 10.1109/MCG.1981.1673799.

[16] A. Bicchi and V. Kumar. Robotic grasping and contact: a review. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 348–353 vol.1, 2000. doi: 10.1109/ROBOT.2000.844081.

[17] Antonio Bicchi. On the closure properties of robotic grasping. *The International Journal of Robotics Research*, 14(4):319–334, 1995. doi: 10.1177/027836499501400402. URL `https://doi.org/10.1177/027836499501400402`.

[18] G. Biegelbauer and M. Vincze. Efficient 3d object detection by fitting superquadrics to range image data for robot's object manipulation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1086–1091, April 2007. doi: 10.1109/ROBOT.2007.363129.

[19] J. Bohg and D. Kragic. Grasping familiar objects using shape context. In *2009 International Conference on Advanced Robotics*, pages 1–6, June 2009.

[20] J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesis -a survey. *IEEE Transactions on Robotics*, 30(2):289–309, April 2014. ISSN 1552-3098. doi: 10.1109/TRO.2013.2289018.

[21] Jeannette Bohg and Danica Kragic. Learning grasping points with shape context. *Robotics and Autonomous Systems*, 58(4):362 – 377, 2010. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2009.10.003. URL `http://www.sciencedirect.com/science/article/pii/S0921889009001699`.

[22] R. M. Bolle and D. B. Cooper. Bayesian recognition of local 3-d shape by approximating image intensity functions with quadric polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(4):418–429, July 1984. ISSN 0162-8828. doi: 10.1109/TPAMI.1984.4767547.

[23] F. Burget and M. Bennewitz. Stance selection for humanoid grasping tasks by inverse reachability maps. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5669–5674, May 2015. doi: 10.1109/ICRA.2015.7139993.

[24] F. Bernardini C. Bajaj and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 109–118, 1995.

[25] B. Calli, M. Wisse, and P. Jonker. Grasping of unknown objects via curvature maximization using active vision. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 995–1001, Sept 2011. doi: 10.1109/IROS.2011.6094686.

[26] S. Caselli, E. Faldella, B. Fringuelli, and F. Zanichelli. A hybrid system for knowledge-based synthesis of robot grasps. In *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 3, pages 1575–1581 vol.3, Jul 1993. doi: 10.1109/IROS.1993.583849.

[27] I-Ming Chen and J. W. Burdick. Finding antipodal point grasps on irregularly shaped objects. *IEEE Transactions on Robotics and Automation*, 9(4):507–512, Aug 1993. ISSN 1042-296X. doi: 10.1109/70.246063.

[28] Jen-Hui Chuang, Narendra Ahuja, Chien-Chou Lin, Chi-Hao Tsai, and Cheng-Hui Chen. A potential-based generalized cylinder representation. *Computers & Graphics*, 28(6):907–918, 2004.

[29] T. T. Cociaş, S. M. Grigorescu, and F. Moldoveanu. Multiple-superquadrics based object surface estimation for grasping in service robotics. In *2012 13th International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, pages 1471–1477, May 2012. doi: 10.1109/OPTIM.2012.6231780.

[30] J. Cornella and R. Suarez. On 2d 4-finger frictionless optimal grasps. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 4, pages 3680–3685 vol.3, Oct 2003. doi: 10.1109/IROS.2003.1249727.

[31] N. Curtis and Jing Xiao. Efficient and effective grasping of novel objects through learning and adapting a knowledge base. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2252–2257, Sept 2008. doi: 10.1109/IROS.2008.4651062.

[32] M. Kazemi J. A. Bagnell D. Katz, A. Venkatraman and A. Stent. Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. In *Robotics: Science and Systems Conference*, 2013.

[33] H. Dang, J. Weisz, and P. K. Allen. Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics. In *2011 IEEE International Conference on Robotics and Automation*, pages 5917–5922, May 2011. doi: 10.1109/ICRA.2011.5979679.

[34] R. Detry, D. Kraft, A. G. Buch, N. Krüger, and J. Piater. Refining grasp affordance models by experience. In *2010 IEEE International Conference on Robotics and Automation*, pages 2287–2293, May 2010. doi: 10.1109/ROBOT.2010.5509126.

[35] R. Detry, C. H. Ek, M. Madry, and D. Kragic. Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In *2013 IEEE International Conference on Robotics and Automation*, pages 601–608, May 2013. doi: 10.1109/ICRA.2013.6630635.

[36] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010. URL `http://www.programmingvision.com/rosen_diankov_thesis.pdf`.

[37] Jun Dong and J. C. Trinkle. Orientation-based reachability map for robot base placement. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1488–1493, Sept 2015. doi: 10.1109/IROS.2015.7353564.

[38] K. Duncan, S. Sarkar, R. Alqasemi, and R. Dubey. Multi-scale superquadric fitting for efficient shape and pose recovery of unknown objects. In *2013 IEEE International Conference on Robotics and Automation*, pages 4238–4243, May 2013. doi: 10.1109/ICRA.2013.6631176.

[39] H. Maas E. Schwalbe and F. Seidel. 3d building model generation from airborne laser scanner data using 2d gis data and orthogonal point cloud projections. *Proceedings of the International Society for Photogrammetry and Remote Sensing*, 3:12–14, 2005.

[40] S. Ekvall and D. Kragic. Learning and evaluation of the approach vector for automatic grasp generation and planning. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4715–4720, April 2007. doi: 10.1109/ROBOT.2007.364205.

[41] S. El-Khoury and A. Sahbani. A new strategy combining empirical and analytical approaches for grasping unknown 3d objects. *Robotics and Autonomous Systems*, 58 (5):497 – 507, 2010. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2010.01.008. URL `http://www.sciencedirect.com/science/article/pii/S0921889010000096`.

[42] Sahar El-khoury and Anis Sahbani. Handling objects by their handles. In *IROS Workshop on Grasp and Task Learning by Imitation*, 2008.

[43] Diego R. Faria, Ricardo Martins, Jorge Lobo, and Jorge Dias. A probabilistic framework to detect suitable grasping regions on objects. *IFAC Proceedings Volumes*, 45(22):247 – 252, 2012. ISSN 1474-6670. doi: https://doi.org/10.3182/20120905-3-HR-2030.00090. URL `http://www.sciencedirect.com/science/article/pii/S1474667016336187`. 10th IFAC Symposium on Robot Control.

[44] B. Faverjon and J. Ponce. On computing two-finger force-closure grasps of curved 2d objects. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 424–429 vol.1, Apr 1991. doi: 10.1109/ROBOT.1991.131614.

[45] C. Ferrari and J. Canny. Planning optimal grasps. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2290–2295 vol.3, May 1992. doi: 10.1109/ROBOT.1992.219918.

[46] D. Fischinger and M. Vincze. Empty the basket - a shape based learning approach for grasping piles of unknown objects. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2051–2057, Oct 2012. doi: 10.1109/IROS.2012.6386137.

[47] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL http://doi.acm.org/10.1145/358669.358692.

[48] David Fornas, Pedro J. Sanz, Josep M. Porta, and Federico THomas. Improving local symmetry estimations in rgb-d images by fiting superquadrics. In *XXXVII Jornadas de Automática*, Madrid (Spain), 09/2016 2016. ISBN 978-84-617-4298-1.

[49] Hiroshi Fukuda, Naohiro Fukumura, Masazumi Katayama, and Yoji Uno. Relation between object recognition and formation of hand shape: A computational approach to human grasping movements. *Systems and Computers in Japan*, 31(12):11–22, Oct 2000. ISSN 0882-4967. doi: 10.1002/1520-684X(20001115)31.

[50] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof. Grasp planning via decomposition trees. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4679–4684, April 2007. doi: 10.1109/ROBOT.2007.364200.

[51] Corey Goldfeder, Matei Ciocarlie, Hao Dang, and Peter K. Allen. The columbia grasp database. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, ICRA'09, pages 3343–3349, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-2788-8. URL http://dl.acm.org/citation.cfm?id=1703775.1703988.

[52] Marcus Gualtieri, Andreas ten Pas, Kate Saenko, and Robert Platt Jr. High precision grasp pose detection in dense clutter. *CoRR*, abs/1603.01564, 2016. URL http://arxiv.org/abs/1603.01564.

[53] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal. Template-based learning of grasp selection. In *2012 IEEE International Conference on Robotics and Automation*, pages 2379–2384, May 2012. doi: 10.1109/ICRA.2012.6225271.

[54] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, Apr 2013. ISSN 1573-7527. doi: 10.1007/s10514-012-9321-0. URL https://doi.org/10.1007/s10514-012-9321-0.

[55] K. Hsiao, L. P. Kaelbling, and T. Lozano-Perez. Grasping pomdps. In *Proceedings*

2007 IEEE International Conference on Robotics and Automation, pages 4685–4692, April 2007. doi: 10.1109/ROBOT.2007.364201.

[56] K. Hsiao, S. Chitta, M. Ciocarlie, and E. Gil Jones. Contact-reactive grasping of objects with partial shape information. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1228–1235, Oct 2010. doi: 10.1109/IROS.2010.5649494.

[57] Kaijen Hsiao, Paul Nangeroni, Manfred Huber, Ashutosh Saxena, and Andrew Y. Ng. Reactive grasping using optical proximity sensors. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, ICRA'09, pages 4230–4237, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-2788-8. URL `http://dl.acm.org/citation.cfm?id=1703775.1704128`.

[58] K. Huebner and D. Kragic. Selection of robot pre-grasps using box-based shape approximation. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1765–1770, Sept 2008. doi: 10.1109/IROS.2008.4650722.

[59] K. Huebner, K. Welke, M. Przybylski, N. Vahrenkamp, T. Asfour, D. Kragic, and R. Dillmann. Grasping known objects with humanoid robots: A box-based approach. In *2009 International Conference on Advanced Robotics*, pages 1–6, June 2009.

[60] J. B. Cherrie T. J. Mitchell W. R. Fright B. C. McCallum J. C. Carr, R. K. Beatson and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques,*, pages 67–76, 2001.

[61] G. Gruener J. Weingarten and R. Siegwart. A state-of-the-art 3d sensor for robot navigation. *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2160, 2004.

[62] Aleš Jaklič, Aleš Leonardis, and Franc Solina. *Segmentation and Recovery of Superquadrics: Computational Imaging and Vision.* Kluwer Academic Publishers, Norwell, MA, USA, 2000. ISBN 0-7923-6601-8.

[63] Yan-Bin Jia. Curvature-based computation of antipodal grasps. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 2, pages 1571–1577 vol.2, 2002. doi: 10.1109/ROBOT.2002.1014767.

[64] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. *2011 IEEE International Conference on Robotics and Automation*, pages 3304–3311, 2011.

[65] F. Brent Neal John C. Russ. *The Image Procesing Handbook Seventh Edition*. CRC Press, 2017. ISBN 9781138747494.

[66] A. Johnson. Spin-images: A representation for 3d surface matching. *PhD Thesis*, 1997.

[67] D. Kappler, J. Bohg, and S. Schaal. Leveraging big data for grasp planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311, May 2015. doi: 10.1109/ICRA.2015.7139793.

[68] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164, 2003.

[69] E. Klingbeil, D. Rao, B. Carpenter, V. Ganapathi, A. Y. Ng, and O. Khatib. Grasping with application to an autonomous checkout robot. In *2011 IEEE International Conference on Robotics and Automation*, pages 2837–2844, May 2011. doi: 10.1109/ICRA.2011.5980287.

[70] Gert Kootstra, Mila Popović, Jimmy Alison Jørgensen, Kamil Kuklinski, Konstantsin Miatliuk, Danica Kragic, and Norbert Krüger. Enabling grasping of unknown objects through a synergistic use of edge and surface information. *The International Journal of Robotics Research*, 31(10):1190–1213, 2012. doi: 10.1177/0278364912452621. URL `https://doi.org/10.1177/0278364912452621`.

[71] O. Kroemer, H. Ben Amor, M. Ewerton, and J. Peters. Point cloud completion using extrusions. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 680–685, Nov 2012. doi: 10.1109/HUMANOIDS.2012.6651593.

[72] J. D. Moriarty L. G. Shapiro. Sticks plates and blobs: A three-dimensional object representation for scene analysis. In *Proc. Annu. Nat. Conf. A.I*, pages 28–30, 1980.

[73] P. G. Mulgaonkar R. M. Haralick L. G. Shapiro, J. D. Moriarty. A generalized blob model for three-dimensional object representation. In *IEEE Workshop Pict. Data Descript. Management,*, 1980.

[74] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng. Learning to grasp objects with multiple contact points. In *2010 IEEE International Conference on Robotics and Automation*, pages 5062–5069, May 2010. doi: 10.1109/ROBOT.2010.5509508.

[75] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *Int. J. Rob. Res.*, 34(4-5):705–724, April 2015. ISSN 0278-3649. doi: 10.1177/0278364914549607. URL `http://dx.doi.org/10.1177/0278364914549607`.

[76] Beatriz León, Stefan Ulbrich, Rosen Diankov, Gustavo Puche, Markus Przybylski, Antonio Morales, Tamim Asfour, Sami Moisio, Jeannette Bohg, James Kuffner, and Rüdiger Dillmann. *OpenGRASP: A Toolkit for Robot Grasping Simulation*, volume 6472 of *Lecture Notes in Computer Science*, pages 109–120. Springer Berlin / Heidelberg, 2010.

[77] Ying Li and N. S. Pollard. A shape matching algorithm for synthesizing humanlike enveloping grasps. In *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, pages 442–449, Dec 2005. doi: 10.1109/ICHR.2005.1573607.

[78] Z. Li and S. S. Sastry. Task-oriented optimal grasping by multifingered robot hands. *IEEE Journal on Robotics and Automation*, 4(1):32–44, Feb 1988. ISSN 0882-4967. doi: 10.1109/56.769.

[79] Vincenzo Lippiello, Fabio Ruggiero, Bruno Siciliano, and Villani Luigi. Preshaped visual grasp of unknown objects with a multi-fingered hand. In *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 5894 – 5899, 11 2010.

[80] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, Norwell, MA, USA, 2017. ISBN 9781107156302.

[81] D. Manocha and Yunshan Zhu. A fast algorithm and system for the inverse kinematics of general serial manipulators. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3348–3353 vol.4, May 1994. doi: 10.1109/ROBOT.1994.351055.

[82] M. Vidyasagar Mark W. Spong. *Robot Dynamics and Control*. Wiley, 1989. ISBN 9780471612438.

[83] D.; Blodow N.; Kleinehellefort J.; Beetz M. Marton, Zoltan-Csaba; Pangercic. General 3d modelling of novel objects from a single view. *Proc IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3700–3705, 2010.

[84] Z. C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz. General 3d modelling of novel objects from a single view. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3700–3705, Oct 2010. doi: 10.1109/IROS.2010.5650434.

[85] Matthew T. Mason and J. Kenneth Salisbury, Jr. *Robot Hands and the Mechanics of Manipulation*. MIT Press, Cambridge, MA, USA, 1985. ISBN 0-262-13205-2.

[86] A. T. Miller and P. K. Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics Automation Magazine*, 11(4):110–122, Dec 2004. ISSN 1070-9932. doi: 10.1109/MRA.2004.1371616.

[87] Andrew T. Miller, Steffen Knoop, Henrik I. Christensen, and Peter K. Allen. Automatic grasp planning using shape primitives. In *ICRA*, 2003.

[88] B. Mirtich and J. Canny. Easily computable optimum grasps in 2-d and 3-d. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 739–747 vol.1, May 1994. doi: 10.1109/ROBOT.1994.351399.

[89] N. J. Mitra, L. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (SIGGRAPH)*, 25(3):560–568, 2006.

[90] J. Müller, U. Frese, and T. Röfer. Grab a mug - object detection and grasp motion planning with the nao robot. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 349–356, Nov 2012. doi: 10.1109/HUMANOIDS.2012.6651543.

[91] A. Morales, G. Recatala, P. J. Sanz, and A. P. del Pobil. Heuristic vision-based computation of planar antipodal grasps on unknown objects. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 1, pages 583–588 vol.1, 2001. doi: 10.1109/ROBOT.2001.932613.

[92] A. Morales, T. Asfour, P. Azad, S. Knoop, and R. Dillmann. Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5663–5668, Oct 2006. doi: 10.1109/IROS.2006.282367.

[93] M. H. Mousa, R. Chaine, S. Akkouche, and E. Galin. Efficient spherical harmonics representation of 3d objects. In *Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on*, pages 248–255, Oct 2007. doi: 10.1109/PG.2007.39.

[94] RD Howe MR.Cutkosky. Human grasp choice and robotic grasp analysis. *Dexterous Robot Hands*, (1), 5–31.

[95] V. D. Nguyen. Constructing force-closure grasps. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1368–1373, Apr 1986. doi: 10.1109/ROBOT.1986.1087483.

[96] A. Troccoli B. Smith M. Leordeanu P. Allen, I. Stamos and Y. Hsu. 3d modeling of historic sites using range and image data. *Proceedings of the 2003 IEEE Inter- national Conference on Robotics and Automation*, 1:145–150, 2003.

[97] J. Pan, S. Chitta, and D. Manocha. Fcl: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pages 3859–3866, May 2012. doi: 10.1109/ICRA.2012.6225337.

[98] Chavdar Papazov, Sami Haddadin, Sven Parusel, Kai Krieger, and Darius Burschka. Rigid 3d geometry matching for grasping of known objects in cluttered scenes. *Int. J. Rob. Res.*, 31(4):538–553, April 2012. ISSN 0278-3649. doi: 10.1177/0278364911436019. URL `http://dx.doi.org/10.1177/0278364911436019`.

[99] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, June 2013. doi: 10.1109/CVPR.2013.264.

[100] R. Pascoal, V. Santos, C. Premebida, and U. Nunes. Simultaneous segmentation and superquadrics fitting in laser-range data. *IEEE Transactions on Vehicular Technology*, 64(2):441–452, Feb 2015. ISSN 0018-9545. doi: 10.1109/TVT.2014.2321899.

[101] Nicholas H.E. Pillow. *Recognition of Generalized Cylinders Uisng Geometric Invariance*. PhD thesis, Oriel College, Michaelmas Term, 1999.

[102] Maurizio Pilu and Robert B. Fisher. Equal-distance sampling of superellipse models. In *Proceedings of the 1995 British Conference on Machine Vision (Vol. 1)*, BMVC '95, pages 257–266, Surrey, UK, UK, 1995. BMVA Press. ISBN 0-9521898-2-8. URL `http://dl.acm.org/citation.cfm?id=236190.236215`.

[103] Nancy S. Pollard. Closure and quality equivalence for efficient synthesis of grasps from examples. *The International Journal of Robotics Research*, 23(6):595–613, 2004. doi: 10.1177/0278364904044402. URL `https://doi.org/10.1177/0278364904044402`.

[104] M. Popović, G. Kootstra, J. A. Jørgensen, D. Kragic, and N. Krüger. Grasping unknown objects using an early cognitive vision system for general scene understanding. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 987–994, Sept 2011. doi: 10.1109/IROS.2011.6094932.

[105] Mila Popović, Dirk Kraft, Leon Bodenhagen, Emre Başeski, Nicolas Pugeault, Danica Kragic, Tamim Asfour, and Norbert Krüger. A strategy for grasping unknown objects based on co-planarity and colour information. *Robotics and Autonomous Systems*, 58 (5):551 – 565, 2010. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2010.01.003. URL `http://www.sciencedirect.com/science/article/pii/S0921889010000047`.

[106] Oliver Porges, Roberto Lampariello, Jordi Artigas, Armin Wedler, Christoph Borst, and Máximo A. Roa. Reachability and dexterity: Analysis and applications for space robotics. 2015.

[107] M. Przybylski, T. Asfour, and R. Dillmann. Unions of balls for shape approximation in robot grasping. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1592–1599, Oct 2010. doi: 10.1109/IROS.2010.5653520.

[108] M. Przybylski, T. Asfour, and R. Dillmann. Planning grasps for robotic hands using a novel object representation based on the medial axis transform. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1781–1788, Sept 2011. doi: 10.1109/IROS.2011.6094937.

[109] A. H. Quispe, B. Milville, M. A. Gutiérrez, C. Erdogan, M. Stilman, H. Christensen, and H. B. Amor. Exploiting symmetries and extrusions for grasping household objects. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3702–3708, May 2015. doi: 10.1109/ICRA.2015.7139713.

[110] Ana C. Huamán Quispe, Heni Ben Amor, Henrik I. Christensen, and Mike Stilman. Grasping for a purpose: Using task goals for efficient manipulation planning. *CoRR*, abs/1603.04338, 2016.

[111] N. Blodow R. B. Rusu and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pages 1848–1853, 2009.

[112] BRoth Raghavan M. A general solution for the inverse kinematics of all series chains. In *Proceedings of the 8th CISM-IFTOMM Symposium on Robots and Manipulators*, 1990.

[113] Omid Rezai, Ashley Kleinhans, Eduardo Matallanas, Ben Selby, and Bryan P. Tripp. Modeling the shape hierarchy for visually guided grasping. In *Front. Comput. Neurosci.*, 2014.

[114] M. Richtsfeld and M. Zillich. Grasping unknown objects based on 21/2d range data. In *2008 IEEE International Conference on Automation Science and Engineering*, pages 691–696, Aug 2008. doi: 10.1109/COASE.2008.4626412.

[115] Mario Richtsfeld and Markus Vincze. Grasping of unknown objects from a table top. In *Workshop on Vision in Action: Efficient Strategies for Cognitive Agents in Complex Environments*, 2008.

[116] Máximo A. Roa and Raúl Suárez. Grasp quality measures: review and performance. *Autonomous Robots*, 38(1):65–88, Jan 2015. ISSN 1573-7527. doi: 10.1007/s10514-014-9402-3. URL https://doi.org/10.1007/s10514-014-9402-3.

[117] Maja Rudinac, Berk Calli, and Pieter Jonker. *Item Recognition, Learning, and Manipulation in a Warehouse Input Station*, pages 133–152. Springer London, London, 2012. ISBN 978-0-85729-968-0. doi: 10.1007/978-0-85729-968-0_10. URL https://doi.org/10.1007/978-0-85729-968-0_10.

[118] JE. Saff and A. Kuijlaars. Distributing many points on the sphere. In *Mathematical Intelligencer*, volume 19, pages 5–11, 1997.

[119] P. J. Sanz, A. P. Del Pobil, J. M. Inesta, and G. Recatala. Vision-guided grasping of unknown objects for service robots. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 4, pages 3018–3025 vol.4, May 1998. doi: 10.1109/ROBOT.1998.680889.

[120] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2): 157–173, 2008. doi: 10.1177/0278364907087172. URL https://doi.org/10.1177/0278364907087172.

[121] Ashutosh Saxena, Lawson L. S. Wong, and Andrew Y. Ng. Learning grasp strategies with partial shape information. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1491–1494. AAAI Press, 2008. ISBN 978-1-57735-368-3. URL http://dl.acm.org/citation.cfm?id=1620270.1620316.

[122] D. Schiebener, J. Schill, and T. Asfour. Discovery, segmentation and reactive grasping of unknown objects. In *2012 12th IEEE-RAS International Conference on Humanoid*

Robots (Humanoids 2012), pages 71–77, Nov 2012. doi: 10.1109/HUMANOIDS.2012.6651501.

[123] M. Schoeler, J. Papon, and F. Wörgötter. Constrained planar cuts - object partitioning for point clouds. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5207–5215, June 2015. doi: 10.1109/CVPR.2015.7299157.

[124] K.B. Shimoga. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research*, 15(3):230–266, 1996. doi: 10.1177/027836499601500302. URL `https://doi.org/10.1177/027836499601500302`.

[125] J. Sinnott and T. Howard. A hybrid approach to the recovery of deformable superquadric models from 3d data. In *Proceedings. Computer Graphics International 2001*, pages 131–138, 2001. doi: 10.1109/CGI.2001.934667.

[126] F. Solina and R. 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, Feb 1990. ISSN 0162-8828. doi: 10.1109/34.44401.

[127] S. C. Stein, M. Schoeler, J. Papon, and F. Wörgötter. Object partitioning using local convexity. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, June 2014. doi: 10.1109/CVPR.2014.46.

[128] M. Strand, Z. Xue, M. Zoellner, and R. Dillmann. Using superquadrics for the approximation of objects and its application to grasping. In *The 2010 IEEE International Conference on Information and Automation*, pages 48–53, June 2010. doi: 10.1109/ICINFA.2010.5512331.

[129] J. Subrahmonia, D. B. Cooper, and D. Keren. Practical reliable bayesian recognition of 2d and 3d objects using implicit polynomials and algebraic invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):505–519, May 1996. ISSN 0162-8828. doi: 10.1109/34.494640.

[130] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, Nov 1991. ISSN 0162-8828. doi: 10.1109/34.103273.

[131] Andreas ten Pas and Robert Platt. Using geometry to detect grasp poses in 3d point clouds. In *ISRR*, 2015.

[132] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Symmetry-seeking models and 3d object reconstruction. *International Journal of Computer Vision*, 1(3):211–221, 1988. ISSN 1573-1405. doi: 10.1007/BF00127821. URL http://dx.doi.org/10.1007/BF00127821.

[133] Sebastian Thrun and Ben Wegbreit. Shape from symmetry. In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, pages 1824–1831, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2334-X-02. doi: 10.1109/ICCV.2005.221. URL https://doi.org/10.1109/ICCV.2005.221.

[134] N. Vahrenkamp, T. Asfour, and R. Dillmann. Robot placement based on reachability inversion. In *2013 IEEE International Conference on Robotics and Automation*, pages 1970–1975, May 2013. doi: 10.1109/ICRA.2013.6630839.

[135] Nikolaus Vahrenkamp and Tamim Asfour. Representing the robot's workspace through constrained manipulability analysis. *Auton. Robots*, 38(1):17–30, January 2015. ISSN 0929-5593. doi: 10.1007/s10514-014-9394-z. URL http://dx.doi.org/10.1007/s10514-014-9394-z.

[136] K. M. Varadarajan and M. Vincze. Object part segmentation and classification in range images for grasping. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 21–27, June 2011. doi: 10.1109/ICAR.2011.6088647.

[137] G. Vezzani, U. Pattacini, and L. Natale. A grasping approach based on superquadric models. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1579–1586, May 2017. doi: 10.1109/ICRA.2017.7989187.

[138] Charles Wampler and Alexander Morgan. Solving the 6r inverse position problem using a generic-case solution methodology. *Mechanism and Machine Theory*, 26(1):91 – 106, 1991. ISSN 0094-114X. doi: https://doi.org/10.1016/0094-114X(91)90024-X. URL http://www.sciencedirect.com/science/article/pii/0094114X9190024X.

[139] P. Whaite and F. P. Ferrie. From uncertainty to visual exploration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1038–1049, Oct 1991. ISSN 0162-8828. doi: 10.1109/34.99237.

[140] William T. Vetterling William H. Press, Saul A. Teukolsky and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[141] W. Wohlkinger and M. Vincze. 3d object classification for mobile robots in home-environments using web-data. In *19th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD 2010)*, pages 247–252, June 2010. doi: 10.1109/RAAD.2010.5524578.

[142] Kenong Wu and Martin D. Levine. 3-d shape approximation using parametric geons. *Image Vision Comput.*, 15(2):143–158, February 1997. ISSN 0262-8856. doi: 10.1016/S0262-8856(96)01124-9. URL `http://dx.doi.org/10.1016/S0262-8856(96)01124-9`.

[143] K. Yamazaki, M. Tomono, T. Tsubouchi, and S. i. Yuta. A grasp planning for picking up an unknown object for a mobile manipulator. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2143–2149, May 2006. doi: 10.1109/ROBOT.2006.1642021.

[144] N. Yokoya, M. Kaneta, and K. Yamamoto. Recovery of superquadric primitives from a range image using simulated annealing. In *[1992] Proceedings. 11th IAPR International Conference on Pattern Recognition*, pages 168–172, Aug 1992. doi: 10.1109/ICPR.1992.201533.

[145] Tsuneo Yoshikawa. *Foundations of Robotics: Analysis and Control*. MIT Press, Cambridge, MA, USA, 1990. ISBN 0-262-24028-9.

[146] F. Zacharias, C. Borst, and G. Hirzinger. Capturing robot workspace structure: representing robot capabilities. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3229–3236, Oct 2007. doi: 10.1109/IROS.2007.4399105.

[147] F. Zacharias, C. Borst, M. Beetz, and G. Hirzinger. Positioning mobile manipulators to perform constrained linear trajectories. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2578–2584, Sept 2008. doi: 10.1109/IROS.2008.4650617.

[148] F. Zacharias, W. Sepp, C. Borst, and G. Hirzinger. Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories. In *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pages 55–61, Dec 2009. doi: 10.1109/ICHR.2009.5379601.

[149] Yan Zhang, Andreas Koschan, and Mongi Abidi. Superquadrics based 3d object representation of automotive parts utilizing part decomposition. In *2006 SPIE 6th Interna-*

tional Conference on Quality Control by Artificial Vision, volume 5132, pages 241–251, May 2003.

Appendix

The superellipsoid fitting and grasping library can be found online at github.com/jontromanab/sq_grasp/

The reachability mapping libreary Reuleaux can be found online at http://wiki.ros.org/reuleaux

List of Publications:

Abhijit Makhal, Federico Thomas, Alba Perez Gracia. Grasping unknown objects in clutter by superquadric representation. In Proceedings. 2nd IEEE International Conference on Robotic Computing, pages 292–299, January 2018.

Abhijit Makhal, Alex K.Goins . Reuleaux: Robot base placement by reachability analysis. In Proceedings. 2nd IEEE International Conference on Robotic Computing, pages 137–142, January 2018.

Abhijit Makhal, Alba Perez Gracia. Solvable Multi-fingered hands for exact kinematic synthesis. In Proceedings. Advances in Robot Kinematics, Pages 139-147, 2014.

Movassagh-Khaniki R, Hassanzadeh N, Makhal A, Perez-Gracia A. Design of a Multi-Palm Robotic Hand for Assembly Tasks. ASME. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 5B: 40th Mechanisms and Robotics Conference ():V05BT07A079. doi:10.1115/DETC2016-59980.