แผนการจับวัตถุที่เสถียรและมีประสิทธิภาพ โดยใช้บริเวณสัมผัสอิสระและการกักขังวัตถุ

นายธีศิษฏ์ มคะปุญโญ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2560

EFFICIENT AND ROBUST GRASP PLANNING BASED ON INDEPENDENT CONTACT REGION AND CAGING

Mr. Teesit Makapunyo

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2017

Thesis Title	EFFICIENT AND ROBUST GRASP PLANNING BASED ON INDEPENDENT CONTACT REGION AND CAGING
By	Mr. Teesit Makapunyo
Field of Study	Computer Engineering
Thesis Advisor	Assistant Professor Doctor Attawith Sudsang

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

.....Dean of the Faculty of Engineering

(Associate Professor Supot Teachavorasinskun, D.Eng.)

THESIS COMMITTEE

.....Chairman

(Assistant Professor Doctor Sukree Sinthupinyo)

.....Thesis Advisor

(Assistant Professor Doctor Attawith Sudsang)

.....Examiner

(Assistant Professor Doctor Nattee Niparnan)

.....Examiner

(Assistant Professor Doctor Nuttapong Chentanez)

.....External Examiner

(Doctor Thanathorn Phoka)

ธีศิษฏ์ มคะปุญโญ : แผนการจับวัตถุที่เสถียรและมีประสิทธิภาพ โดยใช้บริเวณสัมผัสอิสระและการกักขังวัตถุ (EFFICIENT AND ROBUST GRASP PLANNING BASED ON INDEPENDENT CONTACT REGION AND CAGING) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร. อรรถวิทย์ สุดแสง, หน้า.

งานวิจัยส่วนใหญ่คำนวณหาท่าจับที่เหมาะสำหรับการจับวัตถุใดๆ ด้วยการวัดความมั่นคงของท่าจับนั้น ซึ่งมักจะใช้คุณสมบัติทางฟิสิกส์ที่เรียกว่า คุณสมบัติแรงแบบปิด (force-closure) วิธีการหาท่าจับวัตถุแบบนี้มักจะทำงานได้ดีในทางทฤษฎี แต่มักเกิดข้อผิดพลาดเวลาใช้งานบนหุ่นยนต์จริง ซึ่งเกิดอาจเกิดจากหลายปัจจัยประกอบกัน เช่น สัญญาณรบกวนในระบบเซนเซอร์ ความไม่แม่นยำของตัวหุ่นยนต์ และความคลาดเคลื่อนระหว่างสิ่งที่เกิดขึ้นจริงและสิ่งที่คำนวณได้จากทฤษฎีทางด้านฟิสิกส์ นักวิจัยจึงนำเสนอวิธีการบรรเทาผลกระทบที่เกิดจากปัจจัยเหล่านี้ เพื่อให้ท่าจับที่คำนวณได้มีโอกาสสำเร็จมากขึ้น หนึ่งในนั้นก็คือ การกักขังวัตถุ (caging) และบริเวณสัมผัสอิสระ (independent contact regions) งานวิจัยนี้จึงศึกษาถึงความเป็นไปได้ที่จะนำเทคนิคทั้งสองมารวมกัน เพื่อหาท่าจับที่ดีขึ้นกว่าเดิม ผลการวิจัยที่ได้คือวิธีการเรียนรู้หาท่าจับวัตถุด้วยเครือข่ายประสาทเทียม (artificial neural network) ให้รู้จำท่าจับที่เหมาะสำหรับการจับวัตถุใดๆ โดยใช้เทคนิคที่เราสนใจทั้งสองและวิธีการแบบศึกษาสำนึก

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2560

5471411721 : MAJOR COMPUTER ENGINEERING
KEYWORDS: GRASPING / GRASP PLANNING / ARTIFICIAL NEURAL NETWORK / PARALLEL COMPUTING / GRASP QUALITY MEASURE / GRASP SYNTHESIS / HEURISTIC APPROACH

TEESIT MAKAPUNYO: EFFICIENT AND ROBUST GRASP PLANNING BASED ON INDEPENDENT CONTACT REGION AND CAGING. ADVISOR: ASST. PROF. DR. ATTAWITH SUDSANG, pp.

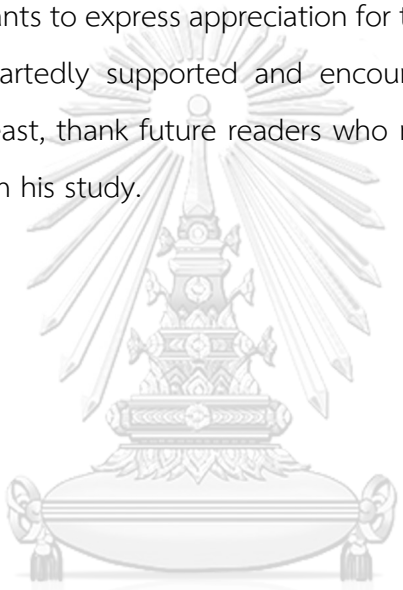
A conventional way to find a proper grasp to grab and hold any object is to measure its stability which usually is based on physical constraint called force-closure. This execution works well from the theoretical point of view but often fails on an actual robot due to many reasons such as intrinsic errors in robot's system and a disparity between real and simulated physics. Several research works introduced methods to alleviate those issues and increase the success rate of grasping for a real robot. Caging and Independent Contact Region are ones of them. In this work, we investigate a method to find a grasp that is more stable and robust by combining those two techniques which result in a learning-based approach that utilizes an artificial neural network to find a proper grasp based on those techniques and some heuristic methods.

Department: Computer Engineering Student's Signature .....
Field of Study: Computer Engineering Advisor's Signature .....
Academic Year: 2017

ACKNOWLEDGEMENTS

The author would like to thank undergraduate students for useful information in the grasping survey, his advisors, Dr. Attawith Sudsang and Dr. Nattee Niparnan, for their guidance and technical knowledge and friends and colleagues in the department of computer engineering and Intelligent System Engineering Laboratory 2 at Chulalongkorn University for inspiration and helpful comments.

Finally, he wants to express appreciation for the people closest to him: his family, who wholeheartedly supported and encouraged him to complete this thesis. Last but not least, thank future readers who might be inspired or discover useful knowledge from his study.



CONTENTS

Page

Page

Page

Chapter 1 Introduction

Robots play an indispensable role in automated manufacturing processes where they carry out a set of predefined tasks in the structured environment of a factory such as material transport, part assembly, and packaging. They proved to be reliable and more cost-effective than human labor. Nowadays, robot application applies to many domains and consumer products such as service and entertainment. The interaction between the robot and surrounding environment plays a vital role in those applications. One of the most common interaction is object manipulation where a robot carries out a set of actions to accomplish a specific task on a specified object, e.g., hammering a nail, pouring water into a glass and opening a door. Those tasks look simple yet very challenging for a robot in an unstructured environment. Despite decades of effort, object manipulation in an unstructured environment is still, in general, an open problem.

One of the most crucial actions in many manipulation tasks is grasping, i.e., to immobilize and hold an object. This action is simple for human yet very challenging for a robot because, with present technology, it is difficult to accurately predict or simulate the physical interaction between a robot's end effector (i.e., a gripper or an anthropomorphic hand) and an object. So, in robotics theories, the grasping problem is usually simplified to make its computation feasible. One of the most popular grasping models is called dexterous grasping which limits the physical interaction to a set of contact points where a robot's fingertip touches the object. A general statement of the dexterous grasping problem is as follow: given the position and shape of an object to be grasped, find a grasp, which is a set of contact points or a robot's configuration, that satisfies specific properties which is suitable for a given task.

Early grasping research works in the late 80s and early 90s focused mainly on force and mobility analysis of a grasp and an object such as the concept of force-closure and form-closure. Those conditions have served as the foundation for many conventional grasp planning methods often called analytical approach. They usually assumed a known, perfect object model and robot control. This assumption works

well in ideal scenarios such as simulation, but it is hardly satisfied in practice due to a limitation of computational power and sensor's hardware. Even the smallest error in sensor data may cause a catastrophic failure in grasping process. One of the most common errors in grasping is a premature contact where one of the robot's fingers prematurely touches and moves an object from its original pose. The change in object position makes the actual contact points deviated from the planned ones and may cause a grasp to become unstable and ultimately failed.

There are several approaches to mitigate positional errors and improve the success rate of grasping. For example, Nguyen introduced the concept of Independent Contact Regions (ICRs) which is a set of contact regions on an object surface that satisfies the force-closure property. When a robot makes a grasp within those contact regions, the grasp will tolerate positioning errors as long as it remains within the contact regions. Thus, its force-closure property is still valid. Another interesting approach is a method based on a kinematic constraint called caging. A robot forms a cage when an object cannot move arbitrarily far from the robot's hand. There are several advantages If a robot achieves a cage during grasping. Firstly, a cage is a good waypoint to form a stable grasp in some situations. For two-fingered grasp, either squeezing or stretching fingers while maintaining a cage will result in a form-closure grasp, i.e., an object is immovable within grasp. Secondly, it is easier to repositioning contact points or re-grasp an object since an object cannot move arbitrarily far from robot's hand. Caging is also a smart way to move an object around since it requires less control precision and guarantees that an object remains within a limited region (cage).

During our research, we found that conventional methods based on analytical approach, such as caging and force-closure property, are inflexible and failed to recognize many grasps that are good from the human's perspective. Since the emergence of machine learning in the early 2000s, many novel grasp planning methods tried to learn the quality of a grasp by training a learning model from examples and demonstrations on a real robot. Those methods are called data-driven grasp planning methods and often categorized as an empirical approach in robotic grasping. The data-driven methods usually classified a grasp into two classes: good grasps and bad grasps. This approach is very good at adapting to various scenarios, but it is also hard to

understand why it succeed and how to implement the method correctly. Both conventional and novel approaches have their distinct advantages and drawbacks. In the light of this, our new grasp planning method aims at implementing a hybrid method between both approaches to eliminate their drawbacks while maintaining their advantages.

The culmination of our research in the grasp planning method based on caging, ICRs, and data-driven grasp planning methods consists of 1) a heuristic method to measure an ability of a grasp to cage an object called *partial cage* 2) a simple iterative algorithm to find ICRs for two-fingered grasp. 3) a set of grasping features to learn the grasp quality via a simple neural network. 4) qualitative analysis on learning the grasp quality from the grasping feature and the grasps labeled by the human survey.

A partial cage explores a way to assess caging ability of a grasp that cannot cage an object entirely. The crux of the partial cage is to measure how hard for an object to move arbitrarily far from a grasp. In this work, we deploy a physics simulator and a motion planning method to move an object randomly until it is far from a grasp and analyze object's trajectories to determine an ability to cage of a grasp.

The iterative ICRs algorithm provides a parallelizable and straightforward way to find the largest ICRs on an object. We proved that a pair of triangles form ICRs if all pairs of their extreme points satisfy force-closure properties and utilize this fact to iteratively expand ICRs originated from every pair of triangles on an object represented by a triangular mesh. In the experiment, the algorithm efficiently found the largest ICRs on objects with more than 20,000 triangles within a few seconds using parallel computational power on GPU.

Our data-driven method utilizes a simple neural network to learn a good grasp from features derived from both analytical and heuristic methods. We demonstrate it in the simplest scenario which is a two-fingered grasp for a 2D object. The grasp samples are labeled by 1) volunteers with no prior knowledge of robotic grasping 2) three individuals who had background knowledge in grasping theory 3) heuristic methods that determine the label of a grasp on the predefined objects. In the experiment, we review the grasps obtained from the survey, investigate the suitable structure for a neural network to learn the grasp quality, and verify our model against

the collected data. This investigation provides remarkable clues about the fundamental problems in the robotic grasping and finally leads to the final chapter, our opinions about the present and future state of the robotic grasping.

After the introduction, this thesis is structured as follows. Chapter 2 summarizes fundamental knowledge and briefly reviews related works in robotic grasping. Chapter 3 presents our contributions toward grasp planning methods based on caging and independent contact regions. We introduced a heuristic method to measure an ability of a grasp to cage an object called *partial cage* and a simple and a novel parallel algorithm to find independent contact regions for a two-fingered grasp on a 3D object. 0 discusses our hybrid method that deploys machine learning to learn a good grasp from its properties derived from both analytical and heuristic methods. Finally, Chapter 5 expresses our opinions for the current state of robotic grasping and its future.

Chapter 2 Definitions and Literature review

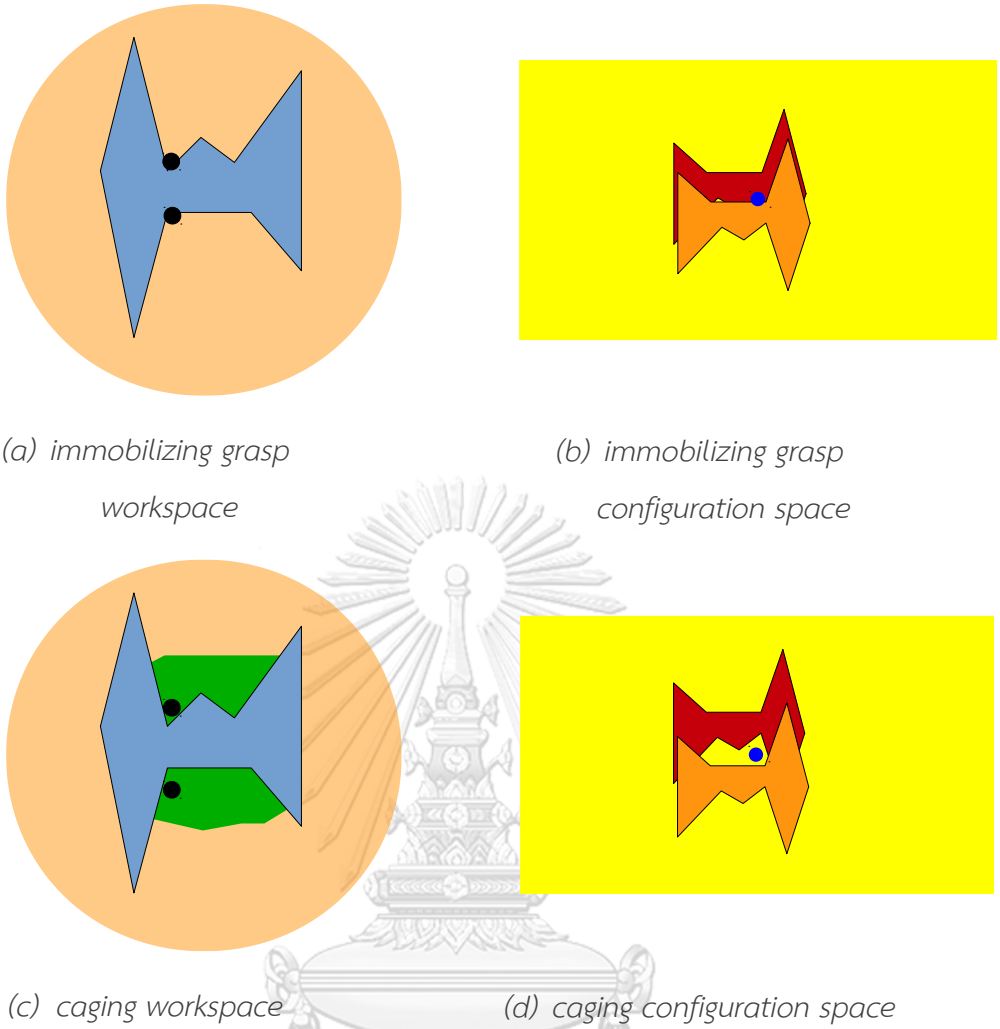
In this chapter, we discuss terminologies that often used in robotic grasping including the related research topics to provide fundamental knowledge and background of our works. Additionally, we also briefly discuss classical works and similar studies in this field.

2.1 Space Definition in Grasping

Analytical grasping involves calculation in various fields such as kinematic, geometric and physics problems. Many studies utilize *n*-dimensional spaces to explain and visualize a grasp in a specific field and use it to solve a grasping problem. In this section, we describe the definition of spaces that often used in caging and ICRs. The most common one is a *working space* which is the physical coordinate space that describes relative position and orientation of a robot and an object to be grasped. A *wrench space* visualizes forces and torques that apply to an object from robot's grasp. Lastly, a *configuration space* represents kinematic states of an object relative to robot's grasp or vice versa.

2.1.1 Workspace

Workspace usually refers to a 2D or 3D real coordinate space that robot can operate in the scene. For 2D workspace, a polygon or a curved function usually represents an object to be grasped. While in 3D, it can be in many forms depending on several types of input such as points cloud, polyhedral, triangle mesh, and composition of primitive shapes. Manipulator often represented as a set of fingers, is a part of the robot that interacts and manipulates the target. For simplicity, it usually represents by a set of points in workspace because, in theory, we only interest interaction between object and manipulator that occurs at the contact point where finger's tip touches an object. This way, grasping computation can be invariant to manipulator mechanics.



(a) immobilizing grasp workspace

(b) immobilizing grasp configuration space

(c) caging workspace

(d) caging configuration space

Figure 1 Example of workspace and configuration space.

In workspace (figure (a) and (c)), a blue polygon represents an object, and black dots represent fingertips that act as obstacles and gain control over the target. A green region in figure (c) illustrates caging set, an independent region of each finger placement that can cage the object. In configuration space (figure (b) and (d)), red and orange polygons are the area that the object cannot translate into without colliding one or more fingers and the blue dot is the origin of configuration space that represents the current position of the object in the workspace. It also exhibits a significant difference between immobilizing grasp and caging, an area of isolated free space.

2.1.2 Wrench space

Wrench space represents force and torque that exert on an object in 6D space (3D for 2D workspace). It can efficiently resolve resultant force and moment that act on the rigid body and evaluate its linear and angular velocity. A wrench that represents forces and torque applied to contact points is called *primitive wrench*. The arrangement of the primitive wrenches, especially their convex hull, determines the capability of a grasp to resist against any forces acting to an object. The convex hull of primitive wrenches is often called *grasp wrench space*. In some articles, the wrench subspace that formulates a specific task objective is called *task wrench space*. It is usually an ellipsoid or a convex hull of wrenches that represent minimum force and torque needed to be exerted to complete the objective. The analysis of both grasp wrench space and task wrench space can determine the likelihood of a given grasp completing a given task.

2.1.3 Configuration space

Configuration space describes all possible motion states of the finger position in workspace having an object as an obstacle or vice versa. Figure 1 shows an example mapping between workspace and configuration space. A subspace that an object does not overlap fingers is called free configuration space, in short, *free space*. The application of configuration space mostly notable in kinematics such as finding a collision-free path that connects between two configurations, determine a degree of freedom and formulate a cage that bound an object within a limited space. When all fingers' position is fixed relative to each other, they are called *preshape* or *finger formation*. Some works represented finger formation as a shape function. The parameter of shape function controls distribution and internal structure of finger formation. When fingers surround an object and completely isolate object's free space as depicted in Figure 1(d), a cage is formed. In other words, fingers cage an object when there is no path connecting an initial pose to an arbitrarily far configuration without colliding with an obstacle. Such path, if existed, called an *escaped path*. It is sufficient to assume that an arbitrarily far configuration is a state that the convex hull of an

object did not overlap with the convex hull of a finger formation. Sometimes, it is sufficient to evaluate grasp pose and caging set only on the object boundary. *Contact space* can express a finger position on a planar object's boundary with a single parameter: object perimeter. Some research works computed the optimal grasp pose and caging set in this space because their algorithms only compute on the object boundary, not free space.

2.2 Analytical Grasp Planning

In robotics, dexterous grasping refers to a grasp that only utilizes fingertips to grasp or manipulate an object. Limiting the physical interaction of a grasp to fingertips simplifies the grasping problem and make it easier to simulate the physical interaction between a grasp and an object. A grasp is usually a set of contact points where robot's fingertip touches and exerts force and torque to manipulate an object. Roboticists often represent a contact point by a pair of fingertip's position on an object and its contact normal, a directional vector that perpendicular to object surface on that position. So, grasping problem is simplified to the determination of a suitable set of contact points or approach direction to grasping a given object. A method that solves the grasping problem, i.e., finding a set of contact points to grasp an object, is often called *grasp planning*. There are generally two types of grasp planning method: analytical and empirical. In this section, we briefly discuss the first one, analytical grasp planning methods while the upcoming section explores empirical grasp planning methods.

The analytical grasp planning derives from the mechanical model and the physical interaction between an object and robot's hand. It usually involves calculation of *grasp quality* from a set of sampling grasps and returns a grasp with the highest quality to perform a grasping task. Grasp quality is an index that indicates goodness of a grasp. There are several grasp qualities associated with the analysis of physics and kinematics model of a grasp. In this section, we will discuss exceptional grasp qualities from analytical approach such as force-closure, ICRs and caging.

2.2.1 Stable Grasp, Force-Closure, and Form-Closure Property

Grasp stability is an ability to keep the object in an equilibrium state. It plays a crucial role in many conventional grasp planning methods to determine the quality of a grasp. They analyzed grasp stability from magnitude and direction of the force that a grasp can exert on an object through a set of contact points to resist the external force acting on an object, e.g., gravitational force. In general cases, we assume that external forces are not known beforehand, so they are arbitrary in both direction and magnitude. As such, an ability to resist forces and torques in any directions became a necessary condition for a stable grasp, and this ability is often called *force-closure* property.

Another property that also defines grasp stability is *form-closure* which stemmed from object's kinematics. A grasp is form-closure when robot's fingertips (contact points) placed along object boundary in a specific formation that will prevent any object movement. In other words, an object is immobilized and cannot move without colliding one or more fingertips. This stability criterion uses fingertips as geometrical constraints that obstruct all object motion as illustrated in Figure 1(a) and (b).

Form closure is closely related to force closure property. Rimon and Burdick [1] showed that form closure grasp is equivalent to force closure grasp with frictionless contact. The difference between form closure and force closure grasp had been pointed out by Bicchi [2]. He stated that the difference between the two is the perspective when analyzing the problem.

A conventional method verifies the force-closure property of a grasp in wrench space. If the origin of wrench space lies strictly within grasp wrench space, a convex hull formed by wrenches that can be exerted by a grasp through contact points, such grasp satisfies force-closure property [3]. Liu [4] verified the force-closure condition by solving the linear optimization problem while Zhu *et al.* [5] tested the condition using collision detection in wrench space.

There are several grasp qualities associated with the force-closure property. For example, Kirkpatrick *et al. [6]* and Ferrari and Canny [7] introduced a grasp quality

based on the shortest distance between the origin of wrench space and the boundary of grasp wrench space. This quality metric is often called ϵ-metric. It had been widely used in many grasp planning and simulator to evaluate grasp stability and realize the optimal grasp for grasping task. Recently, Zheng [8] introduced an efficient way to compute ϵ-metric using an iterative method. A grasp with two soft fingers simplifies the force-closure condition to the intersection between double-sided friction cones and contact points [9, 10]. Our works utilized this method to verify the force-closure property of two-finger grasps. The implementation details can be found in Appendix A. Xiangyang *et al.* [11, 12] proposed a quality metric called *Q-distance* as the highest scale factor of a task wrench space that still contains within in a grasp wrench space. Shi and Koonjul [13] developed a real-time grasp planning method for the multi-fingered robotic hands. They simplified the high-dimensional grasping problem by decomposing it into the lower ones using two proposed strategies called the intersected volume and the finger curling planes. They verified the method on three different robotic hands with the bin-picking and kitting tasks.

There are grasp qualities that based on the characteristic of grasp's geometry. In planar grasp, it is desirable to spread contact points uniformly over object surface to improve grasp stability. Byong-Ho *et al.* [14] proposed a quality index to quantify the uniformity of contact points distribution from the internal angles of a polygon formed by contact points. The optimal polygon is the one corresponding to a regular polygon, i.e., its internal angles are all equaled. To minimize the effect of inertial and gravitational force, some quality indices [15-17] use the distance between the center of mass of an object and the center of a polygon formed by contact points. Several works [18-21] provided comprehensive reviews on grasp quality and analytical grasping in details.

2.2.2 Independent Contact Regions

Nguyen [22] introduced a concept of an Independent Contact Regions (ICRs) for planar grasps which is a set of contact regions such that if a robot places each fingertip within its respective regions, a grasp is always force-closure. This approach

improves robustness against errors in finger placement; thus, it is easier to grasp an object.

Pollard [23] proposed a similar method which generates families of grasps from a single grasp. It preserved grasp quality of a sampled grasp and represented the families as a set of independent contact regions.

Roa *et al.* [24] sampled a force-closure grasp and optimized its quality by swapping the current set of contact points with their respective neighborhood with higher grasp quality. Later, they used a similar approach to synthesize independent contact regions in [25-27]. Given an initial set of contact points, the independent contact regions were expanded by repeatedly adding the neighbor contact points which could substitute the original ones while maintaining grasp quality within a certain threshold. Krug *et al.* [28] and Dang-Vu *et al.* [29] improved the efficiency of an algorithm in the previous works. They relaxed the constraints of the inclusion of neighborhood which increased the size of ICRs. In [30], they focused on the user-defined priority of contact points for ICRs' inclusion. This feature allowed users to change the ICRs' shape to meet the task objective. Phoka *et al.* [31, 32] found the optimal independent contact regions for two-finger planar grasp on a polygonal object. They formulated a grasp in a 2D contact space and constructed the independent contact regions which span across multiple consecutive edges of the polygon. Our proposed method in Section 3.2 is closely related to Roa's works [25-27], but we expand all contact regions simultaneously instead of synthesizing only a single set of independent contact regions.

Rosales *et al.* [33] integrated ICRs concept into Inverse Kinematics (IK) algorithm for motion planning of robotic hand. They formulated ICRs problem as constraints in IK and maximize region-to-region contact surface between an object and a hand. Jeong and Cheong [34] utilized the concept of the independent contact regions in an in-hand manipulation task. They maintain grasp stability while a robot changing a grasp pose by moving its fingertips strictly within independent contact regions. Fontanals *et al.* [35] integrated independent contact regions into a motion planning called Bidirectional Rapidly-Exploring Random Tree (Bi-RRT). Their algorithm concurrently synthesized ICRs and finds motion path toward those ICRs, thus faster than conventional methods which

solve those problems sequentially. Alvarado *et al.* [36] proposed an algorithm to synthesize independent contact regions for articulated objects such as tongs and scissors. They introduced a new wrench space called Generalized Wrench Space specifically for those articulated objects.

2.2.3 Cage

Kuperberg [37] introduced the definition of caging. He stated that a polygon, P, in the plane is captured by a set of n points, C if P cannot be moved arbitrarily far from its original position without overlapping at least one point of C. Simply put, caging has an ability to limit object movement within specific space around points C, in this case, a set of robot's fingers or a group of mobile robots. Caging does not need to make direct contact with an object. Thus, it is easier to achieve than grasping which requires precise control for finger's positioning on the object surface.

There are several ways to construct a cage for a given object. A robot can cage an object by evenly place a sufficient number of fingers around an object in a circle formation such that an object cannot pass a gap between any two adjacent fingers. Sudsang and Vongmasa [38, 39] addressed the maximal width of the gap such that an object cannot pass a circle formation of the fingers.

Pipattanasomporn *et al.* introduced several algorithms to find two-fingered cages for 2D and 3D objects [40-42]. They proved that a cage is satisfied if we maintain the separating distance under (squeezing) or over (stretching) certain threshold. To find the threshold and formulate a cage, they expressed caging as a graph search problem on the object surface and free configuration space where the cost function on the graph's nodes and edges is separation distance between fingers. They utilized an algorithm similar to A* search to find cages on the proposed graph. Allen *et al.* [43] introduced a similar concept for two-fingered cage on a planar object. They proposed a graph on contact space called *caging graph* formed by critical configurations (e.g., an immobilizing grasp and a puncture grasp) on the contact space and derived cages around those configurations from the graph.

Several algorithms focus on a cage with three or more fingers and limited degrees of freedom. Erickson *et al.* [44] analyzed the configuration space to find a set of possible position for the last finger in a three-fingered cage when the other two fingers are stationary. This specific case of the three-fingered cage was further studied in [45, 46]. Pipattanasomporn *et al.* generalized their algorithm in [41] to arbitrary dimension with an arbitrary number of fingers [47, 48]. They introduced conditions for finger arrangement called *dispersion control* and derived the cost function of graph search problem from those conditions.

In some scenarios, partially limiting the object movement might be more desirable than a complete cage since the caging algorithms are computationally expensive. Suarod *et al.* [49] proposed a heuristic method to determine the number of fingers and their position that loosely cage a planar object. Their algorithm initially finds a set of fingers that cages an object with a fixed orientation. Then, it iteratively improves the initial formation by adding or shifting the fingers if it found an escape path when rotating an object by a small angle. Makapunyo *et al.* [50, 51] introduced the concept of the *partial cage*, an empirical method to measure how much effort an object would take to escape finger formation. We will explore and discuss the definition of the partial cage and its measurement methods in Section 3.1. Later, Makita and Nagata [52] applied the concept of the partial cage to their two-fingered manipulator inspired from a fish trap, i.e., its limbs form a partial cage leaving a small gap that an object can enter and leave it. They evaluated the effect of the finger arrangements on the partial cage in a 2D simulation. Mahler *et al.* [53] introduced the energy-bounded caging analysis on a planar space (EBCA-2-D). They analyzed the influence of a force field such as gravity force when a manipulator partially caged an object. They estimated the minimal energy required for an object to escape an energy-bounded cage via the collision analysis and sampling-based approach.

A robot can utilize the concept of caging in various manipulation tasks. For examples, a robot can move an object using a cage since the object would move along the manipulator that cages it [54-59]. The caging concept also utilized in manipulating a constrained object such as doorknob and drawer's handle [60, 61]. The fundamental idea is that a manipulator does not have to be rigidly attached to an object throughout

the entire process. A manipulator has some degree of freedom as long as it cages an object and the task constraint is satisfied. This increased range of possible motion provides more feasible solutions and improve the success rate of those tasks. Vahedi and Stappen [62] provided comprehensive analysis and algorithms of two and three fingers caging for a polygonal object. Makita and Wan [63] summarized the studies on caging and its application in the robotic grasping.

2.2.4 Caging grasp

Several studies focused on grasp planning methods that exploit the relationship between caging and grasping. They introduced several ways to integrate those two concepts and introduced a novel approach called *caging grasp*. This approach is closely related to the primary objective of this thesis, the integration of caging and independent contact regions.

Gopalakrishnan and Goldberg [64] proposed an algorithm to grasp a deformable object using two fingertips. They introduced a *deform closure* grasp which immobilizes a deformed object when the gripping force transforms the area around the contact points into concave sections making them a form-closure grasp which requires positive works to release an object from the grasp. Vahedi and Stappen [45] studied the relationship between a cage and an immobilizing grasp for a three-finger hand and a convex polygonal object. They also introduced an algorithm which determines if a given finger arrangement is a cage or not.

Rodriguez *et al.* [65] introduced a concept called the *pregrasping cage*. A pregrasping cage has a straightforward strategy that leads to an immobilizing grasp. The most trivial case of a pregrasping cage is a manipulator with two fingers. Its caging property remains valid as long as *separation distance*, a distance between two fingers, is kept below (squeezing) or above (stretching) a certain threshold. Under this condition, all two fingers cages are either squeezing cage, stretching cage or both [66]. This characteristic of pregrasping cage provides an easy method to grasp an object: depending on the type of cage, either close or open the fingers until forming an immobilizing grasp. The squeezing and stretching conditions guarantee that an object

cannot escape while manipulator is trying to grasp it. Thus, accurate positioning or closed-loop control is not necessary.

Maeda *et al.* [67] introduced caging-based grasping which utilizes both caging and grasping properties to manipulate an object. Their robot's fingers had two layers. The outer layer was soft while the inner layer was rigid. They caged an object using rigid layer and grasped an object using the soft layer. Wan *et al.* [68, 69] synthesized a squeezing cage from configuration space for the planar object. They showed that a cage tolerates uncertainty from object shape and it is suitable for power grasp tasks in the simulation.

2.3 Empirical Grasp Planning

The grasping in a real robot prefers obtaining a sufficiently good grasp as fast as possible instead of slowly acquiring the best possible grasp. Borst *et al.* [70] showed that a simple grasp planning that randomly generated grasp candidates and filtered them using simple heuristic is sufficient to find relatively good grasps. A grasp planning method that relies on grasp sampling and ranking them based on predefined grasping experience is often called empirical approach or data-driven grasp planning method. Since the early 2000s, the empirical approach had been a highly active area of research in robotic grasping. They extract salient features from a set of sampling grasps and use statistical methods or learning models to recognize and predict the characteristics of good grasps. The sampling data come from several sources such as the human demonstration, simulation, and real robot grasps. Grasp planning methods usually derive the grasp's features from grasp analysis, speculation, and vision-based descriptor. We also proposed another data-driven method that predicts grasp quality based on grasp analysis, speculation, and the data collected from the human's survey in the 0. Unlike previous works in this field, we combined several grasp quality metrics including a heuristic of caging and independent contact regions with the neural networks and made the qualitative analysis to verify our method in predicting the grasp quality from the human perspective.

Several works segmented an unknown object into primitive shapes [71, 72] or superquadrics [73, 74] and synthesized grasp candidates from them. Recently, several works applied machine learning and artificial intelligence to solve the grasping problem. Miller *et al.* [72] provided several strategies to grasp an object which decomposed into primitive shapes. They substituted a grasp with a predefined hand posture called *preshape* and an approach direction instead of a set of contact points. Saxena *et al.* [75, 76] identified a good grasp using supervised learning. Their method learned potential features from synthetic data and obtained the grasp position from two or more images. Huebner *et al. [71]* focused on grasping the small part of an object. They decomposed an object model into primitive shapes such as a box, sphere and, cylinder, then synthesized a good grasp on those shapes independently.

Detry *et al.* [77] learned a grasp through experience by repeatedly grasping an object in various direction and update the probabilistic model from the results. El-Khoury *et al.* [74] represented each part of an object as a superquadratic and then learned which parts of an object that robot can effortlessly grasp. They assumed that objects in everyday tasks have a handle which is a natural graspable part of an object. By learning how to identify object's handles from synthetic data, their algorithm generated the best force-closure grasp using the method presented in [78]. Bohg *et al.* [79] reconstructed an object model from a single depth image. They estimated the occluded part of an unknown object while assuming that objects possessed one or more planar symmetry. Then, they utilized a conventional grasp planning to obtain the final grasp pose. Klingbeil *et al.* [80] analyzed raw depth images to identify the graspable part of an object and controlled a manipulator to pick up an object and read a barcode attached to it. They evaluated a firm grasp by fitting a cross-section of a depth image in the interior of a parallel gripper and found series of similar cross-section to maximize the contact surface between an object and manipulator. Kootstra *et al.* [81] constructed a hierarchical structure of contour and surface extracted from stereo camera to represent an unknown object and proposed several grasp synthesis methods for this object representation. Fischinger *et al.* [82] proposed a Haar-like feature representation called Height Accumulated Feature. They extracted this feature from the height of objects in a scene. Then, they used Support Vector Machine to

learn good grasps from the pattern of those features and successfully grasped an object from a pile of items.

Dang and Allen [83] proposed a semantic grasp planning method which satisfies both stable property and a task objective. They used shape context descriptor and a stochastic grasp planner [84] to match a good grasp similar to a set of predefined candidates which satisfy task constraints, e.g., when pouring water from a mug, the hand must not block the open area of a mug. Sergey *et al.* [85] proposed a grasp planning method using hand-eye coordination to learn how to grasp an object from a monocular image. They trained a big convolutional neural network to predict the probability of a successful grasp from images during the grasping process. Guo *et al.* [86] proposed a hybrid neural network combining information from both visual and tactile sensing. It increased the accuracy of the grasp detection which suggested that visual sensing and tactile sensing complement each other, and both are important for the robotic grasping. Borg *et al.* [87] provided a comprehensive review on data-driven grasping and its related topics.

Mahler *et al.* developed a cloud-based system called Dex-Net 1.0 [88] to learn robust grasps from a massive database of 3D object models. Their system integrated several learning-based algorithms to recognize the object similarity [89] and predict robust grasps [90, 91] using the probability of force-closure [92] as a grasp quality metric. They tested the system's scalability and parallelization on the Google Cloud Platform and claimed that it reduced runtime by three orders of magnitude. Later, they introduced the Dex-Net version 2.0 [93] and version 3.0 [94]. In the second version, they introduced a new learning model called Grasp Quality Convolutional Neural Network (GQ-CNN). They integrated and tested the system in a real robot to grasp the objects on a table using the data from an RGB-D sensor. The third version applied GQ-CNN on the vacuum-based end effector (e.g., suction cup) and introduced a new physical contact model to analyze a single-point suction grasp. They evaluated the system on a real robot equipped with a pneumatic suction gripper. Tian *et al.* [95] introduced a cloud-based robotics system called Berkeley Robotics and Automation as a Service (BRASS). It allowed a robot to communicate with the remote grasp planning system, the Dex-Net 1.0 [88], and acquire a set of suitable grasp candidates

for an object. They verified the service on a real robot and tested the effect of network latency during the operation.

Chapter 3 Novel Caging and Independent Contact Region methods

Our first (and several) attempt(s) at combining caging and ICRs is to improvise existing methods and apply simple logical process to query the best grasp. For examples, select ***N*** best grasps from method ***A***, then use method ***B*** to find the best one among those ***N*** grasps. Another way is to design a simple mathematical formula to evaluate grasp's score based on the outcomes of method ***A*** and ***B***. Those naïve approaches sounded plausible at first, but we stuck at the very last step: how to verify and compare our approach against existing ones? Since we pointed out that force-closure property alone is insufficient to determine a good grasp, what measurement method should we choose to verify our hypothesis? Due to those philosophical questions, we realized that a gap between grasping theories and practical grasping is more significant than we predicted and there is no ground-truth in measurement for the goodness of all grasps. In the end, we changed our approach toward machine learning discussed in the next chapter. Before that, we would like to share the experiences on development and improvising novel methods to find cages and independent contact regions namely *Partial Cage* and *Iterative Independent Contact Region*.

3.1 Partial cage

The concept of caging which captures an object within a limited region around fingers is an additional feat that makes a grasp more robust. It is especially true for a two-fingered grasp since a cage can trivially change into a form-closure grasp by either squeezing or stretching the fingers. We can establish a simple measurement for a grasp based on its ability to cage an object, for example, we prefer a grasp that associates with a cage than a grasp that has no cage. To compare two grasps that both associates with a cage, we can evaluate the characteristic of their cages such as their sizes and shape. However, how to compare grasps with no cages? There is no formal way to evaluate such cases before. Thus, we introduce a novel concept called *partial cage* and a heuristic method to evaluate its ability to cage an object. A partial cage is a

finger formation which does not cage an object entirely but can obstruct its mobility in some ways. The idea of measuring partial cage's quality is straightforward; how hard is it for an object to escape arbitrarily far from finger formation?

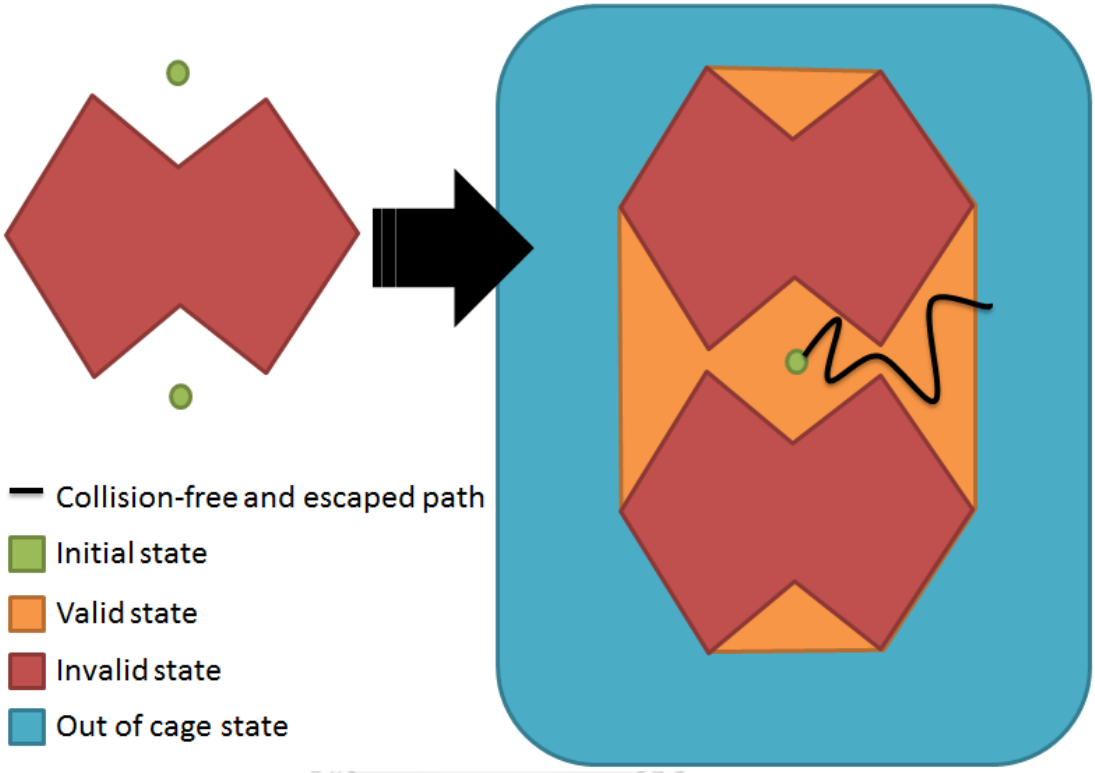


Figure 2 An example of a partial cage in the workspace (Left) and simplified configuration space (Right) that only allow translation.

3.1.1 Partial cage definition

Let formulate our partial cage in 2D space (see Figure 2). An object to be (partially) caged, O, is a red polygon defined by a sequence of points in the workspace. The initial fingers position, F_0, are defined by a set of green points and have a fixed formation constraint, i.e., every pair of fingers in a set always preserves a fixed distance. Each configuration state, $s = (x_s, y_s, \theta_s) \in SE(2)$, in the fingers' configuration space C_F represents a pose of fingers relative to their initial position at the origin, $s_0 = (0,0,0)$, which is a green point in configuration space. In another word, the

corresponding fingers position at $\boldsymbol{S}$ in configuration space, $\boldsymbol{F_s}$, can be calculated from the initial fingers position $\boldsymbol{F_0}$:

$$F_s = translate\left(rotate\left(F_{s_0}, \theta_s\right), x_s, y_s\right)$$

There are several essential sets of $\boldsymbol{S}$ in our partial cage problem. The red region in Figure 2 is a set of configuration states where one or more fingers intersect with an object. Those states are *invalid* since fingers cannot physically overlap with an object. The configuration states in other regions (blue and orange) did not have such intersection and fingers can move freely within or between those regions. The orange region is a set of *valid* states where the convex hull of an object intersects with the convex hull of fingers. We consider states in the orange region as *partial cages* and determine their ability to cage an object by an *escaped path*, a path that connects them to an out-of-cage state in the blue region. We define a path in configuration space as a series of valid states which have a direct, collision-free path connect between adjacent states. An out-of-cage state is a state where the convex hull of an object did not intersect with convex hull of fingers. In this context, an object successfully escapes (moved arbitrarily far) from fingers when it reached an out-of-cage state.

3.1.2 Partial cage quality measurement

Next, we introduce the idea of a heuristic method that predicts the quality of partial cage, i.e., how hard an object to escape from fingers? Let us consider a straightforward case of a cage where two fingers are forming a squeezing cage on a planar object. Suppose further that a cage is critical, i.e., if fingers move slightly away from each other, an object will be able to escape from those fingers. The further the fingers move away, the easier an object can escape (see Figure 3 (a)). In this scenario, it is easy to determine an ability to cage an object based on the distance between fingers, but the distance alone cannot justify other cases (see Figure 3 (b) and (c)). Ideally, we would like to have a quality metric that can distinguish partial cages based on the distance to the nearest critical cages. However, measuring the exact distance in

configuration space and finding the nearest critical cages are not trivial tasks, so our measurement relies on the probability that the object escapes from fingers instead.

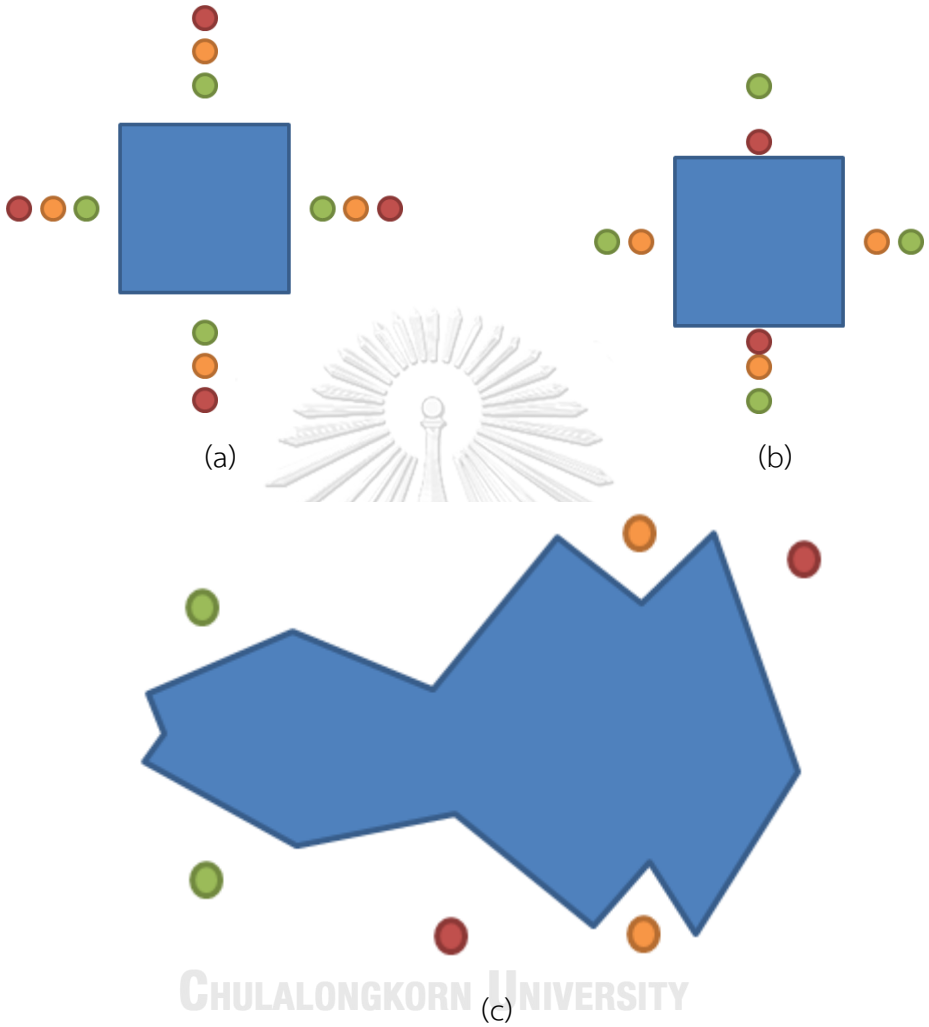


Figure 3 The quality of partial cage may depend on many properties: (a) for fixed finger formations different only in the distance between fingers, the shorter one (green pair) should have better quality than longer ones (orange and red pairs). (b) A higher number of fingers usually means better quality. (c) The assumption of distance between fingers is invalid when comparing between arbitrary formations. The green pair should have the worst quality since it is very close to out of cage state.

In real-world scenarios, the probability of object escaping depends on the characteristic of a task, for example, if a robot wants to lift an object, a cage that excels at preventing object escaped due to gravity force should score better than a cage that emphasizes on preventing object moving against gravity. However, if there is no prior knowledge of a task, we are left to assume that task space is uniform. Of course, it would be more practical if the quality can be adjusted to align with task objective. Our first work [50] demonstrates a very naïve method to compare partial cage quality based on the length of an escaped path and runtime used for finding an escaped path in simulator under several conditions.

In [51], we assumed that task space is uniform, so we moved an object around randomly until it freed from fingers and formed an escaped path starting from an initial position to an out-of-cage state. Our method sampled a set of escaped paths and made some quantitative analyses from those paths that resulted in a quantitative value that represent the quality of partial cage. Our framework consisted of two steps. The first step was sampling escaped paths of the partial cage using probabilistic-based path planning such as probabilistic roadmap (PRM) [96], single-query bi-directional probabilistic roadmap (SBL) [97] and rapidly-exploring random tree (RRT) [98]. A path planning takes geometrical information of an object and position of fingers as input and finds escaped paths by randomly sample the valid configuration states and connect them until forming an escaped path. Due to probabilistic nature of these planners, they will eventually find escaped paths if they exist and there are sufficient time and resource. The second step was the estimation of partial cage quality from escape paths in the previous step. Our method examined escaped paths under several scoring criterions such as the number of escape paths and some properties of the paths, e.g., length and curvature. Then, those scores are aggregated to represent the quality of partial cage.

3.2 Iterative ICRs algorithm

Given a force-closure grasp on the object surface, there always exists independent contact regions (ICRs) around associated contact points. For example, a

two-fingered grasp has two ICRs, i.e., one ICR for each finger. A grasp will always satisfy force-closure property if its contact points remain within their associated ICRs. In general cases, we prefer a grasp with larger ICRs since it can tolerate more errors in finger placements (i.e., the position of contact points) and more comfortable to be executed. Several works [25-28, 99] synthesized a grasp with large ICRs by sampling a sub-optimal force-closure grasp then expanded independent contact regions around its contact points. Our algorithm also follows this concept, but instead of working on a single grasp at a time, it expands several force-closure grasps simultaneously. Our expansion process performs in the iterative fashion where ICRs would gradually expand to adjacent regions until finding a grasp with the largest ICRs. This process utilizes our recursive definition of ICRs, which allows efficient parallel implementation on GPU.

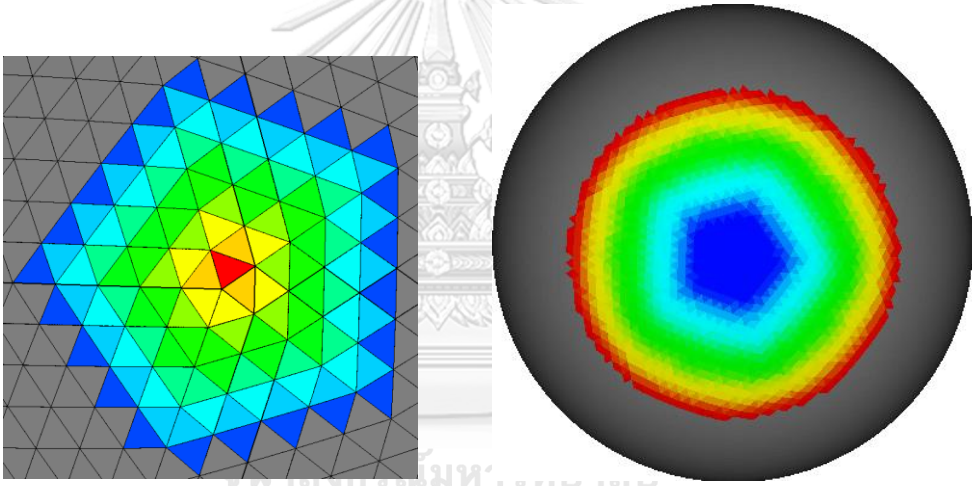


Figure 4 The left image shows examples of adjacency sets A_i^h *on a spherical mesh centered at a red triangle* t_i *with various adjacency distance* h *ranging from* 0 *(red) to* 9 *(blue). The right image shows examples of stable sets* S_i^h *associated with each adjacency set on another side of the sphere. Note that, by definition, lower adjacency sets are a subset of higher ones (*$A_i^h \subset A_i^g \mid g > h$*) and higher stable sets are a subset of lower ones (*$S_i^h \subset S_i^g \mid g < h$*)*

3.2.1 Concept

In our work, a triangular mesh represents the object surface. The ICRs that is of our interest is a set of triangles. Besides the convenience of representation, this simplification allows us to take full advantage of connectivity of mesh structure and

simplify the expansion process. Let consider the triangular mesh of object surface as an undirected graph, G, where each node denotes a triangle in mesh, and an edge connects between adjacent triangles which share the same triangle's edge. The number of edges in the shortest path between associated nodes in G is the *adjacency distance* between two triangles (see the left image in Figure 4). We define contact regions in term of adjacency distance between nodes in G called *adjacency set*.

Definition 1 An adjacency set A_i^h is defined as a set of all triangles t_j such that adjacency distance between t_j and t_i in G is not greater than h.

Our method represents ICRs as a pair of adjacency sets at the same adjacency distance. To quickly determine which pair of adjacency sets satisfies ICRs condition, let define a set of triangles that satisfy ICRs condition with adjacency set A_i^h as *stable set* S_i^h (see Figure 4).

Definition 2 A stable set S_i^h is defined as a set of all triangles t_j that can form ICRs with every triangle in the adjacency set A_i^h. In other words, triangle t_j forms ICRs with adjacency set A_i^h.

From the definition of the stable set, it is trivial to prove that any pair of adjacency set forms ICRs if they are a subset of each other associated stable set.

Corollary 3 A pair of adjacency set (A_i^h and A_j^h) forms ICRs if and only if it satisfies one of the following conditions: $A_i^h \subset S_j^h$ or $A_j^h \subset S_j^h$.

The smallest adjacency sets having zero adjacency distance are the sets of one triangle, i.e., $A_i^0 = \{t_i\}$. For their associated stable sets S_i^0, we determine ICRs condition between triangles: t_i and all other triangles. A pair of triangles forms ICRs for two-fingered grasp if all pairs of their extreme points, one from each triangle, satisfied the force-closure condition. See Appendix A and B for more details about ICRs condition for two triangles.

For adjacency sets and stable sets with higher adjacency distance, we construct them from recursive relationship using simple set operations. In another word, a higher-distance adjacency set A_i^{h+1} is the union of lower ones: A_i^h and all A_j^h where associated triangles t_j are adjacent to triangle t_i. Similarly, a new stable set S_i^{h+1} is the intersection of lower ones: S_i^h and S_j^h where associated triangles t_j are adjacent to triangle t_i. The mathematic formulations of those recursive relations are:

Adjacency set	Stable Set
$A_i^0 = \{t_i\}$	$S_i^0 = \{\forall t_j \mid (t_i, t_j)\ forms\ ICRs\}$
$A_i^1 = \{t_i\} \cup \{\forall t_j \mid t_j\ adj.to\ t_i\}$	$S_i^h = \bigcap_{\forall j \mid t_j \in A_i^1} S_j^{h-1}$
$A_i^h = \bigcup_{\forall j \mid t_j \in A_i^1} A_j^{h-1}$	

3.2.2 Algorithm

From the recurrent relations and Corollary 3, ICRs expansion consists of series of set operations on those sets. Let A^h and S^h denotes a collection of all adjacency sets and stable sets with adjacency distance h, respectively. We implement those collections as $n \times n$ matrices where n is the number of triangles in object mesh. Each row in a matrix $\mathrm{M}[i,*]$ represents all members in a set M_i where each cell $\mathrm{M}[i,j]$ is *true* if $t_j \in M_i$, otherwise, it is *false*. From this point onwards, we will use

notions of sets (A_i^h, S_i^h), collections (A^h, S^h) and matrices (A, S) interchangeably since they represent the same thing.

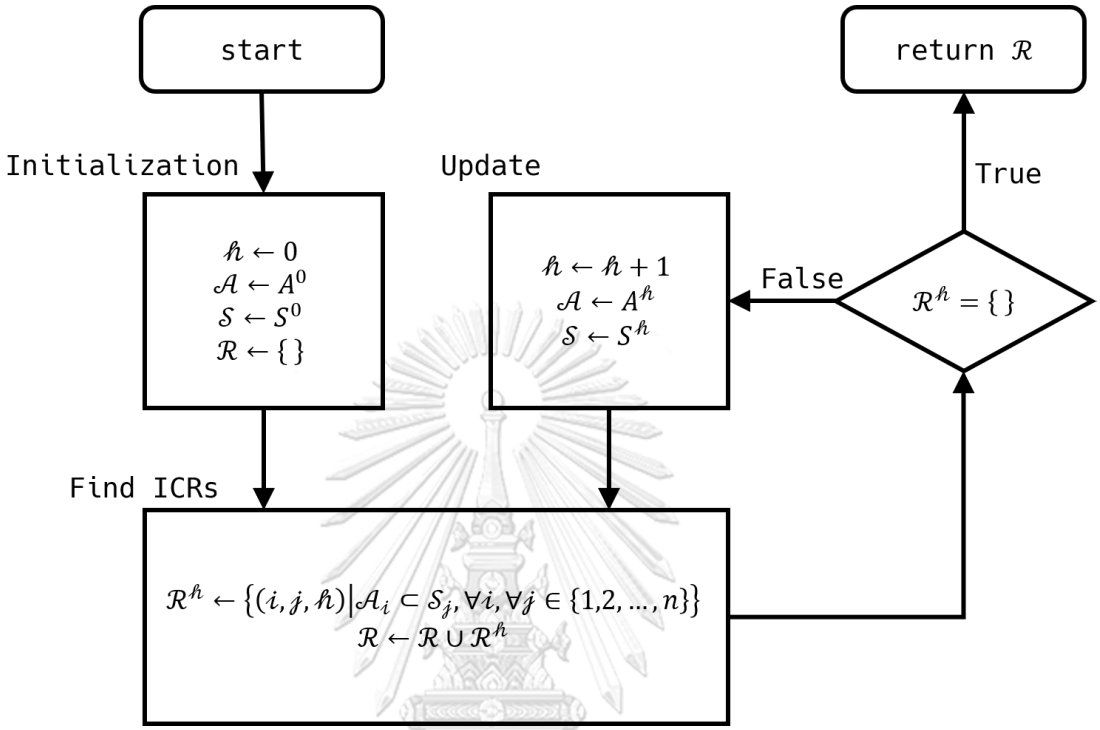


Figure 5 Flowchart of the algorithm. The scripted characters, $\mathcal{A}, \mathcal{S}, \mathcal{R}, \hbar, i, j$, denote variables in our algorithm while non-scripted characters A, S denote collections of adjacency sets and stable sets defined as recurrent relations in Section 3.2.1, respectively.

Figure 5 visualizes the overall process of our method. Starting from zero adjacency distance, $\hbar = 0$, The first step of our algorithm initializes two matrices: $\mathcal{A}$, and $\mathcal{S}$. The first one is, by definition, an identity matrix while the latter one is the results of ICRs tests between all pairs of triangles. Then, the process alternates between finding ICRs and updating matrices to higher adjacency distance. The algorithm finds all possible ICRs in current adjacency distance by testing which pairs of adjacency sets, $\left(\mathcal{A}_i^\hbar, \mathcal{A}_j^\hbar\right)$, form valid ICRs via a subset operation: $A_i^h \subset S_j^h$. The algorithm saves valid ICRs as triplets $(i, j, \hbar)$ in a collection $\mathcal{R}^\hbar$. If the current ICRs collection is empty, the algorithm is terminated and return $\mathcal{R}$, all ICRs found so far.

Otherwise, the algorithm increases adjacency distance $\mathcal{h}$ by one and computes new matrices, $\mathcal{A}$ and $\mathcal{S}$, using recurrent relation defined in the previous section. The algorithm continues back and forth between finding and updating until there is no new ICRs found in finding ICRs step. The returned collection $\mathcal{R}$ are all pairs of adjacency sets having the same adjacency distance and satisfy ICRs condition. Appendix C discusses the pseudo-code of this algorithm in more details. Figure 6 visualizes notable sets: A_i^h, S_i^h, A_j^h at several iterations of our algorithm on a spherical object.

3.2.3 ICRs Comparison

After we obtain a set of valid ICRs from our algorithm, we need to find a method to choose the best ICRs among the candidates. When a robot is grasping an object, it only needs one solution, the optimal ICRs, preferably large and the size of its contact regions are equally the same. The contact points at its center represents the preferable positions to grasp an object. So, we introduce a simple method to measure the quality of any ICRs from its contact regions' size. Let N_1 and N_2 be the number of triangles in two contact regions of given ICRs. Their quality is calculated as:

$$Q(N_1, N_2) = (N_1 + N_2) \times \left(\alpha - \frac{|N_1 - N_2|}{N_1 + N_2} \right)$$

The constant α is a coefficient that weights between two properties of ICRs: the total size of contact regions ($N_1 + N_2$) and the difference in size of contact regions ($|N_1 - N_2|$). In this work, we choose $\alpha = \frac{4}{3}$ which makes the ratio of quality between the best and the worst ICRs being 4 to 1. To be precise, let $2n_t$ be the total number of triangles of all contact regions in ICRs. The best ICRs has its contact regions with equal size: $N_1 = N_2 = n_t$. While the worst one has one of its contact regions with no triangle, i.e., $N_1 = 2n_t; N_2 = 0$. So, the constant α is calculated by solving the equation:

$$Q(n_t, n_t) = 4Q(2n_t, 0)$$

Using this method to measure the quality of all valid ICRs produced by our algorithm, we choose contact points at the center of each contact region in the highest quality ICRs to represent the optimal place to grasp a given object.

3.2.4 Optimization and parallelization

The runtime of our algorithm rapidly grows proportional to the number of triangles of an object. In our preliminary experiment, a naïve implementation based on the pseudo-code shown in Appendix C takes up to several minutes to process an object with 10,000 triangles. This slow performance is not suitable for real-time application. So, we exploit the fact that our algorithm mostly composed of simple set operations which are embarrassingly parallel and deploy several optimizations and parallelization techniques to improve its performance. We test the performance of our parallel implementation using Open Multi-Processing [100], Compute Unified Device Architecture [101] in the next section.

Open Multi-Processing, also known as OpenMP, is a multi-threading library that manages parallel regions of the code through a set of unique keywords and APIs. It can efficiently parallelize any for-loop block with a single line of code: ***#pragma omp parallel for***. The OpenMP library will automatically fork new threads, distribute workloads in each iteration of a for-loop block and concurrently execute them on a multi-core CPU. By default, the number of running threads matches the number of physical cores of CPU. In our OpenMP implementation, we ensure that each iteration in the for-loops are independent of each other and parallelize every outer for-loops to make it run several times faster than the single-threaded implementation.

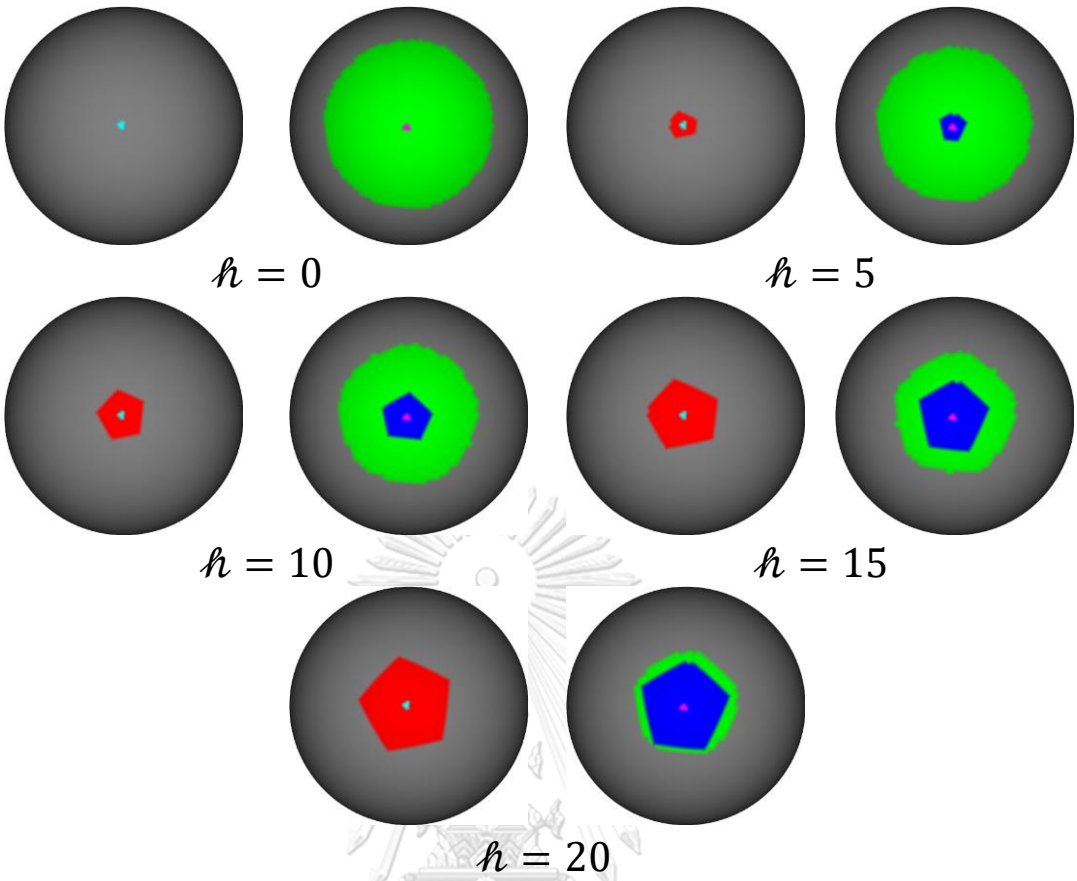


Figure 6 Visualization of ICRs represented by a pair adjacency sets, A_i^h *(red) and* A_j^h *(blue), at different iteration of our algorithm. The teal dot centered at left spheres is a triangle* t_i *while pink dot on the right side is a triangle* t_j *on the opposite side of the sphere. The green region on the right side is a stable set* S_i^h*. The blue region lying inside the green region implies that* A_j^h *is a subset of* S_i^h*, therefore* $\left(A_i^h, A_j^h\right)$ *are ICRs.*

Compute Unified Device Architecture, also known as CUDA, is a parallel framework for general purpose programming with Nvidia graphics card. It allows developer direct access to parallel computation power within GPU through a particular function called *kernel*. Like parallel for-loop in OpenMP, a program concurrently executes code in a kernel function in one or more groups of *CUDA threads* in GPU called *blocks*. The hierarchy of threads and blocks provides a more detailed level of control for thread's resource management in GPU such as shared memory and

thread's synchronization. In our CUDA implementation, we develop several kernel functions to distribute and process the workloads in each for-loop block of original pseudo-code shown in Appendix C. Our kernel functions follow the performance guideline presented in [102] to maximize GPU power to its utmost potential. For example, the number of CUDA threads and blocks for each kernel function are adjusted according to occupancy calculator [103] to maximize utilization of GPU resources. Minimize overhead due to data transfer such as allocating data with the aligned memory address, maximize coalescing data access in global memory and utilize the shared memory for frequently used data to reduce redundant global memory access. For more detail in CUDA optimization, please refer to [102].

Lastly, the most crucial part of our optimization is the data structure that represents a collection of adjacency set $\mathcal{A}$ and stable set $\mathcal{S}$. In the previous section, we represented those sets are as a boolean matrix which stores membership of a set individually in a boolean variable to simplify our algorithm, but in the real implementation, a boolean matrix is very inefficient in both memory space and processing time. So, we encode a boolean matrix in a matrix of 32-bit integers and utilize a bit-wise operation, e.g., bitwise-or and bitwise-and, to efficiently perform set operations such as union and intersection, respectively. To be precise, a $n \times n$ boolean matrix is converted to a $n \times \left\lceil \frac{n}{32} \right\rceil$ integer matrix where the $(j \bmod 32)^{\text{th}}$ bit of the integer at the i^{th} row and the $(j/32)^{\text{th}}$ column represents the membership of t_j in a set M_i. Furthermore, since an adjacency set is usually small, we use sparse matrix format called Compressed Row Storage (CRS) to only keep track of the non-zero integers in a collection of adjacency set $\mathcal{A}$. The CRS format significantly improve the efficiency of subset operation in finding ICRs step of our algorithm.

3.2.5 Experiment

We verify our algorithm with several test objects as shown in Figure 7. Some of them are synthesized from 3D modeling program [104] while the rest are real objects

obtained from 3D scanner available on the internet [105, 106]. All test objects are preprocessed (surface resampling and artifact removal) with MeshLab [107] to make their triangles relatively equilateral and have approximately 20,000 triangles (except for sphere S, M and L). In the experiment, we test our algorithm with three values of half-angle of friction cone θ: 10, 15, 20 degrees. To verify parallelization of our algorithm, we implement it in C++ for a total of five versions: CPU, OMP1, OMP2, OMP4, and GPU. The first one is a standard implementation with no parallel computation. OMP1, OMP2, and OMP4 run concurrently on CPU using OpenMP. The suffix number indicates the number of threads used in that version. The final one runs concurrently on GPU using CUDA.

We implement the testing program with Visual Studio 2013 on Window 8.1. The hardware specifications are Intel Core i5 (3.2GHz with four processor cores), 8GB of RAM and Nvidia GeForce GTX 760 with 4 GB internal memory. The libraries used in the program are Open Asset Import Library [108] for reading 3D models from files, Visualization Toolkit [109] for model visualization, OpenMP 2.0 [100] and CUDA 6.5 [101] for parallel computing.

Table 1 shows the result and average runtime of all implementation for large objects. The runtime is averaged from three trials and converted to speedup ratio compared to the CPU version. The fastest version is naturally GPU while the slowest one is OMP1. OMP1 is slightly slower than CPU due to additional overhead of thread creation and management in OpenMP. A sphere object requires the longest processing time because Its ICRs are large and uniformly spread across its surface. Table 2 shows result and average runtime for spherical objects with vary size. The speedup gained from parallelization is negligible for a tiny object with a few hundred triangles. When an object has more than thousands of triangles, the GPU version outperforms other versions in every test cases.

Figure 8 depicts the best ICRs for each object. According to our ICRs quality measurement Q, the optimal ones bias toward a pair of equally-large contact regions as described in Section 3.2.3. Every version of our algorithm yields precisely same results as shown in Table 1 and Table 2. Figure 9 visualizes the relative quality of the

best ICRs associated with all triangles on each object. The lowest quality is shown as red while the highest quality is blue. A pair of flat surface on the opposite side of an object is usually the best position for grasping an object (blue region). The strange color pattern on the spherical object is caused by the arrangement of triangles on its surface. In theory, a spherical object should have uniform color (i.e., a red sphere) since the quality of ICRs at any points on the smooth sphere are equal.

3.2.6 Conclusion

In this work, we introduce a new concept to synthesize ICRs for a two-fingered grasp on a 3D object model. We prove that the intersection between a set of points and double cones is sufficient for verifying the force-closure condition between two points and between two triangles. Given an object model and friction coefficient as input, our algorithm iteratively expands contact regions centered at every triangle on the object surface and find a pair of contact regions that satisfy force-closure property and form the largest Independent Contact Region.

Our algorithm is composed of simple operations which are suitable for parallel computation. We implement and test parallel implementations on CPU using OpenMP and on GPU using CUDA. In the experiment, we test our algorithm performance with several models with more than 20,000 triangles. The experimental result shows that the parallel implementation on GPU can efficiently find the solution in less than one second on average which should be suitable for real-time application.

The concept of using adjacency set and stable set to synthesize an independent contact region is promising and has much potential to be explored. For instance, we can sacrifice the correctness of algorithm for faster computation. The idea is to apply knowledge of hand model as constraints for the initial stable set to reduce unnecessary computation in the unreachable region and improve the efficiency. In the future work, we would like to explore heuristic function to replace the costly ICR test in verification step. The triangle's position and normal are promising features to identify an independent contact region quickly. This modification may significantly improve the speed of framework at the cost of inaccuracy and possible false-positive solutions. We

also plan to support an object that represented in point cloud from a depth camera such as Kinect.

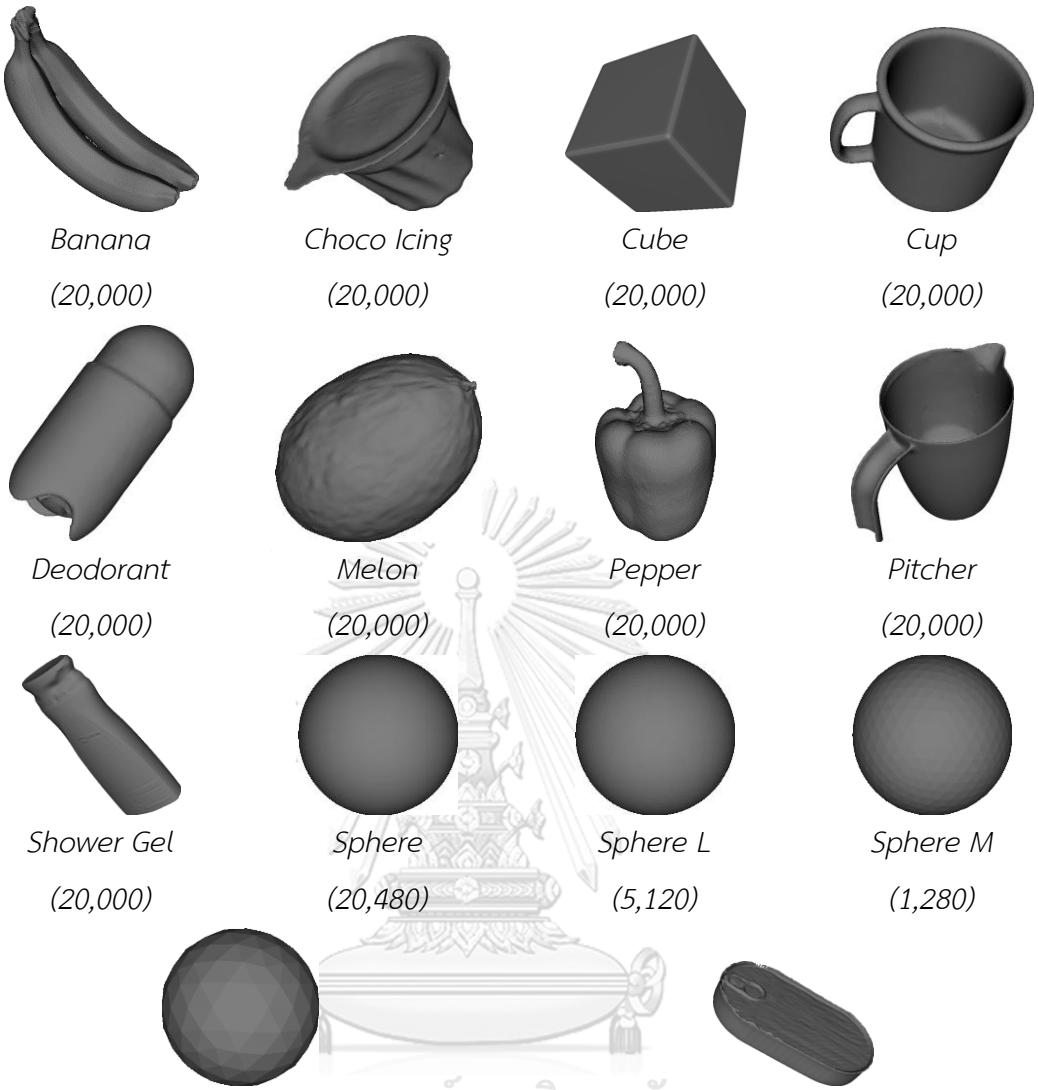


Figure 7 Image of all test objects in the experiment. The number in parenthesis indicates the number of triangles of an object. Cube and Spheres are created using Blender [104]. Choco icing, deodorant, tin, pitcher, shower gel, and cup are from KIT object model database [105]. All fruit models are 3D scanned objects from Artec Gallery [106].

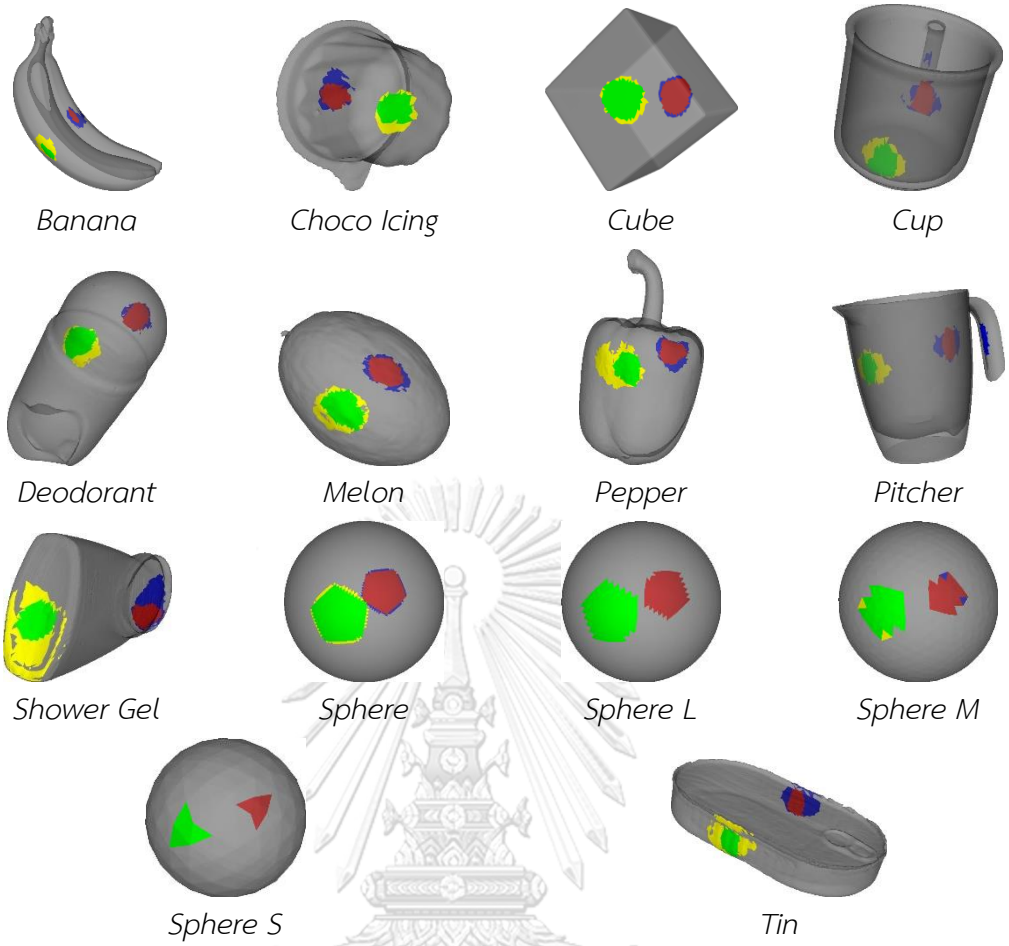


Figure 8 Transparent objects illustrate their largest ICRs when θ is 20 degrees. The ICRs are a pair of adjacency sets indicated by red and green regions which lay inside each other stable sets indicated by yellow and blue regions.

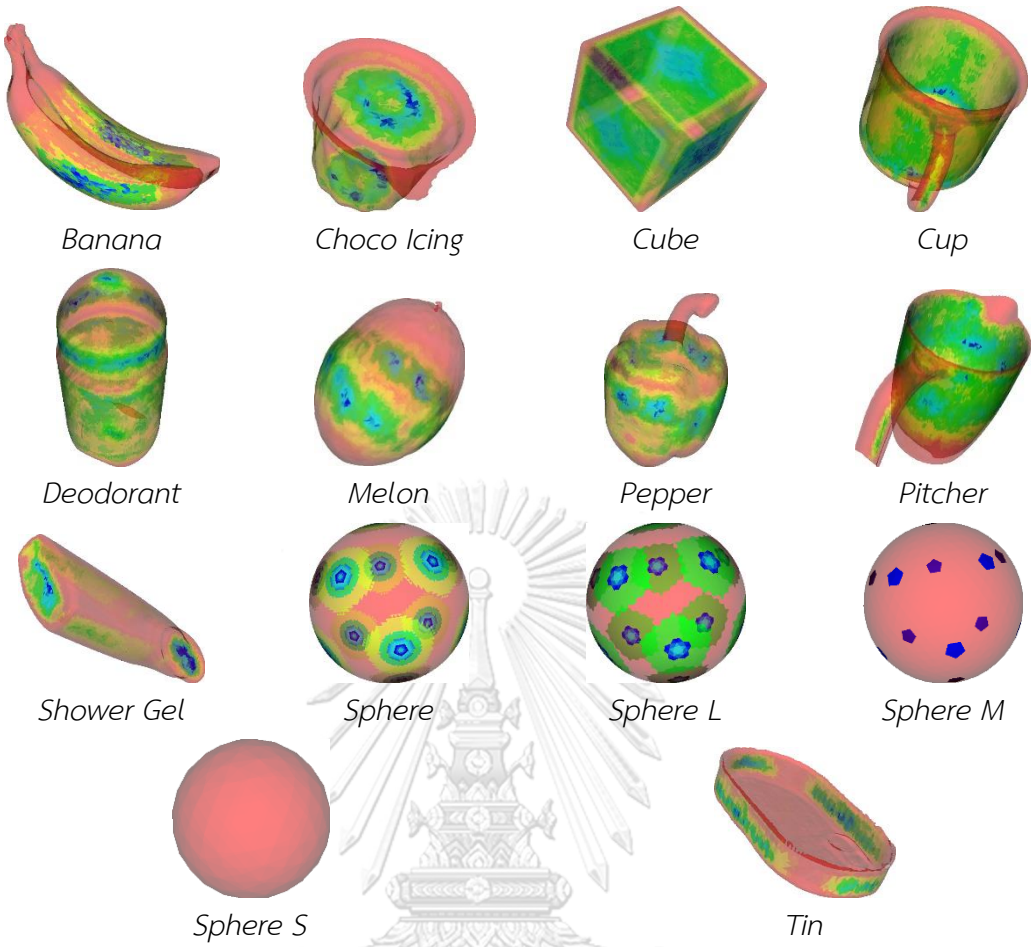


Figure 9 Transparent objects illustrate the relative quality of ICRs found in each triangle according to our measurement function Q. The quality is sorted by hue color starting from red (lowest), yellow, green and blue (highest).

Table 1 Result and runtime comparison for large objects

Object	# of triangles	θ	Best h	Best ICRs size	CPU (ms)	Speed up ratio w.r.t. CPU			
						OMP 1	OMP 2	OMP 4	GPU
Banana	20000	10	1	4, 4	3026.07	0.60	1.13	1.55	41.88
		15	3	20, 21	3566.83	0.64	1.17	1.52	23.56
		20	4	35, 35	4262.91	0.67	1.21	1.49	17.30
Choco Icing	20000	10	3	19, 21	3243.25	0.62	1.13	1.56	34.57
		15	5	50, 53	4009.50	0.65	1.15	1.55	22.79
		20	7	96, 93	5045.62	0.67	1.18	1.52	18.09
Cube	20000	10	6	54, 72	5533.68	0.70	1.27	1.51	24.88
		15	9	147, 147	7506.37	0.73	1.33	1.47	22.43
		20	13	287, 287	9901.59	0.75	1.37	1.46	20.79
Cup	20000	10	3	20, 20	4010.68	0.70	1.26	1.66	26.52
		15	5	46, 49	4898.33	0.71	1.25	1.61	19.07
		20	7	90, 89	6082.88	0.73	1.27	1.60	15.70
Deodorant	20000	10	4	33, 36	4027.44	0.64	1.10	1.54	28.50
		15	7	87, 89	5372.40	0.69	1.15	1.57	22.21
		20	10	178, 165	7147.23	0.72	1.18	1.59	19.00
Melon	20000	10	5	41, 41	3751.98	0.64	1.15	1.53	21.47
		15	7	105, 102	5031.62	0.67	1.17	1.50	15.00
		20	11	233, 246	7475.40	0.70	1.21	1.51	11.97
Pepper	20000	10	4	31, 27	3531.52	0.64	1.16	1.54	25.78
		15	6	69, 73	4734.62	0.68	1.23	1.50	17.38
		20	10	158, 184	6585.05	0.71	1.25	1.48	13.64
Pitcher	20000	10	3	20, 20	3667.75	0.65	1.21	1.64	29.73
		15	5	42, 43	4853.01	0.73	1.33	1.67	20.47
		20	6	66, 72	6244.24	0.76	1.38	1.68	16.94
Shower Gel	20000	10	4	33, 35	3350.12	0.61	1.13	1.54	31.32
		15	7	89, 82	4179.91	0.66	1.18	1.51	21.05

		20	10	177, 189	5240.01	0.68	1.22	1.49	16.29
Sphere	20480	10	10	160, 160	21005.82	0.80	1.17	1.97	28.75
		15	15	336, 336	33625.12	0.81	1.18	1.99	30.35
		20	20	575, 575	46917.55	0.81	1.20	2.01	32.21
Tin	19999	10	4	31, 34	3228.81	0.62	1.14	1.57	31.28
		15	6	64, 68	3774.61	0.65	1.16	1.58	22.14
		20	8	101, 95	4386.81	0.67	1.20	1.56	17.73
Average	-	10	-	-	5307.01	0.70	1.17	1.69	28.38
		15	-	-	7413.85	0.73	1.20	1.70	23.42
		20	-	-	9935.39	0.75	1.23	1.71	20.75
		All	-	-	7552.08	0.73	1.20	1.70	24.18

Table 2 Result and runtime comparison for spherical objects with vary size

Object	# of triangles	θ	Best h	Best ICRs size	CPU (ms)	Speed up ratio w.r.t. CPU			
						OMP 1	OMP 2	OMP 4	GPU
Sphere S	320	10	0	1, 1	1.33	0.91	1.48	1.52	1.29
		15	0	1, 1	1.37	0.74	1.30	1.52	1.21
		20	1	4, 4	1.54	0.77	1.27	1.53	0.96
Sphere M	1280	10	1	4, 4	17.11	0.78	1.34	2.43	5.41
		15	2	10, 10	21.14	0.74	1.36	2.47	4.47
		20	4	29, 29	27.33	0.75	1.38	2.53	3.74
Sphere L	5120	10	4	31, 31	370.56	0.69	1.15	2.40	16.62
		15	7	79, 79	549.82	0.71	1.25	2.50	14.85
		20	9	132, 132	761.30	0.73	1.28	2.64	13.58

Chapter 4 Grasping with machine learning

With the current technology, it is still impossible to predict or simulate real-world physical interactions correctly in a computer. Thus, a grasp planning based on solely physical properties such as force-closure condition does not work well in practical application. It is inflexible and fails to deal with random errors and noises which are very common when working with real-world data. In our opinions, grasp quality that determines the success rate of a grasp is a very subjective term which depends on several factors, e.g., task objective of a grasp, characteristic of robot's arm and manipulator and the surrounding environment. An analytical grasp planning that only focuses on contact points is unaware of changes in those factors. It will only choose the same grasp for the same object in every situation which may be suitable for some cases, but inadequate for the rests. On the other hand, a grasp planning that based on learning from examples, also known as data-driven grasping and empirical approach, is more practical and versatile because it can implicitly deal with those issues with a learning model such as support vector machine and neural network. Given enough time and samples, it is an excellent choice for dealing with erroneous data and hidden factors in the grasping problem. In general, a learning model significantly improves the success rate of a grasp compared to the conventional methods that assume complete data and ideal scenario. Thus, the empirical approach was gradually becoming more and more popular since the beginning of the 2000s to this date.

In the light of this new approach, we investigate the cooperation between our previous methods, caging and ICRs, and learning model. The general idea is to extract features related to caging and ICRs and use them to predict grasp quality in a learning model along with other notable properties of a grasp such as a curvature at contact points and distance to the center of an object. We demonstrate our idea by implementing it as a neural network that predicts grasp quality for a two-fingered grasp in 2D and tests our neural network with both synthetic and human-labeled data. The

synthetic data uses a heuristic method to generate grasp quality of known objects. We gather the human-labeled data from undergraduate students in our department.

4.1 Learning model

Machine learning dominates many fields of research ranging from speech recognition and synthesis to image segmentation and labeling. Many grasp planning methods utilized machine learning as discussed in (see Section 2.3). Their approaches relied on visual-based object recognition, i.e., learn to recognize a good grasp based on labeled images in a training set. The existed methods usually learned grasp quality as a binary classification problem, i.e., labeling grasps as good or bad only. Furthermore, their works often used image descriptors as the input of their learning models such as the scale invariant feature transform (SIFT) and the histogram of oriented gradients (HOG) or, otherwise, directly use the image from the visual sensor. This visual-based approach might work well in object recognition and image processing problems, but we highly doubt its performance in robotic grasping since grasping is not solely about the image of an object to be grasped. It involves physical interactions between hand and object which cannot be deduced from the image. So, unlike existed methods, we apply knowledge from grasping theory and utilize grasping features such as force-closure, independent contact regions and other heuristics as the input of our neural network. Additionally, we choose regression approach to learn grasp quality and map output, i.e., grasp quality, to a decimal starting from 0 (bad) to 1 (good) instead of binary classification. The fundamental idea is to let our neural networks recognize grasp quality based on a grasp, not an object. Thus, our methods should be able to cope with unknown objects with ease.

In this work, we focus on a two-fingered grasp of a polygonal object in 2D, i.e., a pinch grasp to pick up an object from the top. We train a neural network to predict grasp quality from a grasp, a pair of contact points on a polygon. To find the best grasp for a given object, it randomly samples a set of grasps on that object, predicts their quality and returns a grasp with highest grasp quality as output.

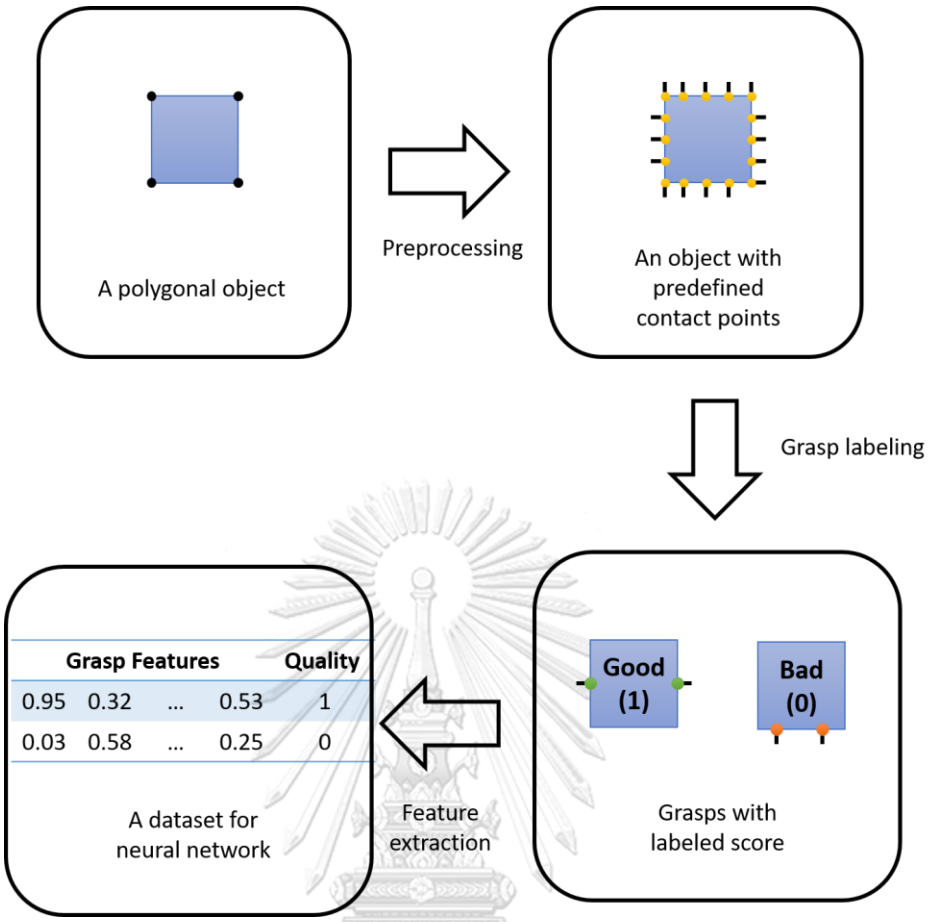


Figure 10 An overview of preparing training data for our neural network. An object is a simple polygon (blue square) defined by its four corners (black dots). The preprocessing step generates its contact points, along with their normal, (yellow dots with black lines). Next, a grasp defined as a pair of contact points is labeled with its quality and transforms to an instance in the dataset by extracting its features.

4.2 Data representation

An object is a simple polygon (no self-intersecting edges) denoted by a series of its vertices. It is preprocessed to generate a predefined set of contact points. We denote a grasp as a pair of contact points in this predefined set, and its quality is a scalar value between zero (bad grasp) to one (good grasp), inclusive. To learn grasp quality, we extract features of a grasp as a scalar vector and map each value to a

unique feature of a grasp discussed in the next section. Our method normalizes the values within the range between zero and one, inclusive.

Figure 10 illustrates the overall process of preparing grasp data to be learned in a neural network. The first step is preprocessing an object by generating its contact points. The contact points consisted of position and normal are spread evenly on the object surface. In this work, we fix the number of contact points on an object at one hundred. Next, we label grasps defined as a pair of contact points from the preprocessing step by its quality. We collect grasp quality from two sources: synthetic data and human-labeled data as discussed in Section 4.3. Then, for each labeled grasp, a set of grasping features is extracted from contact points and an object. Section 4.4 discusses heuristic methods for feature extraction used in this work. Finally, a dataset for training our neural network is a scalar matrix of size $n \times (m + 1)$ where n is the number of labeled grasp and m is the number of grasping features.

4.3 Grasp quality labeling

In training dataset, there are two kinds of labeled grasps: synthetic and human-labeled. The synthetic data is a predefined set of labeled grasps based on heuristic methods. It ensures the fundamental knowledge of grasp quality according to ordinary senses of most humans and filters out grasps that are trivially not feasible. For examples, a grasp having both contact points on the same edge of an object is inadequate, so its quality should be zero. On the other hand, a grasp on the opposite edge of a square should be good grasp and have high quality. Appendix D discusses the heuristic methods to generate synthetic data.

The second type of the dataset is grasps labeled human. We believe that learning grasps labeled by a human should provide adaptability for our neural network to handle the implicit knowledge of grasping such as task objective and properties of the environment. In this work, we assume the task as picking up an object from tabletop using a pinch grasp and ask grasp's quality from human, but in the future, it should gradually replace human-labeled grasps with the actual grasps from real robot when it is performing the task.

Our first attempt at collecting human-labeled grasps was using the survey program (Figure 11) to ask the grasp quality from volunteers in our engineering department. The program randomly generates a grasp on an object and asks a user for its quality ranging from zero (bad grasp) to one (good grasp). The program only generates force-closure grasps since randomly generating grasps with no precondition produced mostly undesirable grasps with a minimal set of good grasps. The users can skip, stop and resume the survey at any time they wished and send the results through the internet.

In the preliminary experiment, we found that a neural network failed to recognize grasps labeled by volunteers and incorrectly classified the undesirable grasps as the good ones. We speculate that several reasons for this unsatisfactory outcome in the Section 4.5.1. In the second attempt, we improved the survey by introducing five labels associated with the grasp quality: Worst (0), Bad (0.25), Mediocre (0.5), Good (0.75), and Best (1). The new survey had 100 predefined grasps per object for a total of 1,300 grasps. To make a balanced dataset, we trained a neural network using $\mathcal{D}_s$ and used it to classify grasps into 3 groups: the high-quality grasps with a score higher than 0.8, low-quality grasps with a score lower than 0.2 and the rest are middle-quality grasps. we distributed the 100 grasps per object in the following ratio: 20 high-quality grasps, 20 low-quality grasps, 51 middle-quality grasps, and 9 duplicated grasps. We randomly selected 3 duplicated grasps from each prior group to examine the consistency of the collecting data. Lastly, instead of volunteer, we survey three individuals who have background knowledge in robotic grasping using the new program shown in Figure 12.

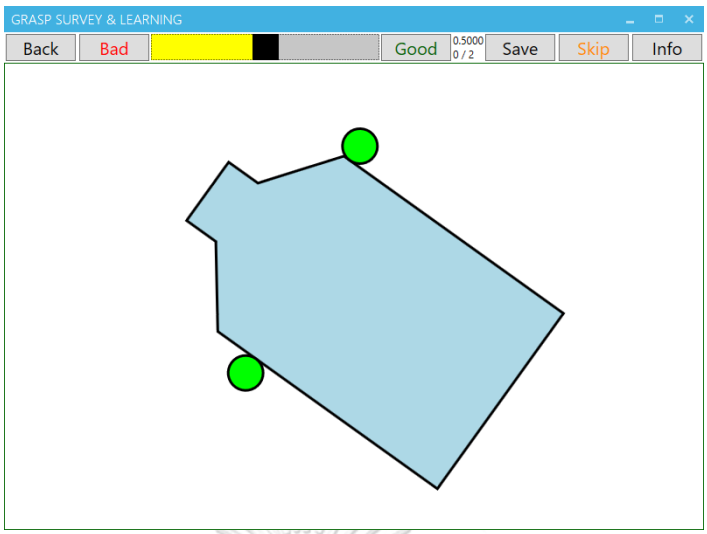


Figure 11 A simple survey program for gathering human-labeled data from volunteers. The program assumes a simple scenario where a volunteer is picking up an object (the blue polygon) from tabletop using a pinch grasp (the green circles) and then ask the quality of that grasp.

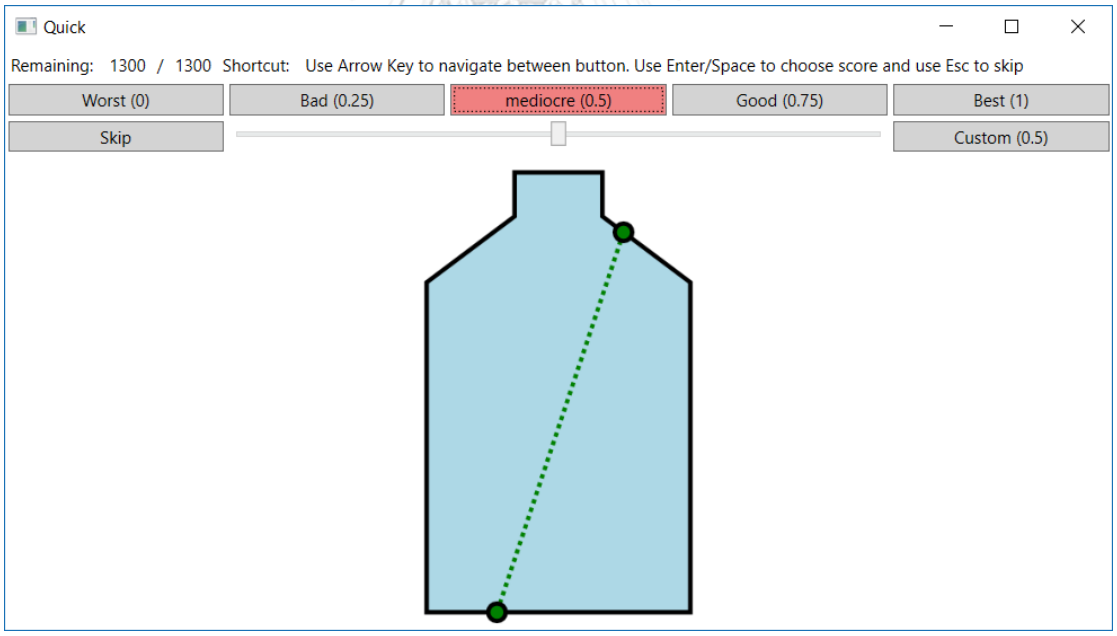


Figure 12 An improved version of the survey program after receiving feedback from the first one. It has five separate labels for grasp's quality and a fixed number of grasps in the survey (100 grasps per object; for a total of 1,300 grasps). A user needs to label all grasps to complete the survey. It collected grasps labeled by three individuals who are knowledgeable about the grasping theory.

4.4 Grasping features

The grasping feature is an essential part of our neural network. They define learning capability of the network since grasp's feature is the input data that directly map a grasp to its quality. So, grasping features should have several criterions: 1) Its value can be normalized (i.e., have a fixed finite range). 2) The features should not be affected when the workspace is arbitrarily translated and rotated (i.e., invariant to translation and rotation). 3) It should provide a meaningful semantic to make our neural network quickly learn grasp quality from the dataset.

In a neural network, we use a total of 8 grasping features: force-closure (f_{fc}), heuristic force-closure (f_{hfc}), linear mobility (f_{lm}), approachability (f_a), distance to the center of an object (f_{oc}), the curvature of contact points (f_c), size of ICRs (f_{is}) and density of ICRs (f_{id}). Those features are inspired from heuristic methods and conventional grasping theory. Roa and Suárez [21] reviewed a collection of notable grasp quality measurements that can be used as grasp's features. According to their work, our features are associated with the location of contact points which can be further classified as features based on geometric relation (f_{oc}, f_{is} and f_{id}) and features based on the limitation of finger's forces (f_{fc} and f_{hfc}). Unlike other features, linear mobility, approachability and curvature are not grasp quality measurement. They are heuristic methods that encode the location of contact points based on object's shape near that location.

The first feature f_{fc} is derived from the force-closure condition of a two-fingered grasp in [22] which stated that *the segment between two contact points must strictly lay within friction cone of those contact points*. To put it simply, f_{fc} is calculated from the largest angle between normal of contact points and the segment between those two points. Let $(\boldsymbol{p}_a, \boldsymbol{n}_a)$ and $(\boldsymbol{p}_b, \boldsymbol{n}_b)$ be position and normal of contact points in a grasp. Cosine of the largest angle $\hat{f}_{fc}$ can be calculated as:

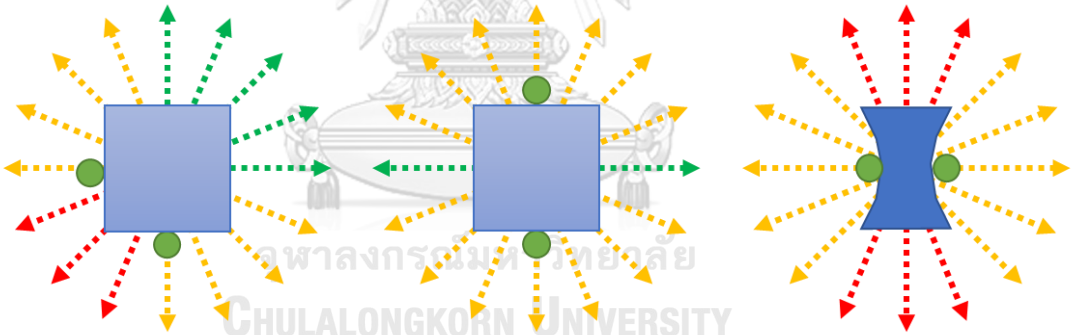
$$\hat{f}_{fc} = \min\left(\boldsymbol{n}_a \cdot \frac{\boldsymbol{p}_a - \boldsymbol{p}_b}{\|\boldsymbol{p}_a - \boldsymbol{p}_b\|}, \boldsymbol{n}_b \cdot \frac{\boldsymbol{p}_b - \boldsymbol{p}_a}{\|\boldsymbol{p}_b - \boldsymbol{p}_a\|}\right)$$

A grasp is force-closure if the angle $\text{acos}(\hat{f}_{fc})$ is less than half-angle of friction cone θ. The force-closure feature f_{fc} is the normalized value of the largest angle $\hat{f}_{fc}$:

$$f_{fc} = \begin{cases} 0, & \text{if } \hat{f}_{fc} > \cos(\theta) \\ \dfrac{\hat{f}_{fc} - \cos(\theta)}{1 - \cos(\theta)}, & \text{if } \hat{f}_{fc} \leq \cos(\theta) \end{cases}$$

The heuristic version of the force-closure feature f_{hfc} is a special case of f_{fc} such that half-angle of friction cone is maximized; $\theta = 180^{\circ}$:

$$f_{hfc} = \frac{\hat{f}_{fc} + 1}{2}$$



$f_{lm} = 0.35625$ $f_{lm} = 0.675$ $f_{lm} = 0.9375$

Figure 13 Examples of linear mobility feature f_{lm} sampled uniformly in 16 directions. The color of an arrow indicates the number of contact points (green dots) that collide with an object (blue polygon) if the object moves in that direction. The red ones collide with both contact points while the yellow ones collide with exactly one contact point and the green ones did not collide at all.

The linear mobility feature f_{lm} is a simplified version of caging. We did not use caging as a grasping feature directly because 1) Two contact points are insufficient to cage a convex object. 2) Caging only determines if a grasp cage an object or not. It

lacks a meaningful way to differentiate two distinct grasps that both cage (or not cage) an object. On the other hand, the partial cage introduced in the Section 3.1 is not suitable to represent caging feature due to its slow verification process. Thus, we simplified the concept of the partial cage as the linear mobility which only considers the linear motion of an object that can escape a grasp. In other words, if an object is only moving in a straight line, how many directions an object could move arbitrarily far without colliding with contact points? A good grasp should limit object movement as much as possible. This feature is consistent and provide more level of detail than caging. A caged object cannot move arbitrarily far from a grasp, it is also cannot move arbitrarily far in any direction. On the other hand, if an object can move arbitrarily far in at least one direction, it means that a grasp failed to cage an object.

We test the object mobility in the workspace by casting rays from each contact point in the opposite direction. If any rays collide with an object, an object cannot move arbitrarily far in that direction. In this work, we measure linear mobility by the ratio of non-movable directions sampled uniformly from 16 fixed directions as shown in Figure 13. Note that, this method is sensitive to workspace's rotation, e.g., if the workspace of the middle square in Figure 13 rotates a little bit, all sampled direction will make an object collides with a grasp and the value of f_{lm} will change. This behavior can be mitigated by increasing number of sampled directions.

Let N_d be the number of sampled direction and R_c^i be the set of directions that an object is blocked by a contact point $i \in \{1,2\}$. The linear mobility feature is calculated as:

$$f_{lm} = 0.9 \times \frac{2}{N_d} \times \max\left(0, \|R_c^1 \cup R_c^2\| - \frac{N_d}{2}\right) + 0.1 \times \frac{\|R_c^1 \cap R_c^2\|}{N_d}$$

The first term is the normalized value of object's linear mobility based on the number of directions that at least one contact point blocks the object (the number of red and yellow arrows in Figure 13). The second term is the redundant factor of linear

mobility based on the number of directions that is blocked by every contact points (only the red ones).

The approachability feature f_a is closely related to linear mobility feature. It tests if contact points in a grasp can easily be reached when the robot moves fingers in a straight line toward them. It is calculated from the summation of predefined directions that contact point does not collide with an object weighted by the angle between that direction and normal of a contact point. Let $\boldsymbol{r_j}$ be the j^{th} vector of a set of directions as defined in linear mobility feature and $F_{i,j} \in \{0,1\}$ is a boolean value indicated that a contact point $\boldsymbol{i}$ can move arbitrarily far without colliding with an object in $\boldsymbol{r_j}$ direction or not. The approachability feature is calculated as:

$$f_a = \frac{\sum_{i=1}^{2}\sum_{j=1}^{N_d} \max(0, F_{i,j} \times \boldsymbol{r}_j \cdot \boldsymbol{n}_i)}{N_d}$$

Next, f_{oc} is a heuristic feature based on the distance between a grasp and the center of an object. It is calculated from the shortest distance from the center of an object to the segment between two contact points of a grasp ($\boldsymbol{D_c}$) normalized by the distance from the top-left corner to the bottom-right corner of object's bounding box ($\boldsymbol{D_o}$).

$$f_{oc} = 1 - \frac{D_c}{D_o}$$

The curvature f_c encodes the curviness of surface around grasp's contact points. It samples a set of contact points within $0.2 \times D_o$ radius centered at each contact points of a grasp and estimates curviness by measuring angular distances from each point to a reference hyperplane (a line in 2D). The hyperplane's origin is an average of all sampled points' positions, and its normal is a normalized vector from the sum of all sampled points' normal vectors. If reference hyperplanes cannot be determined (i.e., its normal is zero), then f_c is set to one.

Let $S_x = \{(\boldsymbol{p}_1, \boldsymbol{n}_1), (\boldsymbol{p}_2, \boldsymbol{n}_2), \dots, (\boldsymbol{p}_n, \boldsymbol{n}_n)\}$ be a set of contact points around grasp's contact point $\boldsymbol{x}$. The hyperplane's origin $\boldsymbol{p}_h$ and normal $\boldsymbol{n}_h$ can be calculated as:

$$\boldsymbol{p}_h = \frac{\sum_{i=1}^{n} \boldsymbol{p}_i}{n}$$
$$\boldsymbol{n}_h = \frac{\sum_{i=1}^{n} \boldsymbol{n}_i}{\left\| \sum_{i=1}^{n} \boldsymbol{n}_i \right\|}$$

The curviness score at a contact point $\boldsymbol{x}$ is:

$$f_c(S_x) = \frac{\sum_{i=1}^{n} \left| \boldsymbol{n}_h \cdot \frac{\boldsymbol{p}_i - \boldsymbol{p}_h}{\|\boldsymbol{p}_i - \boldsymbol{p}_h\|} \right| + \left(1 - \frac{\boldsymbol{n}_h \cdot \boldsymbol{n}_i + 1}{2}\right)}{2n}$$

The curviness score of a grasp is the mean of curviness score at each contact points:

$$f_c(S_x, S_y) = \frac{f_c(S_x) + f_c(S_y)}{2}$$

The last two features, f_{is} and f_{id}, derive from the concept of Independent Contact Regions. In this work, we limit the scope of searching ICRs to the eleven adjacent contact points centered at the current grasp. Hence, for each grasp, the method test 121 force-closure grasps. For example, if a grasp has contact points at the index $(3, 71)$, it will verify force-closure grasps formed by two sets of contact points: $(98,99,0,1, \dots, 7,8)$ and $(66,67, \dots, 75,76)$. We use a simple force-closure test in f_{fc} instead of the iterative ICRs algorithm in the previous chapter because of the size of the problem. The iterative method calculates the ICRs centered at every grasps on an object simultaneously, however, when extracting grasping feature, we only interest in a tiny set of ICRs centered at some grasps.

The ICRs density computes the ratio of force-closure grasps formed by the adjacent contact points near the current grasp. Similarly, the method calculates the ICRs size from the highest adjacency distance which all pairs of contact points within the adjacent regions still form a force-closure grasp. The method normalizes the ICRs density by the number of all possible grasps and the ICRs size by the maximum adjacency distance.

4.5 Experiment

In this work, we investigate the potential of using a neural network to predict grasp's score from grasping features defined in section 4.4. Section 4.5.1 describes the dataset used in this experiment collected from the survey and synthesized from heuristic methods. Next, we trained neural networks with different structures and used 10-fold cross-validation to find the most suitable structure to predict the grasp quality. Then, we demonstrated the satisfactory grasps according to the neural networks trained with different datasets and analyzed their results on both known and unknown objects qualitatively. Lastly, we explored the influence of each grasp's feature toward the grasp quality evaluated from the neural networks.

4.5.1 Dataset

Our neural networks were trained and tested with labeled grasps from 13 different objects as shown in Figure 14 using 10-fold cross-validation. We also qualitatively tested the neural networks on the ten unknown objects as shown in Figure 15. We group labeled grasps into several sets by their sources: synthetic dataset $\mathcal{D}_s$ (18,863 grasps), volunteer dataset $\mathcal{D}_v$ (4,922 grasps) and datasets from three individuals $\mathcal{D}_a$, $\mathcal{D}_b$, and $\mathcal{D}_c$ (1,300 grasps per dataset) as discussed in Section 4.3.

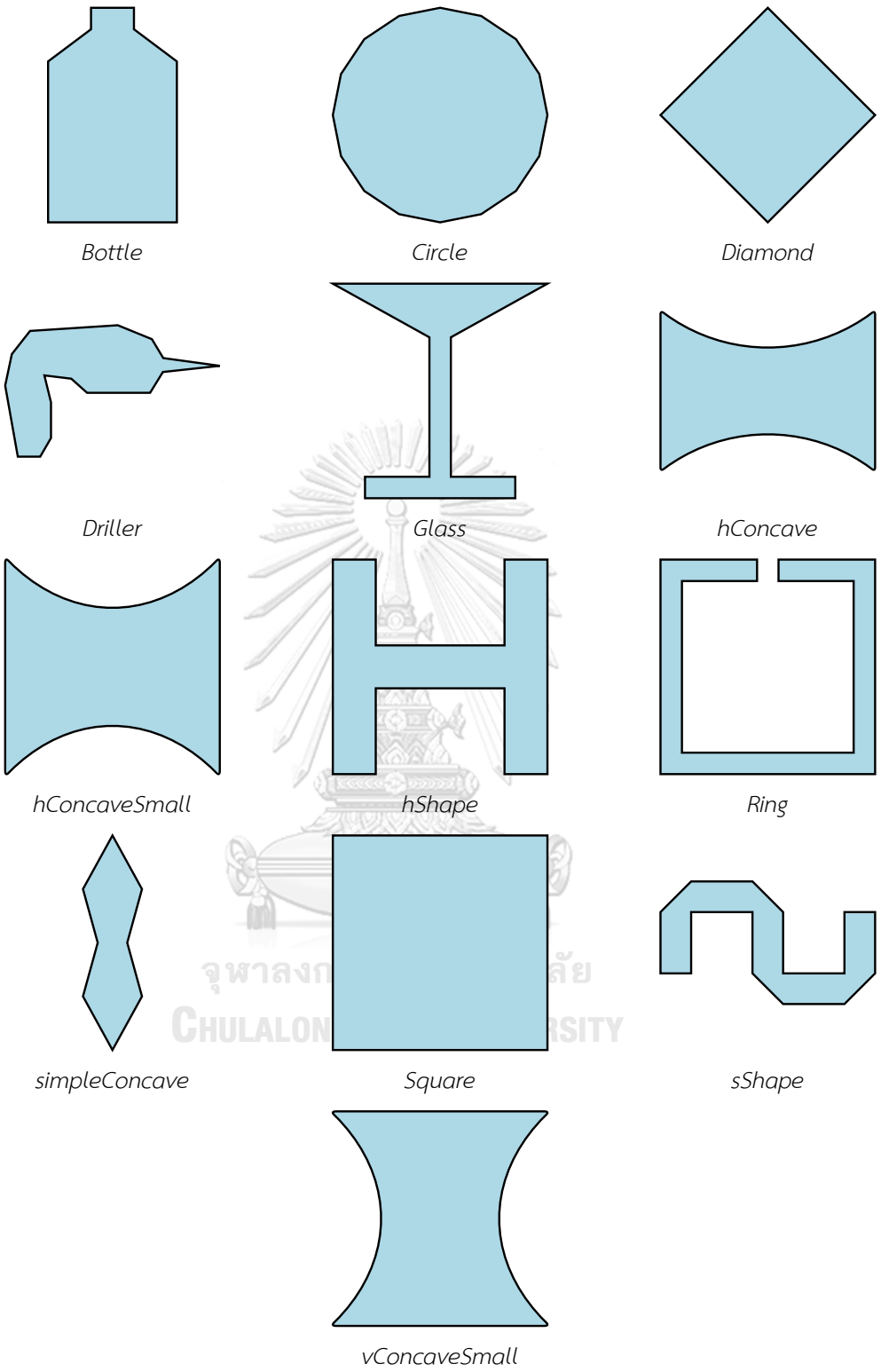


Figure 14 The test objects in the experiment.

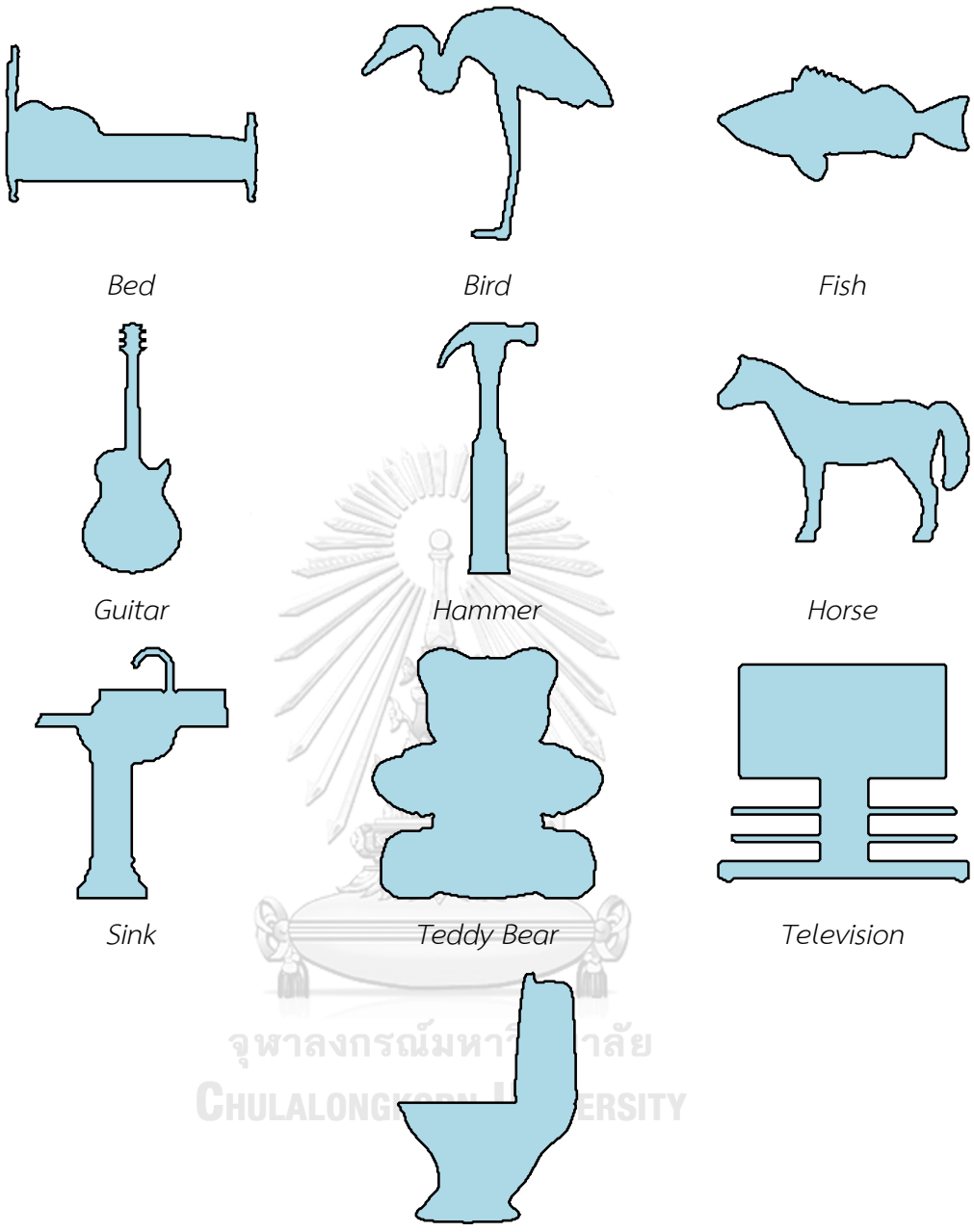


Figure 15 All unknown objects obtained from [110] available on http://www.mfdemirci.etu.edu.tr/index.html. We use OpenCV [111] to convert silhouette images to polygonal contours.

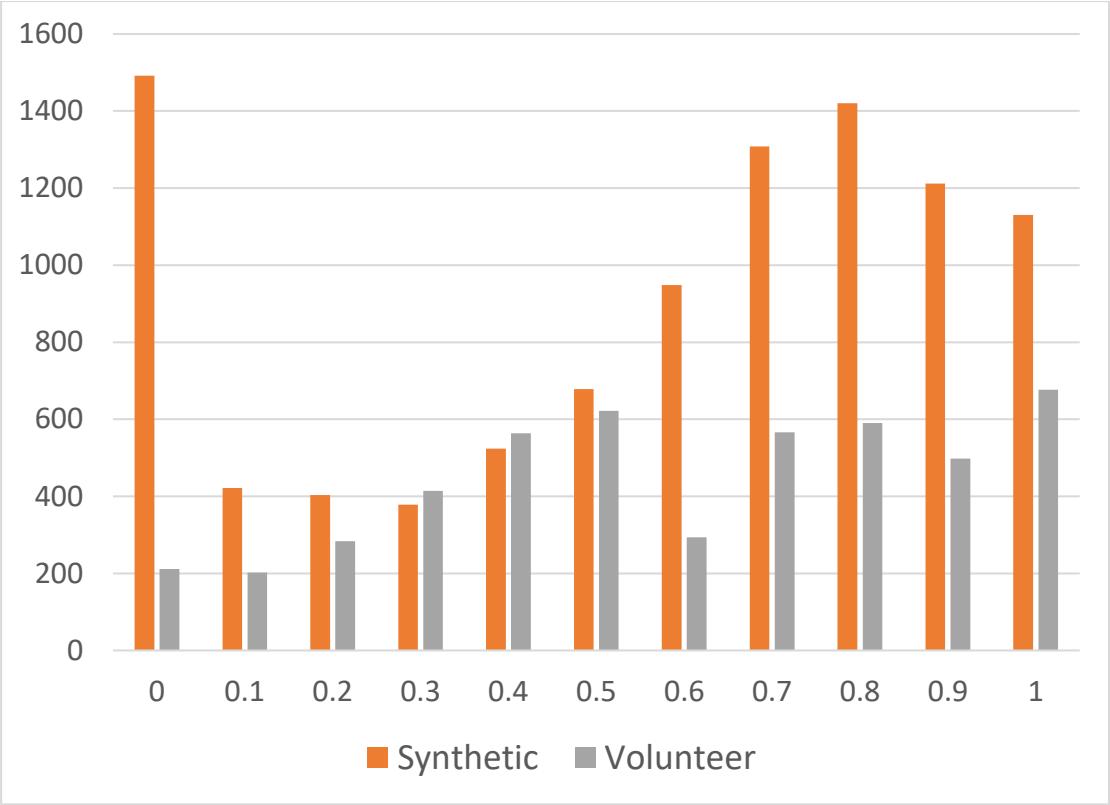


Figure 16 The histogram of the quality of grasps in the synthetic dataset ($\mathcal{D}_s$) and the volunteer dataset ($\mathcal{D}_v$). The Y-axis is the number of labeled grasps that have their quality less than or equal to the value on the X-axis. This histogram shows 9,915 synthetic grasps obtained from only five objects as described in Appendix D. It excludes 8,948 zero-quality grasps which have their contact points on the same edge of an object. So, the total number of labeled grasps in $\mathcal{D}_s$ is 18,863 grasps. On the other hand, the number of labeled grasps in the dataset $\mathcal{D}_v$ is 4,922 grasps.

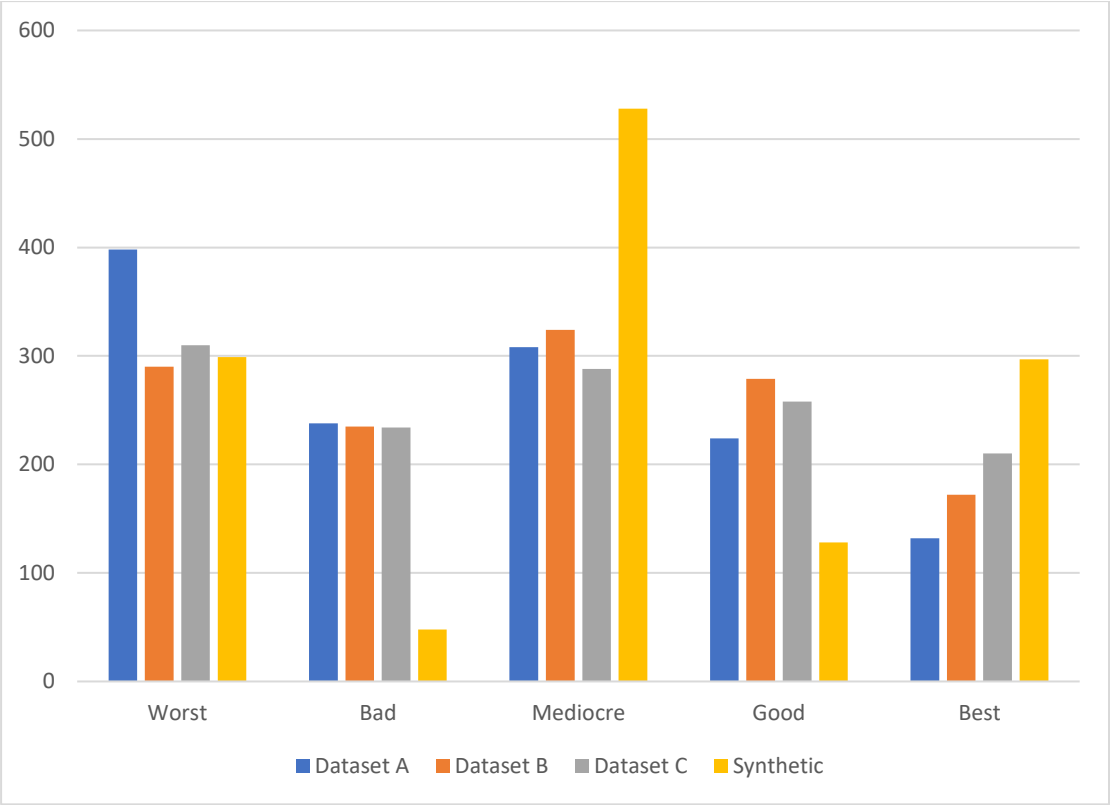


Figure 17 Histogram of labels of 1,300 different grasps from the survey of the three individuals ($\mathcal{D}_a$, $\mathcal{D}_b$, and $\mathcal{D}_c$) and the labels calculated from the neural network trained with the synthetic dataset ($\mathcal{D}_s$). Each survey consists of 100 grasps per object for a total of 1,300 grasps. We classify grasps into five labels based on grasp's score in the following order: A grasp scored lower than 0.1 is the worst. A grasp scored higher than 0.9 is the best. A bad grasp has the score lower than 0.35. A good grasp has the score higher than 0.65. The other grasps are mediocre.

Table 3 The confusion matrix and F1 score between dataset $\mathcal{D}_a$ and $\mathcal{D}_b$.

$\mathcal{D}_a$ \ $\mathcal{D}_b$	*Worst*	*Bad*	*Mediocre*	*Good*	*Best*	*F1 score*
Worst	290	70	29	9	0	*0.8430*
Bad	0	76	115	46	1	*0.3214*
Mediocre	0	68	116	118	6	*0.3671*
Good	0	21	64	89	50	*0.3539*
Best	0	0	0	17	115	*0.7566*
Root-mean-square error			*0.2193*		*Avg. F1*	*0.5284*

Table 4 The confusion matrix and F1 score between dataset $\mathcal{D}_a$ and $\mathcal{D}_c$.

$\mathcal{D}_a$ \ $\mathcal{D}_c$	*Worst*	*Bad*	*Mediocre*	*Good*	*Best*	*F1 score*
Worst	308	76	11	3	0	*0.8701*
Bad	2	106	96	33	1	*0.4492*
Mediocre	0	37	141	124	6	*0.4732*
Good	0	15	40	87	82	*0.3610*
Best	0	0	0	11	121	*0.7076*
Root-mean-square error			*0.1917*		*Avg. F1*	*0.5722*

Table 5 The confusion matrix and F1 score between dataset $\mathcal{D}_b$ and $\mathcal{D}_c$.

$\mathcal{D}_b$ \ $\mathcal{D}_c$	*Worst*	*Bad*	*Mediocre*	*Good*	*Best*	*F1 score*
Worst	289	1	0	0	0	*0.9633*
Bad	18	113	75	27	2	*0.4819*
Mediocre	3	94	129	89	9	*0.4216*
Good	0	24	80	122	53	*0.4544*
Best	0	2	4	20	146	*0.7644*
Root-mean-square error			*0.1879*		*Avg. F1*	*0.6171*

Figure 16 and Figure 17 show histograms of the grasp's quality in the datasets. We programmatically labeled 9,915 grasps in $\mathcal{D}_s$ for only five objects based on prior knowledge of the object's characteristic such as a grasp having contact points on the opposing edges of a square or grasping the concave part of an object (see Appendix D for the detailed algorithm). The other 8,948 grasps in $\mathcal{D}_s$ are not shown on Figure 16. Those grasps had zero quality due to our simple heuristics: a grasp with its contact points being placed on the same edge of an object is undesirable. On the other hand, the labeled grasps in the other datasets are evenly distributed among all objects.

From the preliminary analysis, we found several problems in the volunteer dataset ($\mathcal{D}_v$) when training the neural networks. They failed to converge after training for a predefined number of iteration and they often erroneously labeled undesirable grasps as the good ones. We speculated that the survey method had several faults and the quality of collected data was poor. For example, asking a numerical value of the grasp quality from volunteers might be inappropriate because it is hard for a human to consistently match a numerical value to the proper grasp quality. The volunteers who inexperience in the robotic grasping provided inconsistent data and significantly raised the error rate when training a neural network. Furthermore, labeled grasps from volunteers were limited to the force-closure grasps. This limitation caused the neural networks failed to correctly classify the non-force-closure grasps. So, in the second attempt, we redesigned the survey as discussed in the Section 4.3 and collected the data from three individuals to create three datasets ($\mathcal{D}_a$, $\mathcal{D}_b$, and $\mathcal{D}_c$) their combination ($\mathcal{D}_{ab}$, $\mathcal{D}_{ac}$, $\mathcal{D}_{bc}$, and $\mathcal{D}_{abc}$).

Due to the grasps collected from each volunteer being unique, we unable to make the meaningful comparison and analysis from $\mathcal{D}_v$. On the other hand, the grasps in the datasets $\mathcal{D}_a$, $\mathcal{D}_b$, and $\mathcal{D}_c$ are the same, hence, we compared their labels against each other as shown in Table 3, Table 4, and Table 5. The last column in the table shows the F1 score for each label. We also computed a root-mean-square error from the difference in numerical values associated with each label (0 for worst, 0.25 for bad, 0.5 for mediocre and so on). There was a total of 117 pairs of identical grasps in the survey. So, we verified the consistency of a dataset using the root-mean-square

error from those duplicated grasps. The duplicated errors of the dataset $\mathcal{D}_a$, $\mathcal{D}_b$, and $\mathcal{D}_c$ were 0.0693, 0.0981, and 0.1059, respectively.

From the F1 score, the most consistent labels were the worst grasp and the best grasp. Most grasps belonging to those labels had unique characteristics that everyone could effortlessly decide their quality such as the grasps having contact points on the same edge, the grasps on the opposing parallel edges, and the grasps on the concave part of an object. On the other hand, the participants addressed the remaining three labels differently based on their criterion and preference. There were 540 grasps that all three participants chose the same label. There were 137 controversial grasps that at least one of them chose a negative label (worst or bad) and another one chose a positive label (good or best).

From those statistics, we conjectured that, for more than half of the possible grasps in the survey, the participants unable to consistently evaluate their quality from the position of contact points on the object surface alone. There were several useful feedbacks from the participants about this issue. They complained about some discrepancies between a grasp shown on the screen and the actual human's grasp that they are familiar. For examples, the former one lacks the sense of touch, gravitational direction, and the scale of the object's size relative to their hand. The difference in the number of dimensions and the limited perspective also influenced their indecisiveness. Furthermore, the participants were hesitant to assess the quality of grasps that they never performed or grasps on an imaginary object that they never seen before.

4.5.2 Neural network

The grasp planning method extracts all grasping features from the grasp's position and the object shape as described in Section 4.4 and uses them to predict the grasp quality via a multi-layered neural network. The first layer contains input nodes, i.e., all grasping features. The last layer is a single node that returns grasp quality in the range between zero and one, inclusive. We trained and tested several neural networks with different structures against datasets described in the previous section.

We implemented the neural networks with an open-source framework called Open Neural Networks (OpenNN) [112]. For simplicity, the neural networks only used hyperbolic tangent and linear function as the activation function and used a root-mean-square error as the loss function. For training strategy, we selected an evolutionary algorithm to initialize the weights of a neural network and chose a Quasi-Newton Method as the main training method. The training process ran until reaching the maximum iteration (50 generations for initialization and 250 iterations for the main training). For other parameters, we used the default values provided by the OpenNN framework.

4.5.3 Neural network's structure selection

We tested the neural networks with 14 different structures (two types of activation functions and seven different sizes) to find the suitable structure to learn the grasp quality from the introduced features. The first type ($\mathcal{N}_t$) used a hyperbolic tangent function as activation function for all node in the network while another type ($\mathcal{N}_m$) alternated the activation function in each layer between a linear function and a hyperbolic tangent function. The network's size defined the number of nodes and layers between the input layer and the output layer of a neural network. The first four sizes had the number of layers equal to the number of nodes in each layer: 1x1 ($\mathcal{N}_1$), 3x3 ($\mathcal{N}_3$), 5x5 ($\mathcal{N}_5$) and 11x11 ($\mathcal{N}_{11}$). Another 3 sizes had different number of node in each layer in the following arrangement: 5-3-2 ($\mathcal{N}_{5d}$), 10-8-6-4-2 ($\mathcal{N}_{10d}$) and 20-17-14-11-8-5-2 ($\mathcal{N}_{20d}$). So, the 14 structures in this experiment are $\mathcal{N}_{t1}$, $\mathcal{N}_{t3}$, $\mathcal{N}_{t5}$, $\mathcal{N}_{t11}$, $\mathcal{N}_{t5d}$, $\mathcal{N}_{t10d}$, $\mathcal{N}_{t20d}$, $\mathcal{N}_{m1}$, $\mathcal{N}_{m3}$, $\mathcal{N}_{m5}$, $\mathcal{N}_{m11}$, $\mathcal{N}_{m5d}$, $\mathcal{N}_{m10d}$, and $\mathcal{N}_{m20d}$. We use 10-fold cross-validation to find the most suitable structure with the minimal loss when predicting grasp quality in $\mathcal{D}_s$.

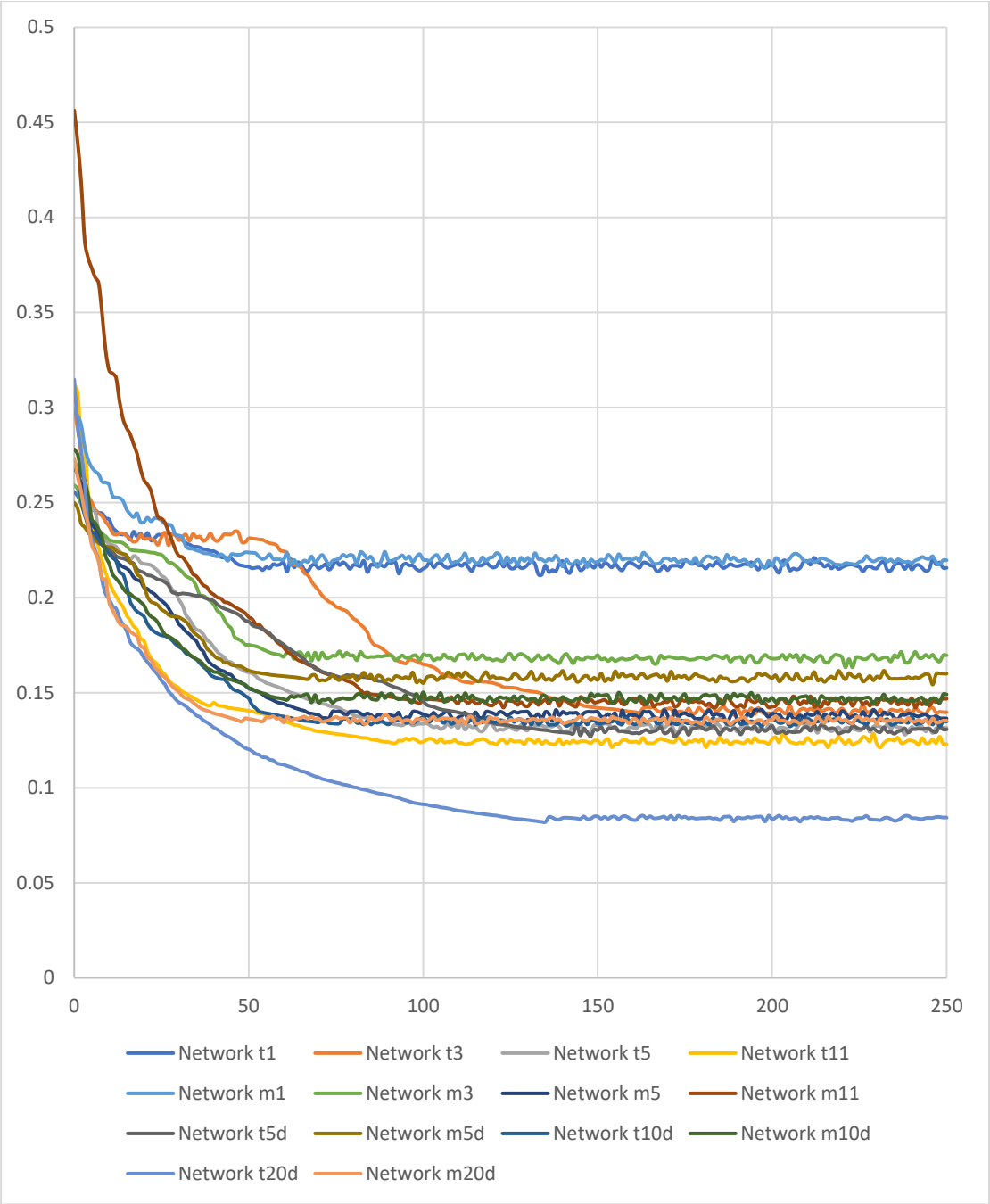


Figure 18 Loss rate of the neural networks with different structure when training with $\mathcal{D}_S$*. The X-axis is the number of iteration and the Y-axis is the root-mean-square error or loss value in each iteration. The loss rate was mostly stabilized after training for 100 iterations. The loss value of the best network's structure,* $\mathcal{N}_{t20d}$*, is 0.08435 at the end of the training.*

Table 6 The loss value and correlation from 10-fold cross-validation of the neural networks with different structure when training and testing with $\mathcal{D}_s$

Network	*Loss*	*Correlation*
$\mathcal{N}_{t1}$	0.217647	0.951281
$\mathcal{N}_{t3}$	0.152478	0.976421
$\mathcal{N}_{t5}$	0.139474	0.980331
$\mathcal{N}_{t11}$	0.106467	0.988574
$\mathcal{N}_{t5d}$	0.136908	0.980985
$\mathcal{N}_{t10d}$	0.107634	0.988267
$\mathcal{N}_{t20d}$	0.087651	0.992231
$\mathcal{N}_{m1}$	0.212847	0.953447
$\mathcal{N}_{m3}$	0.163008	0.972972
$\mathcal{N}_{m5}$	0.139576	0.980287
$\mathcal{N}_{m11}$	0.144535	0.978837
$\mathcal{N}_{m5d}$	0.162464	0.973152
$\mathcal{N}_{m10d}$	0.138001	0.980699
$\mathcal{N}_{m20d}$	0.105523	0.98876

Figure 18 shows the loss value of the neural networks with different structure during their training. Most network converged after training for 100 iterations. Their final loss value and training time were proportional to the network's size. The larger network usually took more training time but yielded lower loss value. Table 6 summarizes the average loss and correlation of 10-fold cross-validation for all network structures. The networks that use only hyperbolic tangent as activation function ($\mathcal{N}_t$) are slightly better than the network with alternated activation function ($\mathcal{N}_m$) for every network's size except the smallest one. With the highest correlation and the lowest loss value, the network $\mathcal{N}_{t20d}$ is the best structure to predict grasps in $\mathcal{D}_s$. The worst ones are $\mathcal{N}_{t1}$ and $\mathcal{N}_{m1}$ which produce comparable high loss value. So, we choose $\mathcal{N}_{t20d}$ as the network's structure to create all neural networks in the following experiments.

Table 7 The loss value and correlation from 10-fold cross-validation of the neural network $\mathcal{N}_{t20d}$ when training and testing with difference dataset

Dataset	*Loss*	*Correlation*
$\mathcal{D}_s$	0.094403	0.991011
$\mathcal{D}_v$	0.515416	0.500545
$\mathcal{D}_a$	0.338635	0.870762
$\mathcal{D}_b$	0.288942	0.903661
$\mathcal{D}_c$	0.353308	0.871573
$\mathcal{D}_{ab}$	0.314196	0.883716
$\mathcal{D}_{ac}$	0.334485	0.876039
$\mathcal{D}_{bc}$	0.319692	0.885351
$\mathcal{D}_{abc}$	0.324606	0.879583

4.5.4 Result and analysis of grasp quality prediction

We trained and tested the neural network $\mathcal{N}_{t20d}$ with different datasets. Table 7 shows the loss value and correlation from 10-fold cross-validation. The dataset that easiest to predict was the synthetic dataset $\mathcal{D}_s$ while the most unpredictable one was the volunteer dataset $\mathcal{D}_v$. Both are the bad examples of poor distribution in a dataset. The grasps in synthetic dataset focused on only five simple objects. A neural network could learn some reasonable grasps on those objects but failed to cope with the objects with a more complex shape. On the other hand, the volunteer dataset only contained force-closure grasps. A neural network was incapable of learning the quality of the non-force-closure grasps from $\mathcal{D}_v$.

To ensure that the neural networks could learn and predict the grasp quality from the survey data, we conducted another survey to collect the data from the same participants. Then, we compared the label between the newly collected data ($\mathcal{D}_a^2$, $\mathcal{D}_b^2$, and, $\mathcal{D}_c^2$) and the ones evaluated by the neural networks trained with the old data ($\mathcal{P}_a$, $\mathcal{P}_b$, and, $\mathcal{P}_c$) as shown in Table 8, Table 9, and, Table 10. The overall result was acceptable. The worst and best labels were the most predictable ones. More than

half of wrong labels were adjacent to the right one (e.g., predict the best grasp, but it is a good grasp). The accuracy of $\mathcal{P}_b$ and $\mathcal{P}_c$ was consistent with the result from 10-fold cross-validation. However, there was significant contrast between the labels in $\mathcal{P}_a$ and $\mathcal{D}_a^2$. The neural network mistakenly identified most good grasps as the best grasps and failed to differentiate between mediocre and good grasps. From the post-survey interview, we found that, in $\mathcal{D}_a^2$, the participant hesitated to judge the quality of stretching grasps and the expectation of the best grasps in the second survey is higher than the first one.

The neural networks trained by those datasets evaluated all possible grasps on the 13 known objects and the ten unknown objects. We visualized the best 20 grasps with the highest quality according to their estimation in Figure 19 - Figure 40. The known objects were used in the survey program and the training of the neural networks while the unknown objects were the ones that the neural networks never perceived before.

We grouped and qualitatively analyzed the best grasps on the test objects based on their shapes: simple objects $\mathcal{O}_s$ (circle, diamond, and, square), complex objects $\mathcal{O}_p$ (hShape, ring, and, sShape), daily objects $\mathcal{O}_d$ (bottle, driller, and, glass), concave objects $\mathcal{O}_c$ (hConcave, hConcaveSmall, simpleConcave, and, vConcaveSmall). The best grasps usually gathered into some small clusters on the object surface. This behavior indicates that a neural network should learn some tacit knowledge or preferences of choosing the desirable grasps from a dataset. The best grasps from the combining datasets generally shared similarity with the original ones. They should represent the common conception of the grasp quality which participants agreed.

Table 8 The confusion matrix and F1 score between human and predicted labels

$\mathcal{D}_a^2$ \ $\mathcal{P}_a$	*Worst*	*Bad*	*Mediocre*	*Good*	*Best*	*F1 score*
Worst	182	51	19	15	3	*0.6868*
Bad	50	120	54	25	10	*0.4624*
Mediocre	23	57	124	114	42	*0.4000*
Good	5	32	63	104	172	*0.3270*
Best	0	0	0	2	33	*0.2237*
Root-mean-square error			*0.2636*		*Avg. F1*	*0.4200*

Table 9 The confusion matrix and F1 score between human and predicted labels

$\mathcal{D}_b^2$ \ $\mathcal{P}_b$	*Worst*	*Bad*	*Mediocre*	*Good*	*Best*	*F1 score*
Worst	194	27	12	1	0	*0.7854*
Bad	64	185	48	27	16	*0.6167*
Mediocre	2	43	115	44	3	*0.4925*
Good	0	5	77	151	104	*0.5059*
Best	0	0	8	37	137	*0.6199*
Root-mean-square error			*0.1991*		*Avg. F1*	*0.6041*

Table 10 The confusion matrix and F1 score between human and predicted labels

$\mathcal{D}_c^2$ \ $\mathcal{P}_c$	*Worst*	*Bad*	*Mediocre*	*Good*	*Best*	*F1 score*
Worst	178	34	15	8	5	*0.7120*
Bad	48	121	76	46	30	*0.4165*
Mediocre	17	80	113	50	17	*0.4209*
Good	14	18	50	120	69	*0.4520*
Best	3	7	6	36	139	*0.6164*
Root-mean-square error			*0.2754*		*Avg. F1*	*0.5236*

The objects in the group $\mathcal{O}_s$ were all convex and uncomplicated. The predicted grasps were force-closure and reasonable except the ones from the volunteer dataset which nominated the non-force-closure grasps on the square and the diamond. Those non-force-closure grasps were mistakenly chosen due to the lack of those type of grasps in the dataset $\mathcal{D}_v$.

The objects in $\mathcal{O}_p$ had a lot of concave parts and parallel edges. From the analytical perspective, they had both squeezing and stretching cages and force-closure grasps. Due to the diversity of feasible grasps, the best grasps from each dataset were mostly unique. This trait was especially true for the ring which had the best grasps scattered all over its surface. Some datasets biased toward squeezing grasps ($\mathcal{D}_s$, $\mathcal{D}_b$, and, $\mathcal{D}_c$) while some other datasets biased toward stretching grasps ($\mathcal{D}_v$ and $\mathcal{D}_a$). All of them were reasonable and justified.

The objects in $\mathcal{O}_d$ represented some daily items that have the easy-to-grasp part. Most humans instinctively hold those objects at their handle (i.e., driller's grip, glass's stem, and middle part of a bottle). Most neural networks failed to recognize the graspable part on the objects. We believed that the reasons for failure were 1) lack of relevant samples in the datasets 2) the grasping features lacked necessary information for a neural network to learn about those parts. However, all neural networks managed to find reasonable grasps except the ones trained with bad datasets, $\mathcal{D}_s$ and $\mathcal{D}_v$.

The last group, $\mathcal{O}_c$, emphasized simple objects with concave sections. We expected that the neural networks should learn to grasp those objects on the concave parts since it has an ability to cage an object. It was work well for the simpleConvex. However, for the other three objects, most neural networks inclined toward the grasps on parallel edges. The ones that nominated the caging grasps are $\mathcal{D}_s$, $\mathcal{D}_c$, and its variants ($\mathcal{D}_{ac}$, $\mathcal{D}_{bc}$, and, $\mathcal{D}_{abc}$). This evidence indirectly suggested that two out of three participants preferred the grasps on opposing edges than the grasps on the concave parts of an object.

The best grasps on the unknown objects were diverse and inconclusive, but most of them were reasonably good considering the underlying problem of the

object's shape. We suspected that the chaotic results were caused by the representation of an object, a polygon. It failed to express the smooth surface of an object accurately and affected the grasping features that relied on the contact normal. The uneven surface, e.g., zigzag lines, replaced the smooth surface of an object during the conversion from an image of object's silhouette to a polygon using the "findContours" function in the OpenCV library. The neural networks mistakenly predicted the quality of grasps on the uneven surface. This problem profoundly affected the curvy objects such as bird, fish, guitar, and, teddy bear.

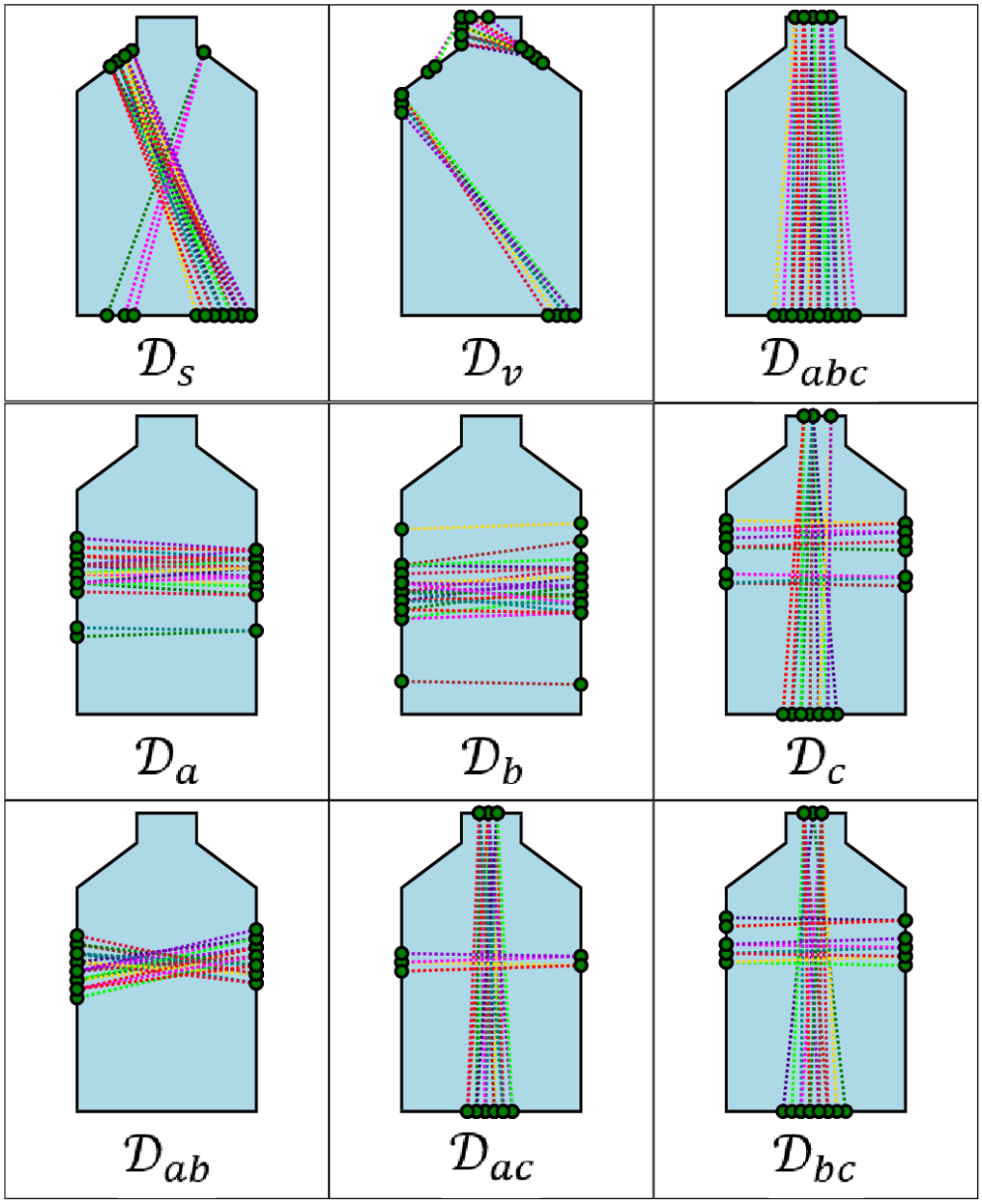


Figure 19 The best 20 grasps according to the neural networks trained by different datasets.

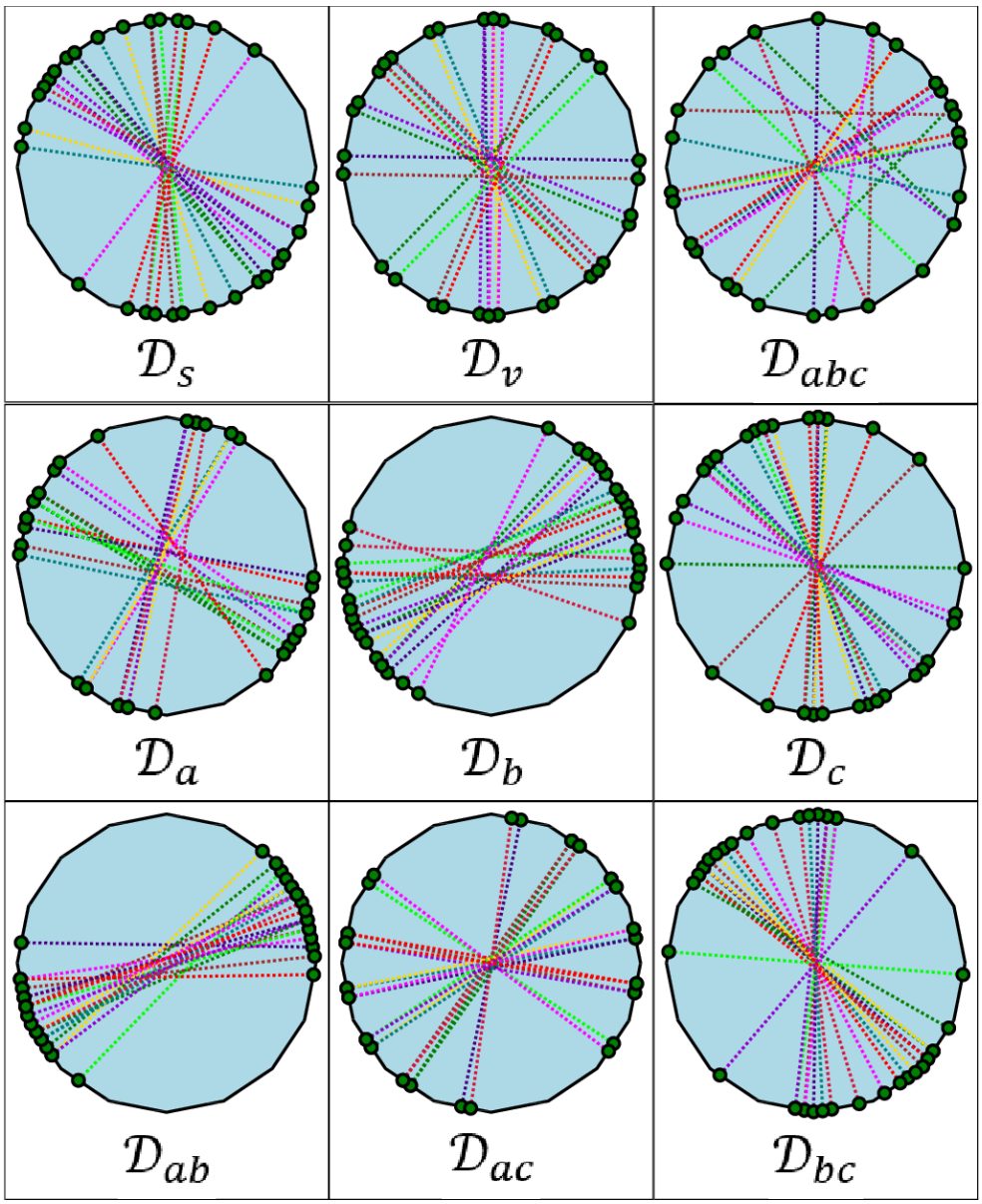


Figure 20 The best 20 grasps according to the neural networks trained by different datasets.

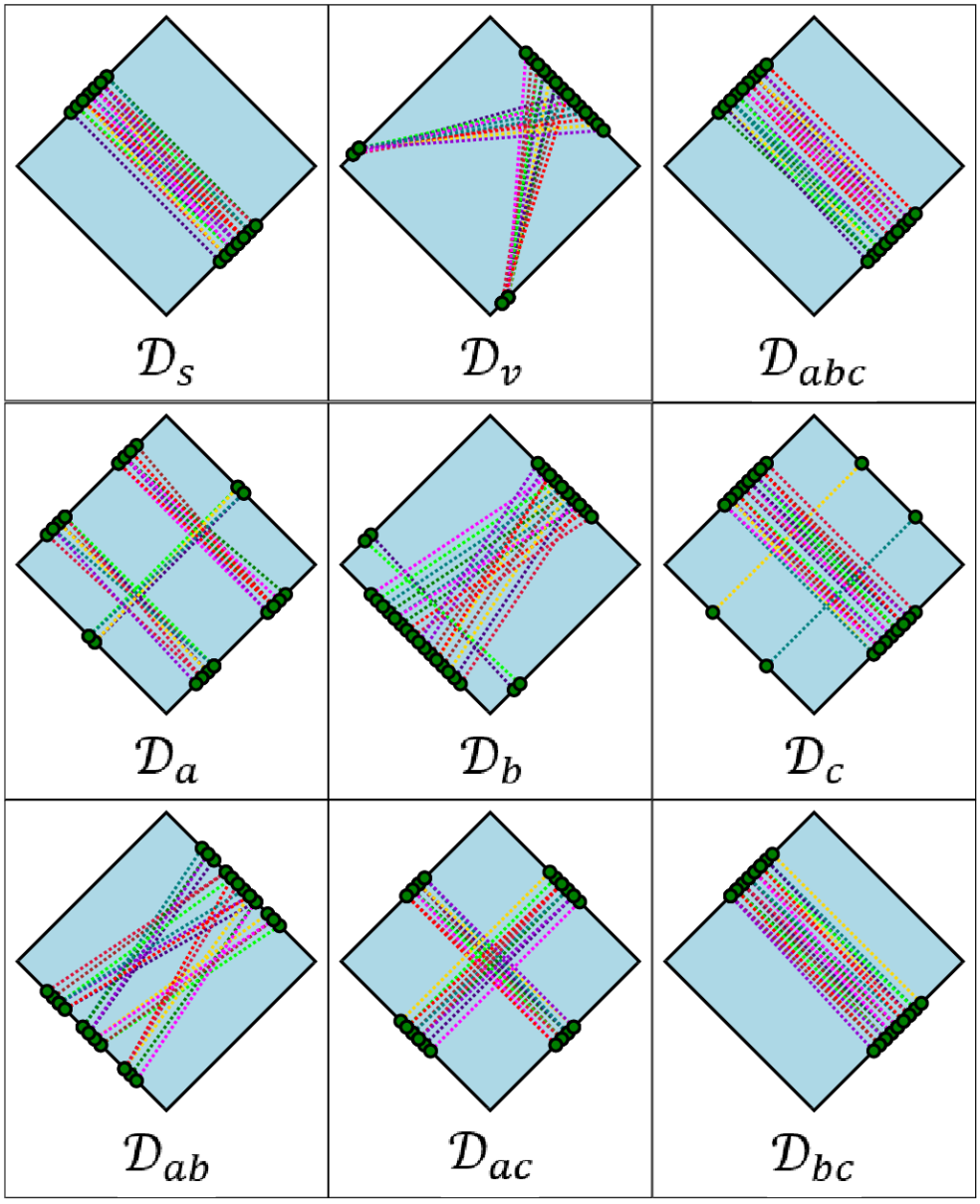


Figure 21 The best 20 grasps according to the neural networks trained by different datasets.

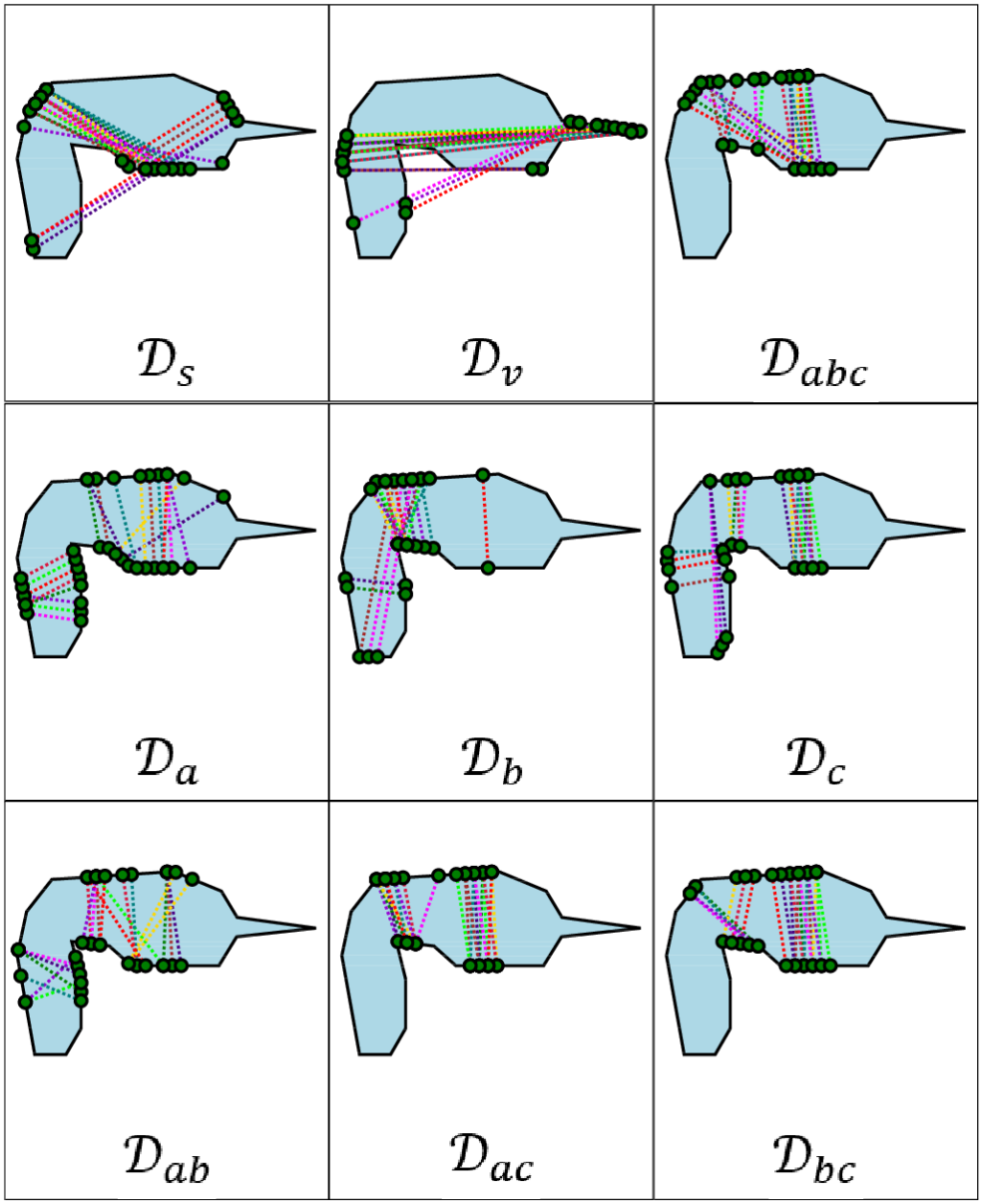


Figure 22 The best 20 grasps according to the neural networks trained by different datasets.

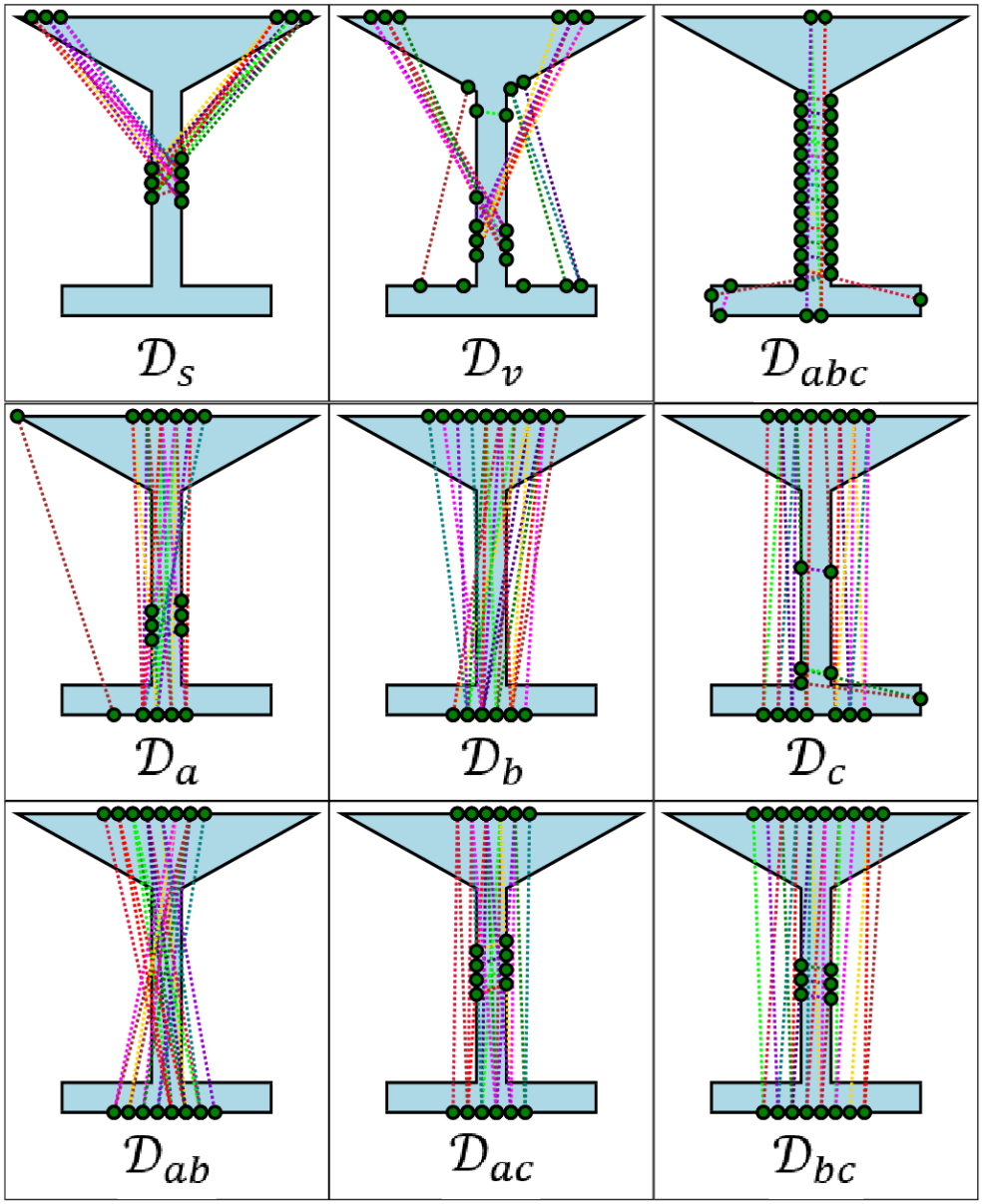


Figure 23 The best 20 grasps according to the neural networks trained by different datasets.

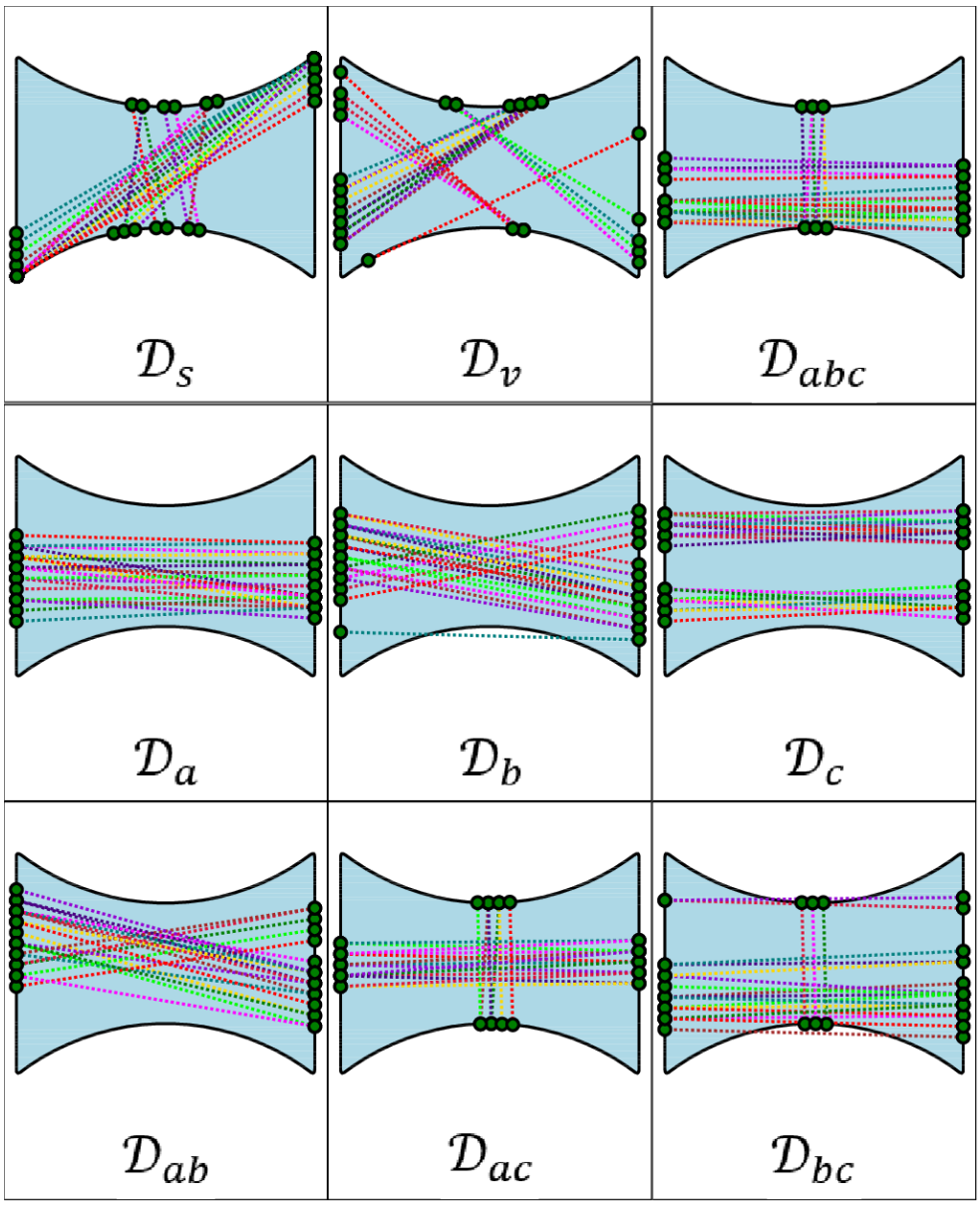


Figure 24 The best 20 grasps according to the neural networks trained by different datasets.

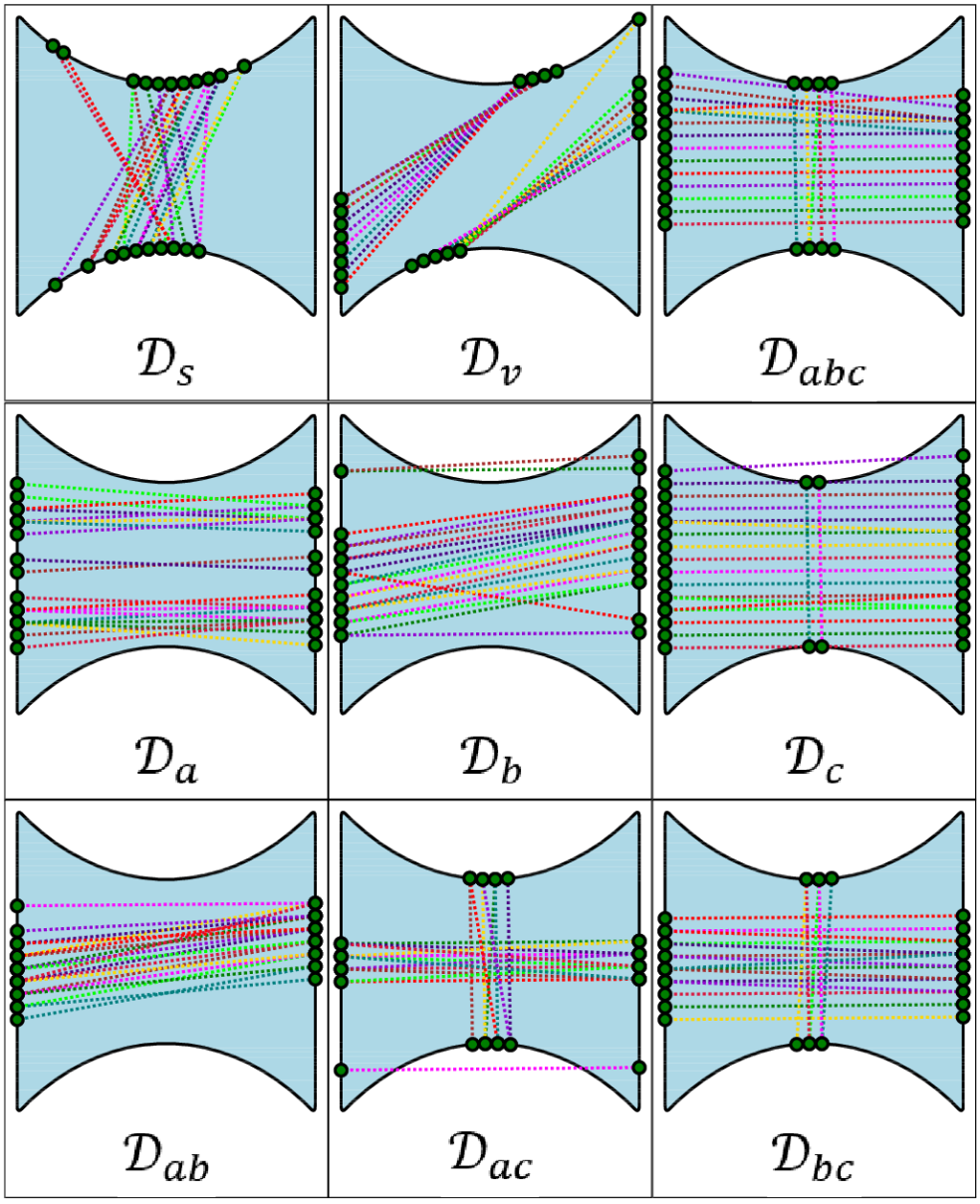


Figure 25 The best 20 grasps according to the neural networks trained by different datasets.

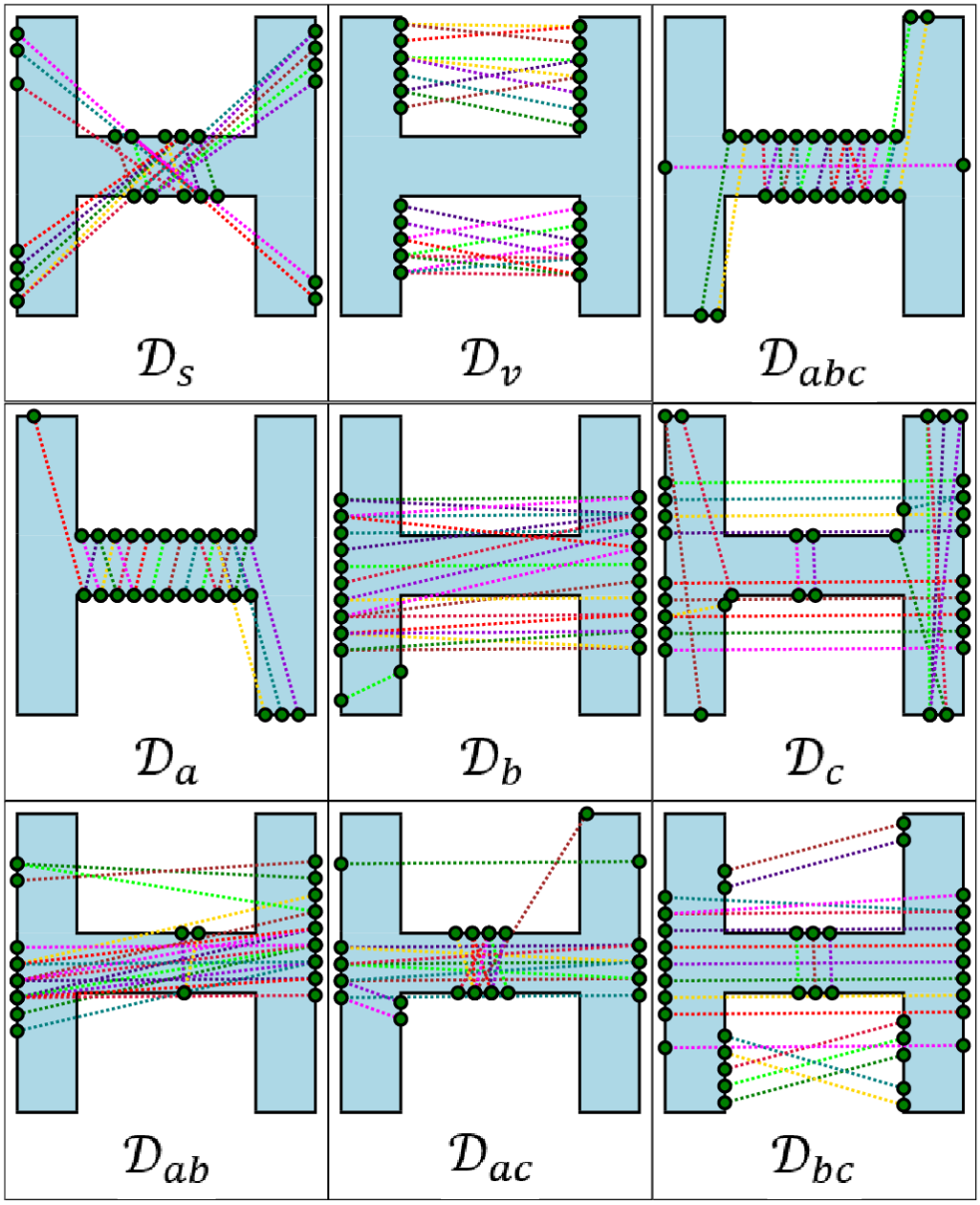


Figure 26 The best 20 grasps according to the neural networks trained by different datasets.

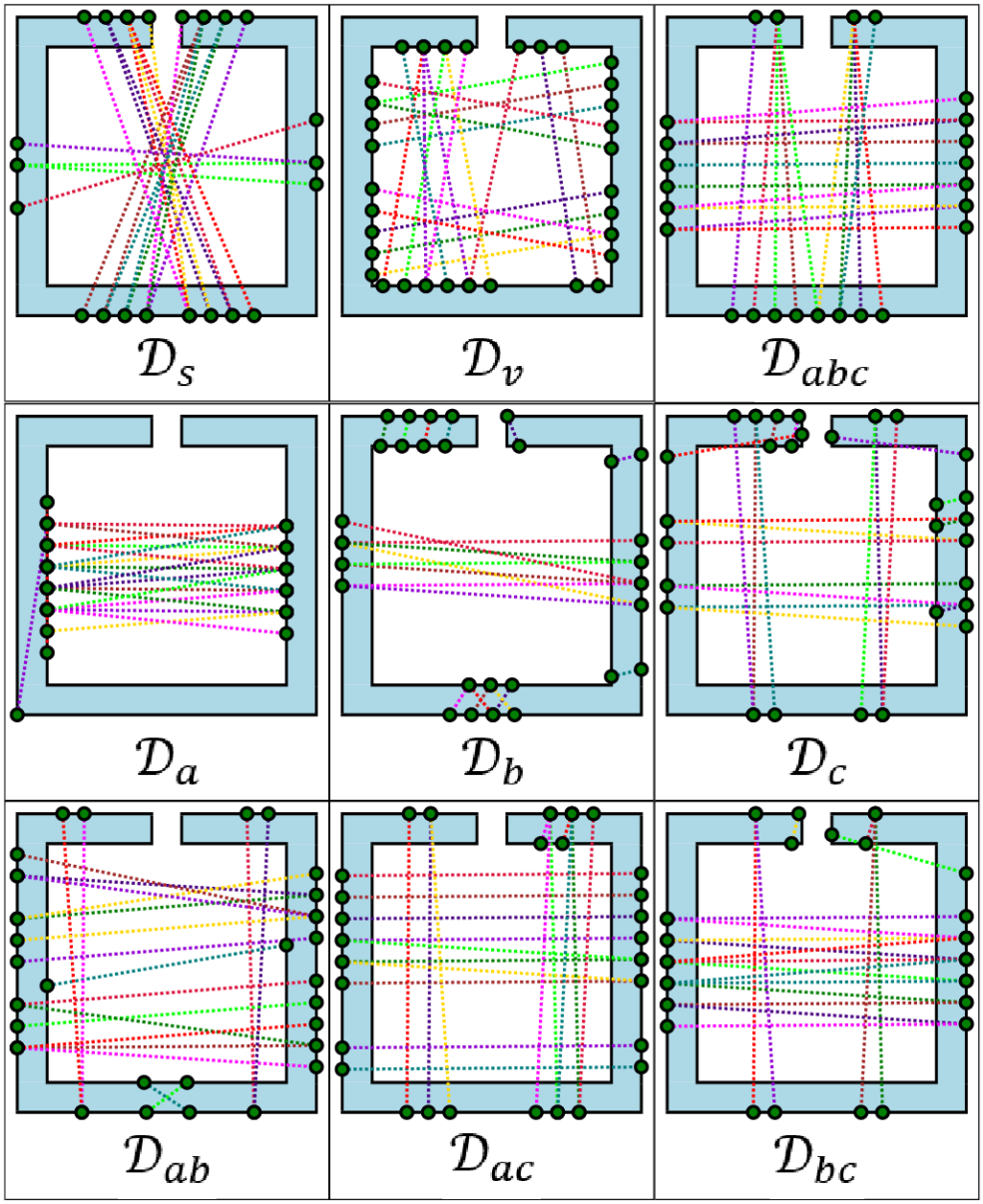


Figure 27 The best 20 grasps according to the neural networks trained by different datasets.

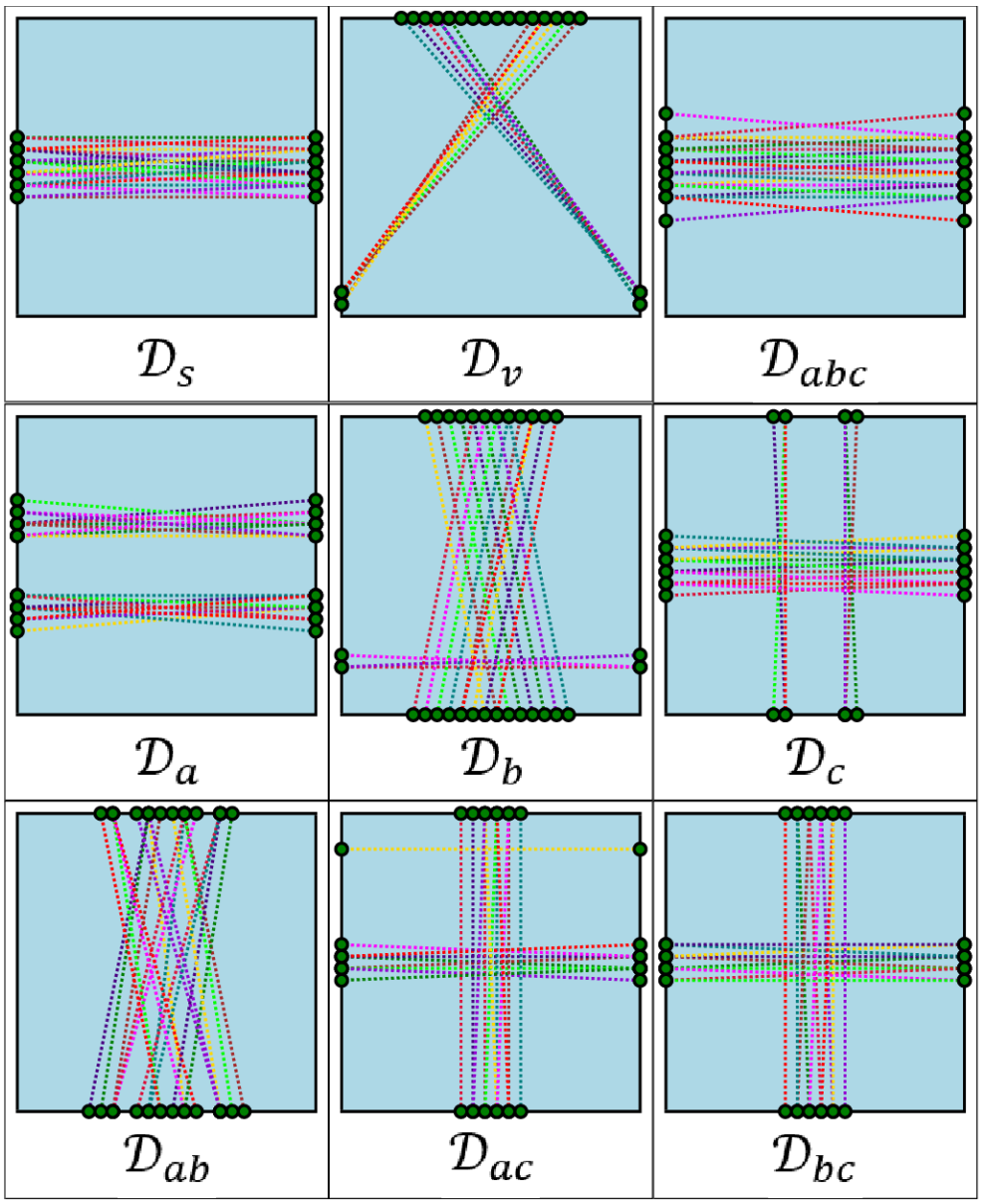


Figure 28 The best 20 grasps according to the neural networks trained by different datasets.

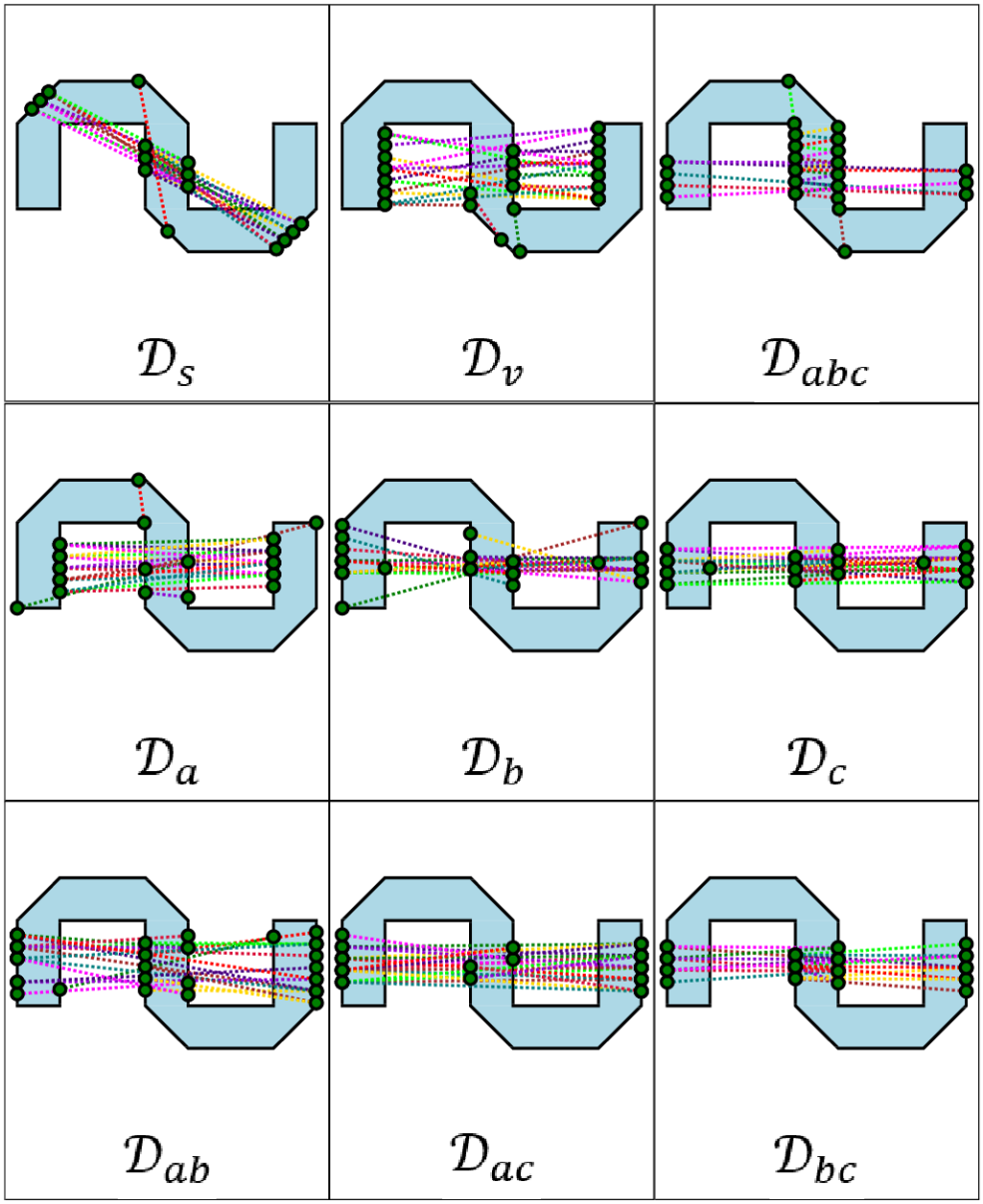


Figure 29 The best 20 grasps according to the neural networks trained by different datasets.

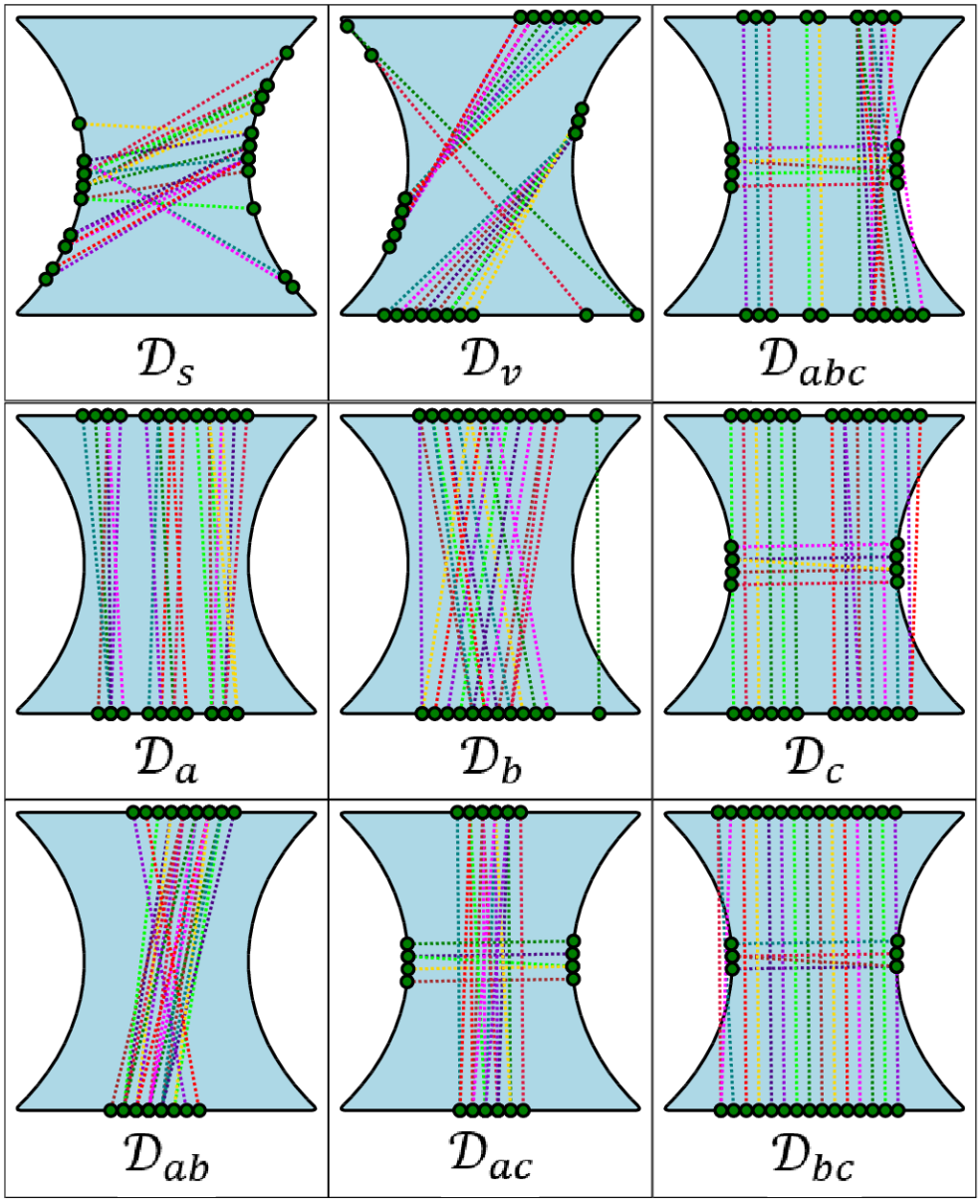


Figure 30 The best 20 grasps according to the neural networks trained by different datasets.

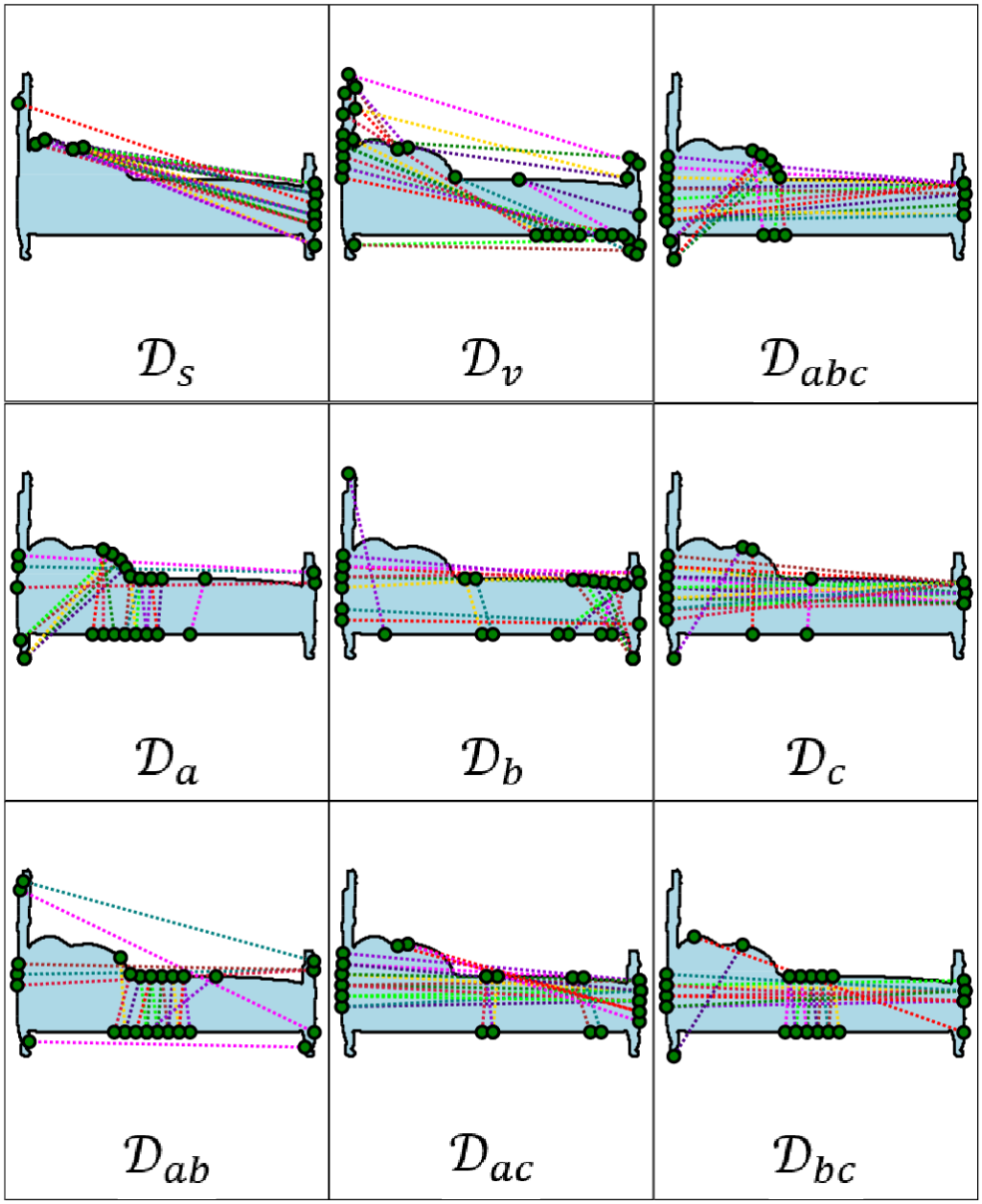


Figure 31 The best 20 grasps according to the neural networks trained by different datasets.

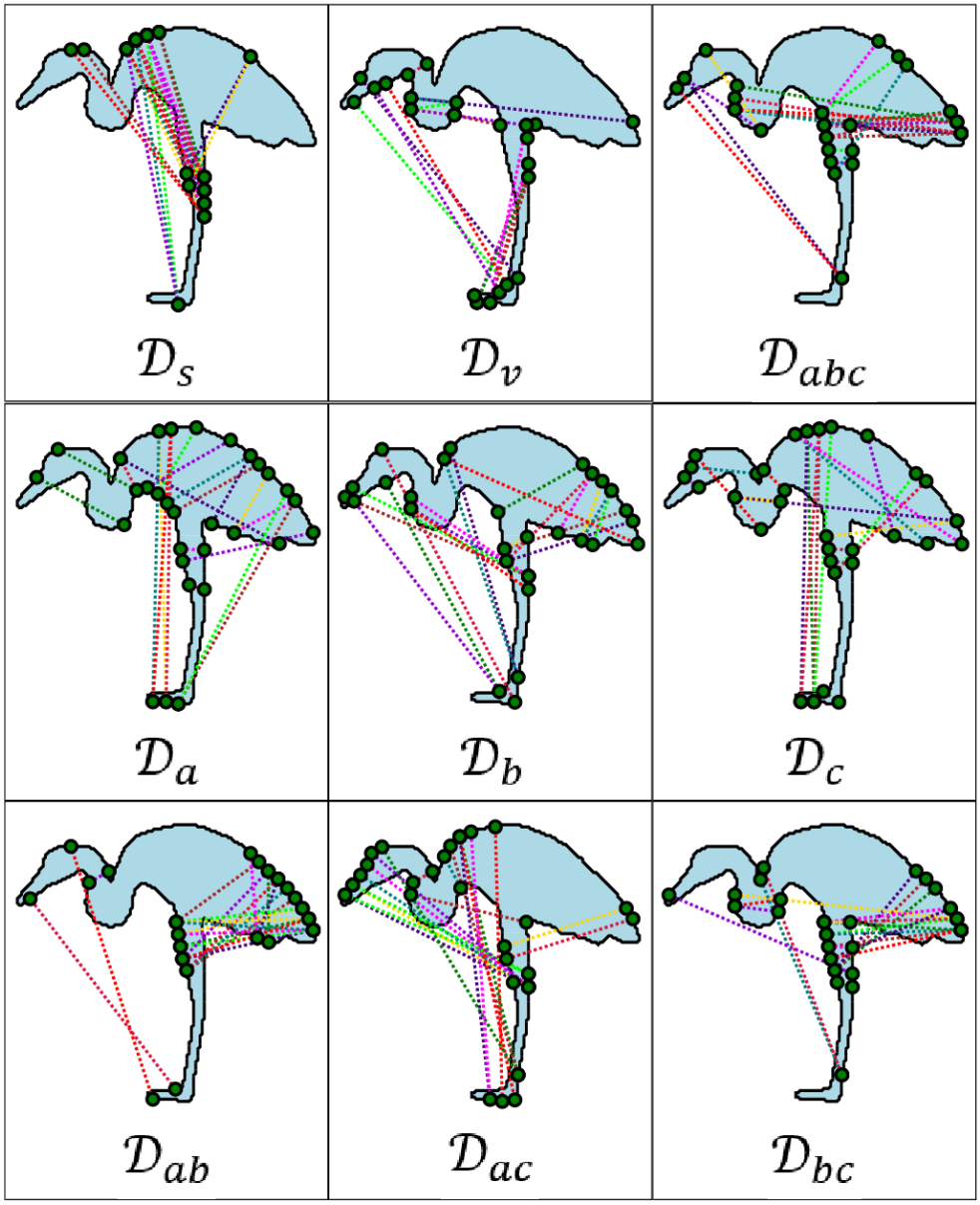


Figure 32 The best 20 grasps according to the neural networks trained by different datasets.

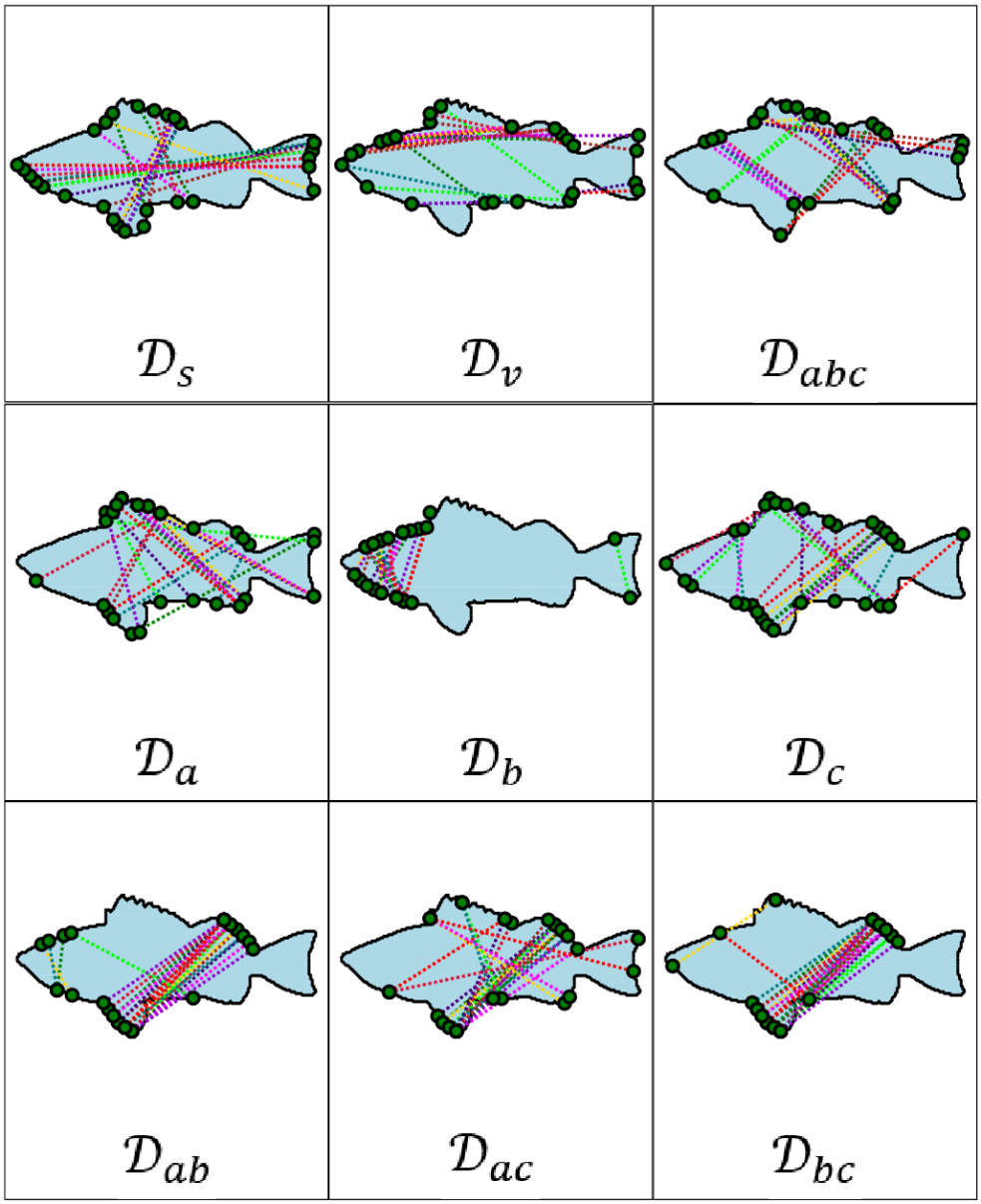


Figure 33 The best 20 grasps according to the neural networks trained by different datasets.

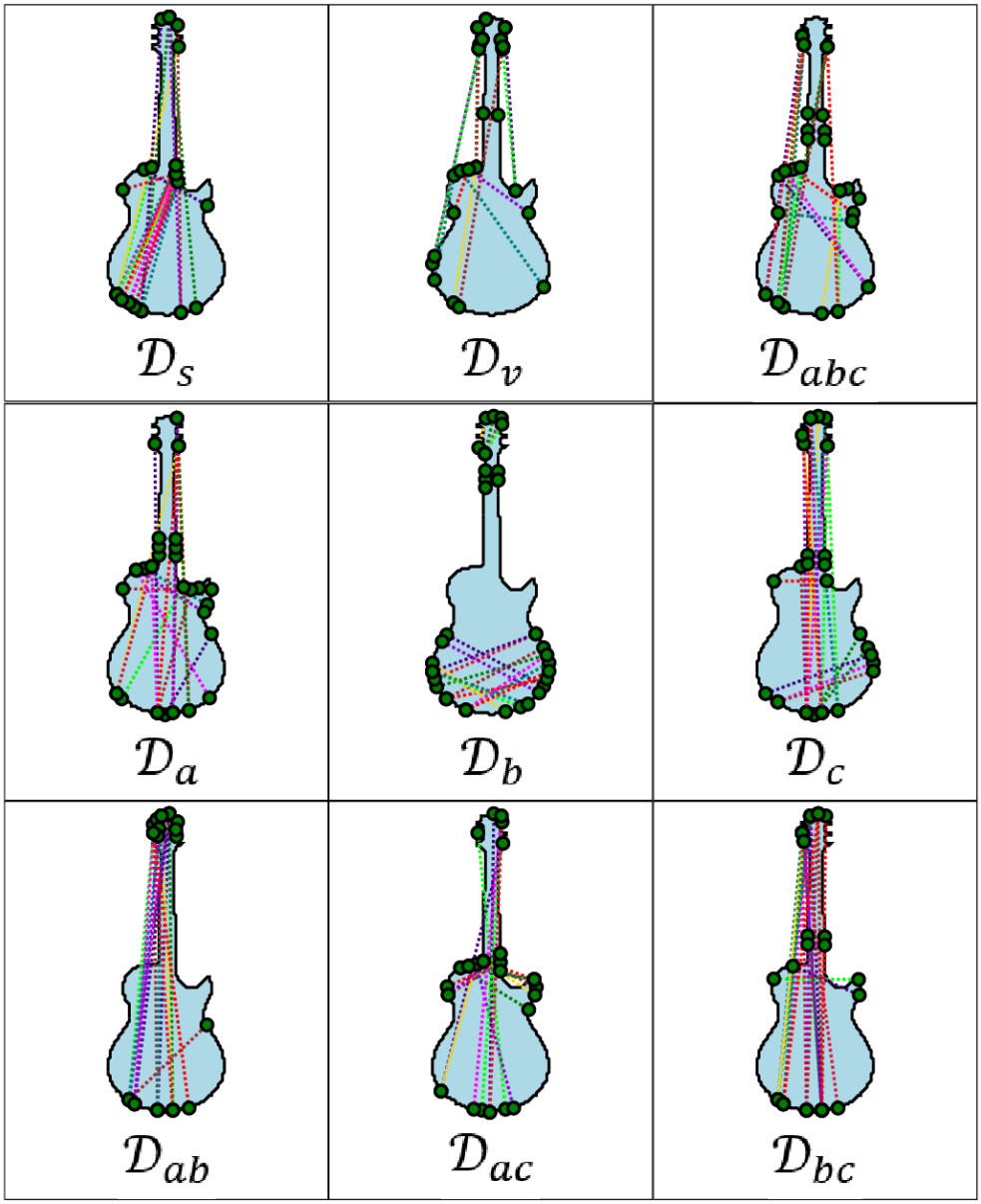


Figure 34 The best 20 grasps according to the neural networks trained by different datasets.

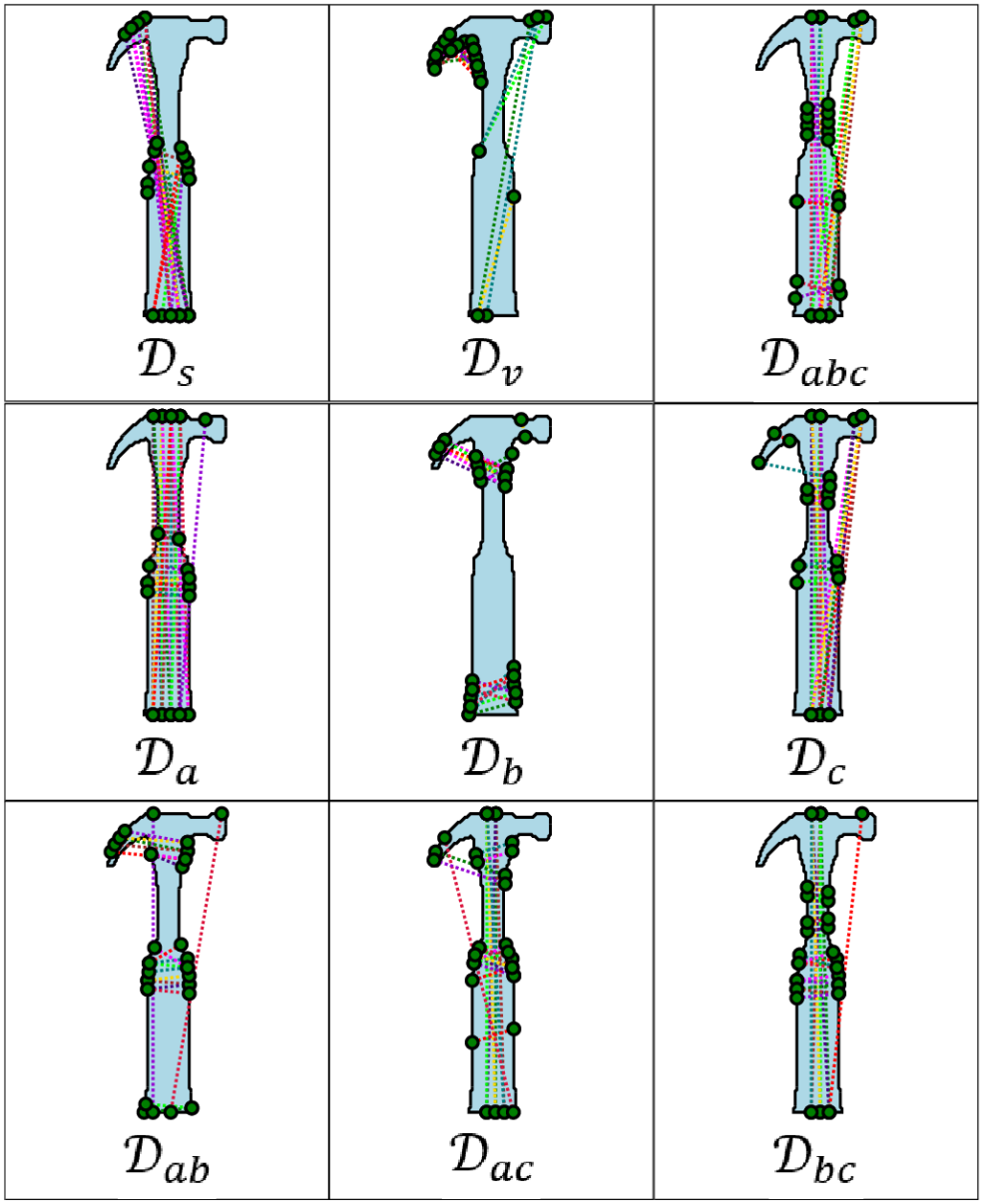


Figure 35 The best 20 grasps according to the neural networks trained by different datasets.

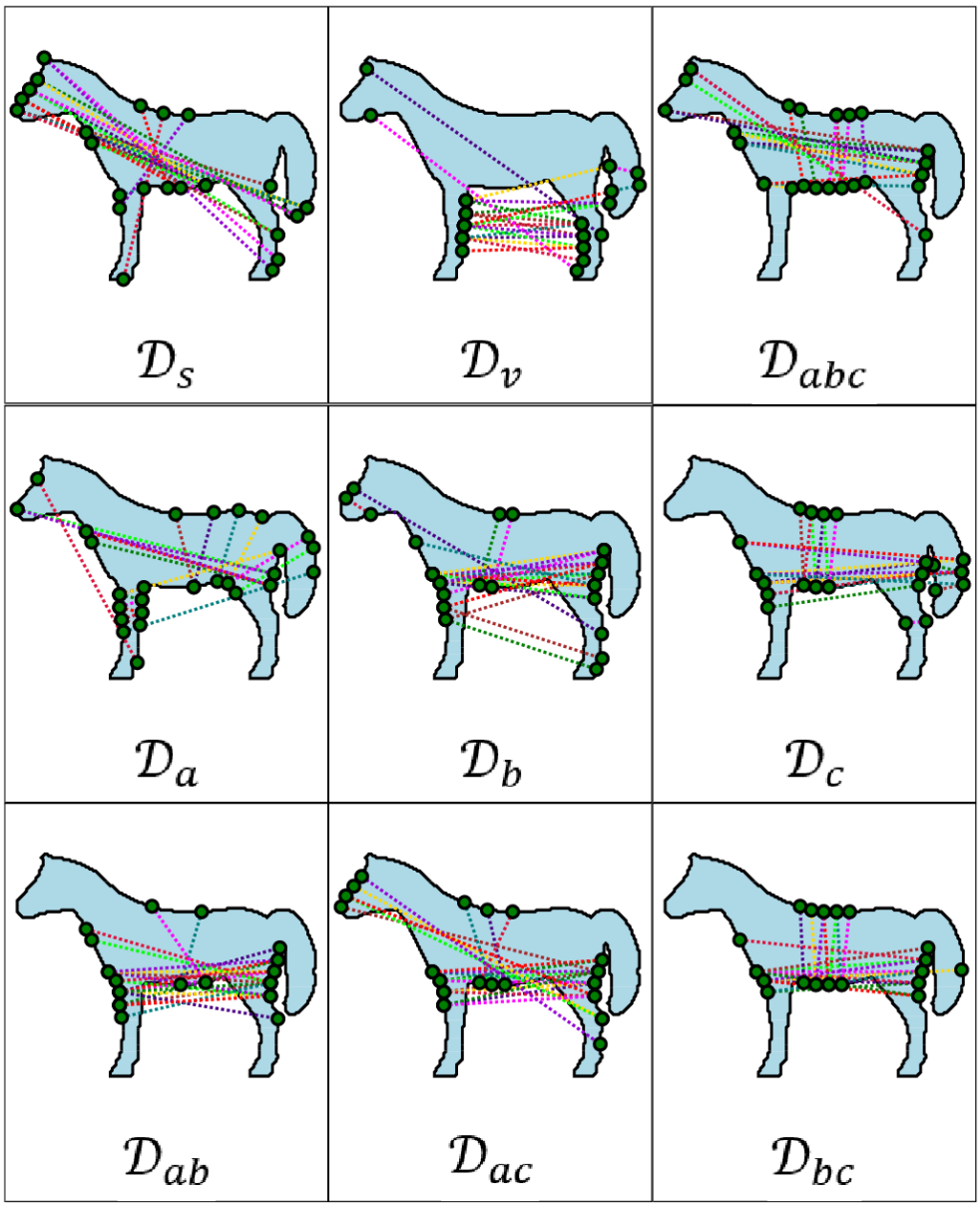


Figure 36 The best 20 grasps according to the neural networks trained by different datasets.

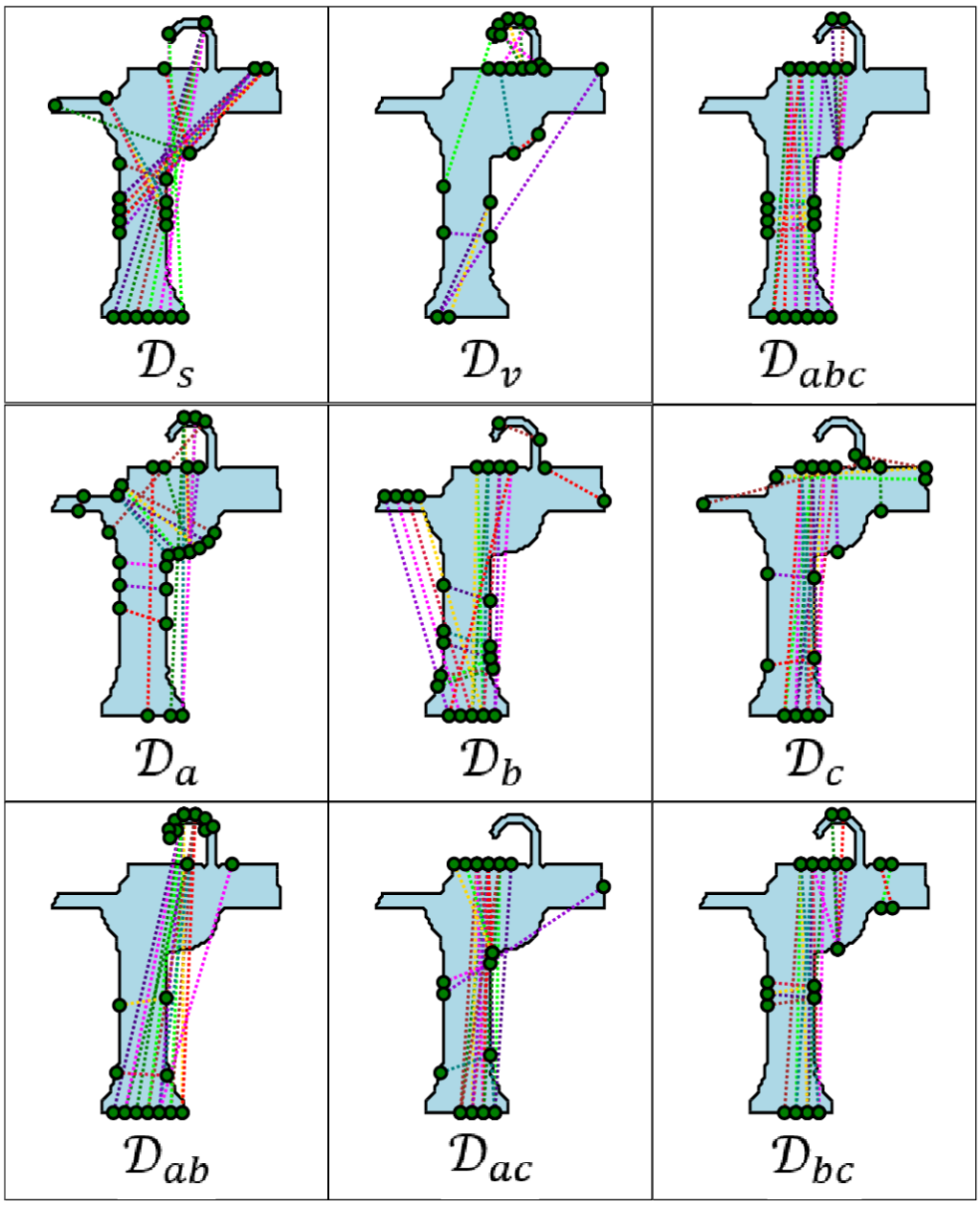


Figure 37 The best 20 grasps according to the neural networks trained by different datasets.

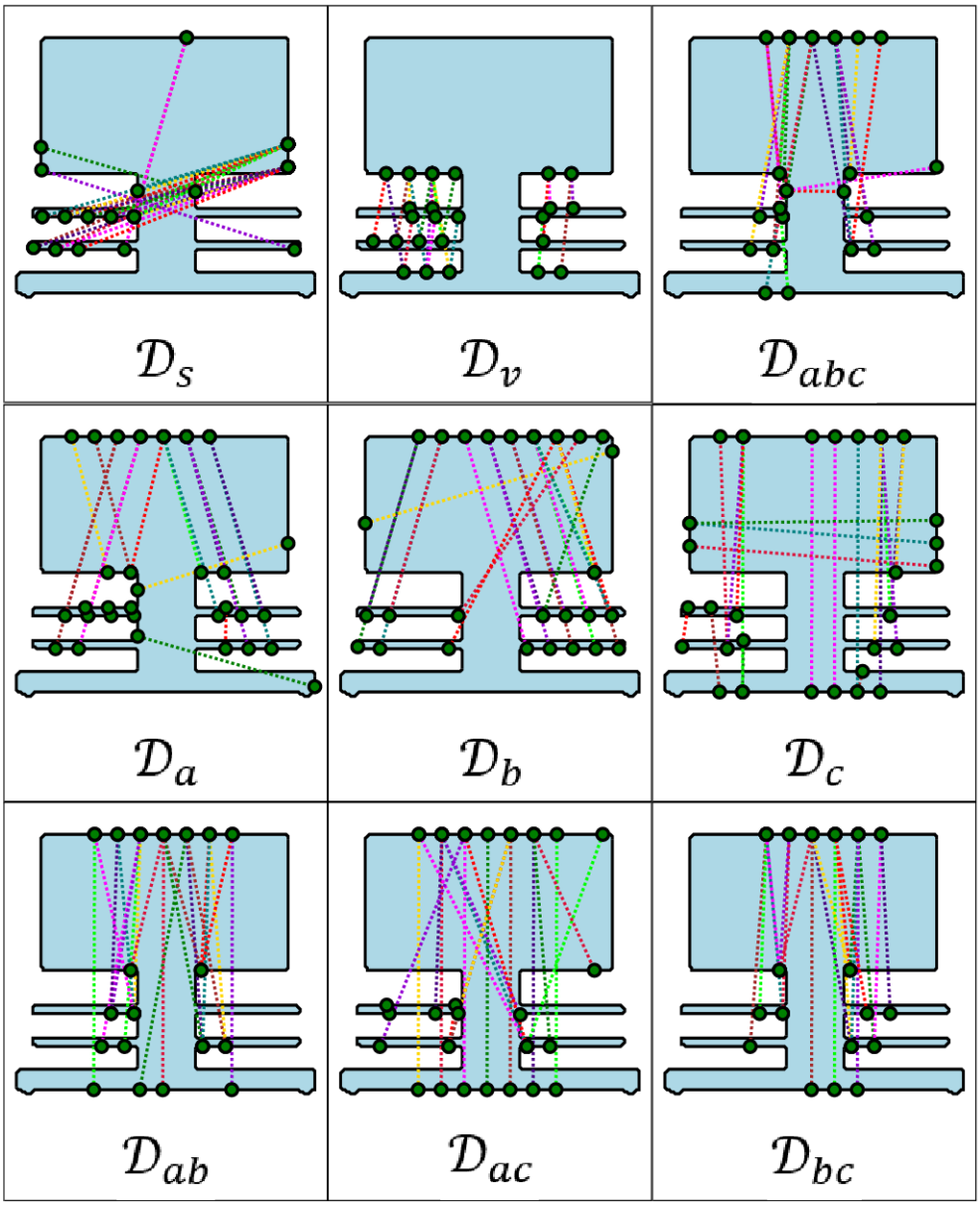


Figure 38 The best 20 grasps according to the neural networks trained by different datasets.

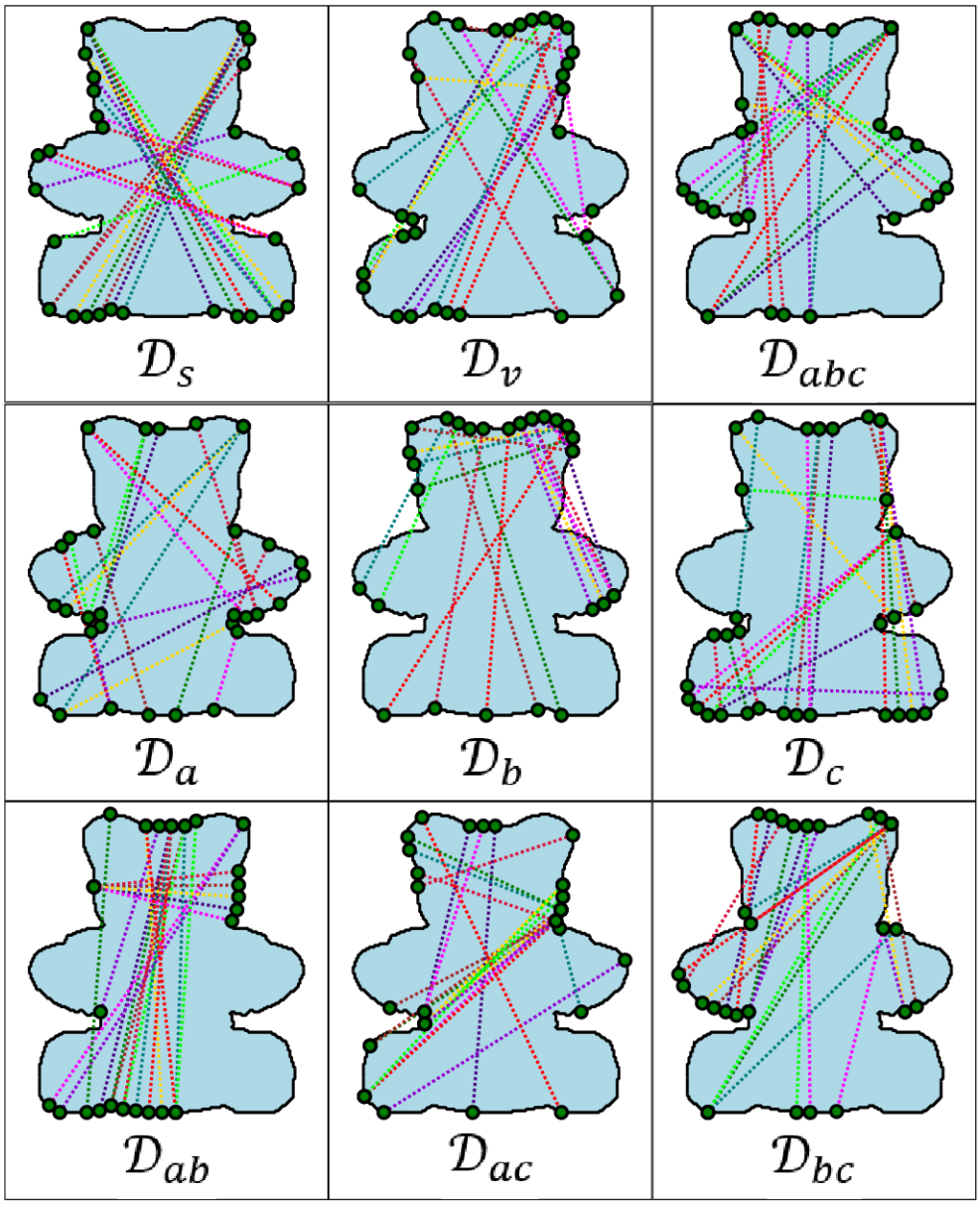


Figure 39 The best 20 grasps according to the neural networks trained by different datasets.

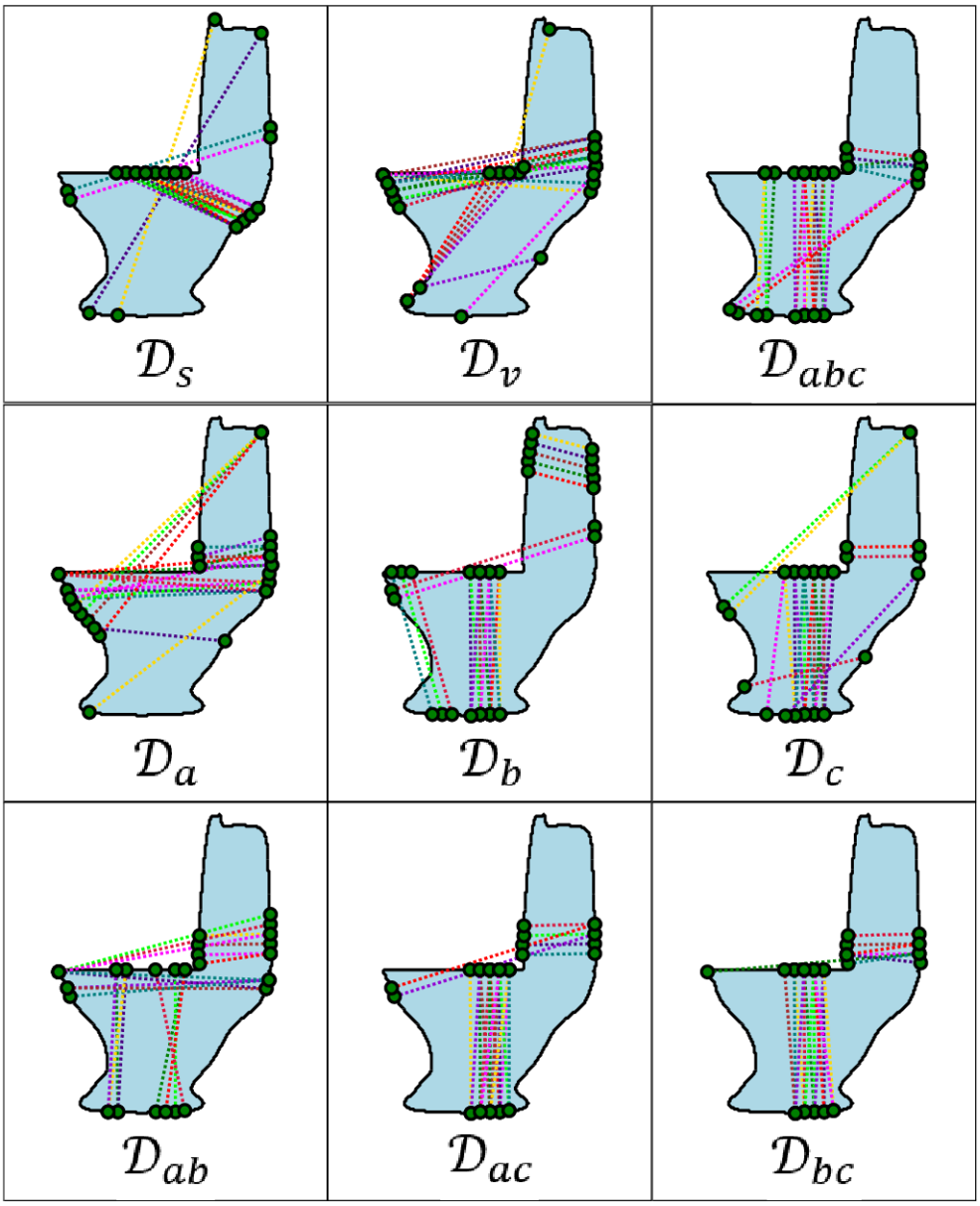


Figure 40 The best 20 grasps according to the neural networks trained by different datasets.

4.5.5 The contribution of the grasping feature

We estimated the contribution of each grasping feature introduced in Section 4.4 by removing one feature at a time from neural networks, trained them and performed 10-fold cross-validation, then, compared its loss against the original one. Table 11 shows the loss comparison obtained from the neural networks trained with different datasets. We believe that the difference in the loss values between the original neural network and the one trained without a specific feature should indirectly indicate how much influence of that feature affect the capability of the neural networks. The positive value indicates that the accuracy of a neural network without the specific feature is worse than the original network. Therefore, that particular feature is essential for the evaluation of the grasp quality. Conversely, the negative value means a neural network has a better performance without the specific feature, and that feature misleads the prediction of grasp quality in the neural network. If the difference is insignificant, it indirectly indicates that the particular feature is irrelevant to the grasp quality. This investigation also told us the general concept of the participant's preference when they assessed the grasp quality in the survey program.

Unsurprisingly, the most notable features of a neural network trained from synthetic dataset were the distance-to-center (f_{oc}), the heuristic-force-closure (f_{hfc}), the linear mobility (f_{lm}) and, the density of ICRs (f_{id}) which influenced predicted grasp's scores positively. The heuristic methods in Appendix D indirectly utilized those grasping features to generate the grasps in $\mathcal{D}_s$. There was no remarkable grasping feature in $\mathcal{D}_v$ due to its inconsistency as discussed in the Section 4.5.1. According to Table 11, All participants had diverse preferences when evaluating the grasp quality. The first one, $\mathcal{D}_a$, focused on f_{lm}, the ability to cage an object. The second one, $\mathcal{D}_b$, preferred f_{is}, f_{hfc} and f_{lm}; the size of ICRs as the first criteria and the heuristic of the force-closure condition and caging as secondary choices. The last one, $\mathcal{D}_c$, leaned toward f_{lm}, f_c, and, f_{fc}; the caging ability and the force-closure condition while also considered the curvature of contact points. In general, the most crucial grasping feature is linear mobility (f_{lm}) which measures how many directions an object can freely move without colliding with the grasp and directly relates with an ability of

a grasp to cage an object. Lastly, the force-closure feature (f_{fc}) had the most negative impact on the accuracy of the learning model.

Table 11 The feature contribution based on the neural networks trained from different datasets. The value is a percentage difference of the loss values between the original neural network and a neural network which removed a specific feature.

	f_c	f_a	f_{fc}	f_{lm}	f_{hfc}	f_{is}	f_{id}	f_{oc}
$\mathcal{D}_s$	12.79	21.83	12.16	30.71	35.28	8.34	34.44	60.85
$\mathcal{D}_v$	-3.54	-0.62	-3.33	2.71	-1.01	0.20	-1.87	0.13
$\mathcal{D}_a$	-5.25	-0.46	-8.71	11.60	-10.18	-6.55	-5.84	-4.28
$\mathcal{D}_b$	2.47	1.96	-4.45	3.93	4.07	4.81	1.38	-3.44
$\mathcal{D}_c$	7.17	4.65	5.98	7.73	0.51	2.44	1.75	0.40
$\mathcal{D}_{ab}$	0.09	2.15	-0.95	5.50	0.72	-0.52	-0.60	1.40
$\mathcal{D}_{ac}$	-2.60	-0.24	0.35	8.77	1.13	1.00	1.35	1.31
$\mathcal{D}_{bc}$	-1.51	-0.35	-7.53	3.28	-1.00	-3.55	-2.69	-4.11
$\mathcal{D}_{abc}$	-1.80	-0.63	-3.48	4.80	-2.00	-0.08	0.79	-2.27

4.6 Conclusion

We investigated an empirical approach to predict the grasp quality based on grasping features from both analytical and heuristic method. The samples of the grasp quality were collected from the human survey and synthesized from predefined heuristics. We explored a suitable structure of neural networks for the task and the contribution of the introduced features toward the accuracy of the neural networks.

The results of the experiment indicated the feasibility and flexibility of proposed method. It can recognize the desirable grasps on various objects based on different training data. We analyzed the preferences of the individuals when they evaluate the grasp quality from the contribution of the grasping features. Lastly, we found some untenable evidence on the limitation of evaluating the grasp quality based on the force analysis and the contact points.

Chapter 5 Future of Grasping

This chapter is dedicated to our speculation on the current state of robotic grasping and its future direction. There is no experiment nor solid proof to support our claims, just purely opinions and suggestions on the topic. To sum up this chapter in two sentences: Dexterous grasping based on force analysis did not work well on a real robot, and the next breakthrough of robotic grasping should be one of those studies: re-definition of robotic grasping, soft-body manipulator, haptic sensing and learning-based grasp planning method.

With current technology, it is still impossible to predict or simulate a seemingly simple, yet complex physical interaction such as grasping an object with 100% accuracy within limited time and resources. The precise prediction requires an insurmountable amount of information and complete control of the environment to prevent unaccountable factors that might interfere with the outcome. Such requirements are impractical and overwhelmingly hard to accomplish in a real robot operated in the human-friendly environment. Furthermore, It is questionable that any grasp planning method can objectively classify a grasp as good or bad based on the number of fingers and object's shape. Furthermore, a grasp might instantly change from success to failure the moment when a robot failed to maintain the equilibrium state of an object which is hardly detectable from a visual sensor. The likelihood of choosing some grasps over the others might arise from some other factors such as a task objective, approachability, comfortability, and the physical states of a robot's hand and an object. For example, a grasp for handling a tool might be different from a grasp for moving an object around. The kinematics of a robot's hand and relative position between the hand and an object also influences the feasibility of a grasp. Lastly, the characteristic of the surface of a robot's hand and an object (e.g., softness, fragility, stickiness, and slipperiness) also affects the physical interaction of a grasp. In fact, from our perspective, the position of human grasp largely depends on mood, concentration, and personal preferences. Additionally, the preference of human grasps might change over time due to knowledge learning from prior experiences. In robotics, robot's hand

or manipulator also has diverse models with various kinds of kinematics and controls that affect the preference of choosing one grasp over the others.

The dexterous grasp planning is a reasonable post-grasped analysis at best and an unreliable hint for grasp planning at worst. The fundamental problems of conventional dexterous grasp planning originate from several aspects such as over-simplified physical models, unrealistic assumptions, ambiguous definition of a good grasp and its verification. For example, many studies focused on the force-closure property of a grasp after it reached an equilibrium state and introduced grasp quality measurements based on analysis of forces that robot's fingers exert on an object. This approach usually assumes that:

- Everything is a rigid body.
- Robot's sensors and controls are accurate and precise.
- Substitute a grasp by a set of contact points where robot's fingertips touch an object.
- Use Coulomb frictional model for frictional forces and know the frictional coefficient between robot's fingertips and an object beforehand.
- Robot's fingertips can exert arbitrary force at contact points.
- The robot can accurately place every fingertip at contact points at the same time.
- The robot can accurately exert forces at contact points to balance out external forces applied on an object and keep it at equilibrium state.

Those meticulous and detailed assumptions are unrealistic and hard to achieve for a commercial-grade robot outside strictly controlled environment.

Additionally, the verification process of grasping result and its reproducibility in a real robot is another controversial topic in robotic grasping. This groundless complication arose from a straightforward question during our research: How to compare grasps synthesized from a grasp planning method that combine both ICRs and caging and the ones from conventional methods? In grasp theory, a stable grasp

is a good one and a conventional grasp planning method usually measured by the goodness of a grasp from force analysis such as force-closure condition. How to add caging to this existing quality measurement? Alternatively, is it better to introduce a new one? As a matter of fact, what is the standardized way to classify a grasp as good or bad/ success or failure? Some studies claimed that a grasp is successful if a real robot picked an object up and a grasp remained stable for a predefined amount of time, while the others grasped an object and intentionally shook it a few times to prove that a grasp can tolerate some external forces. On the other hand, task-oriented studies deemed a grasp is useful if a robot accomplished the task objective with it (e.g., moving an object to the designated place or emptying a bucket of gadgets).

In our opinion, using only contact points to represent a grasp is unreliable and insufficient to determine that such grasp can be achieved or not. In fact, the definition of dexterous grasping is overly simplified and ambiguous. We doubt that the resulting grasps executed based on contact points reflect the performance of a grasp represented by those contact points or the grasp planning that synthesizes them. Many other factors might influence the result of grasping such as the manipulator's kinematics, the presence of finger's links and palm, distribution of forces exerted on an object, object's physical states and gravity force. They are inherent properties of a manipulator, an object, and the environment. It might be hard to observe or control them accurately. Thus, grasp planning based on perfect information is impractical. With all those lengthy persuasions, we firmly believe that analytical methods are not suitable and insufficient for autonomous robot's grasp planning and we should revise grasp planning methods from another perspective, for example, a learning-based approach.

In our opinion, there is no silver bullet to the grasping problem that can make any robot grasps any object in any situation. It is practically impossible to correctly express the problem in mathematical expressions or physics models when dealing with noisy information from sensors and all kinds of different systems in each robot. Furthermore, grasping is a continuous process and usually is a part of more complicated operations. Isolating grasping from its actual context increases the problem's complexity (e.g., a grasp that can resist *arbitrary* force might be an extreme

condition for picking up an object from the ground) and its solution might be incompatible with the actual task (e.g., a grasp might be unreachable by a robot). Different situations have different requirements. It is easier to devise a specific method to solve a specific task. In this case, it might be easier to let a robot develop a grasp planning method best fit for its demand through trial and error.

With the emergence of learning-based grasp planning methods, haptic sensors and soft-body robots, the direction of robotic grasping should incline toward a grasp that maximizes contact surface between whole hand and an object instead of a grasp which only utilizes contacts on fingertips. Grasping an object by enclosing it with soft-body manipulator should be much easier to achieve than making fingertips touching their designated places at the same time. In our opinion, the visual cue of an object did not play a crucial role in grasping. It only provides a general position and shape of an object which leads to the initial pose that is easy to grasp an object. The critical part of the grasping process is how manipulator *reacts* when making contacts with an object. The information collected from touching an object or haptic sensing should provide necessary data for grasp planning to make a fine-tuned adjustment in fingers' and hand's pose best fitted for grasping an object in a closed-loop feedback control system. The fine-tuning process might be a learning-based method that specially trained for specific robot's hand, sensors and task objective.

The process described above is the outlines of robotic grasping that we envision it in the future. The approach is still very abstract and raises several problems of its own. For example, how to design a learning model to recognize good grasp? How to maintain equilibrium state from haptic data? How to implement a reliable and robust control system for soft-body manipulator? How to integrate soft hand and haptic sensors? How to manage the transition between approaching an object using visual cue and fine-tuning grasp's pose based on haptic sensing? All those open problems are left for the future studies.

APPENDIX

A. Force-closure condition between two points

Nguyen [22] stated that two soft fingers contacted object surface at points, $\boldsymbol{p}$ and $\boldsymbol{q}$, achieve force-closure if and only if the segment $\overline{\boldsymbol{pq}}$ joining two contact points lies strictly inside both of the friction cones or lies strictly inside both of the inverse friction cones. A simple approach to determine the force-closure conditions is to use an intersection test between double cone and point. We derive a force-closure test between two contact points from the inequalities in [113]. Let friction cone's half angle be $\boldsymbol{\theta}$ and two contact points with their normals be $(\boldsymbol{p_x}, \boldsymbol{n_x})$ and $(\boldsymbol{p_y}, \boldsymbol{n_y})$. A two-soft-finger grasp at those contact points achieves force-closure if all following predicates return true:

$$\mathcal{F}_p(\boldsymbol{p}_x, \boldsymbol{n}_x, \boldsymbol{p}_y, \boldsymbol{n}_y) =$$
$$\left((\boldsymbol{p}_y - \boldsymbol{p}_x) \cdot \boldsymbol{n}_x\right)^2 > \cos^2\theta \times \|\boldsymbol{p}_y - \boldsymbol{p}_x\|^2$$
$$\wedge \left((\boldsymbol{p}_x - \boldsymbol{p}_y) \cdot \boldsymbol{n}_y\right)^2 > \cos^2\theta \times \|\boldsymbol{p}_y - \boldsymbol{p}_x\|^2$$
$$\wedge \left((\boldsymbol{p}_y - \boldsymbol{p}_x) \cdot \boldsymbol{n}_x\right) \times \left((\boldsymbol{p}_x - \boldsymbol{p}_y) \cdot \boldsymbol{n}_y\right) > 0$$

The first inequality verifies that the point $\boldsymbol{p_x}$ lies strictly inside the friction cone or the inverse friction cone at the point $\boldsymbol{p_y}$. Likewise, the second inequality ensures that the point $\boldsymbol{p_y}$ lies in the cones at point $\boldsymbol{p_x}$. The last one determines that those points both lie either in the friction cones (squeezing grasp) or in the inverse cones (stretching grasp).

This method verifies the force-closure condition for a grasp with two soft-fingers. It is more straightforward than the conventional ones that test for origin-containment in the wrench space, e.g., $\boldsymbol{\epsilon}$-metric. It also yields more accurate results since it does not approximate a friction cone as a $\boldsymbol{N}$-sided pyramid which is widely used in many conventional grasp planning.

B. The condition of Independent contact regions between two triangles

The algorithm can efficiently determine if a pair of triangles form ICRs by testing the force-closure condition between all pairs of their extreme points. Let $\boldsymbol{ext(t_i)}$ be a function that returns a set of extreme points of a triangle $\boldsymbol{t_i}$. The ICR test between two triangles, $\boldsymbol{t_a}$ and $\boldsymbol{t_b}$, with their normals, $\boldsymbol{n_a}$ and $\boldsymbol{n_b}$, can be defined as:

$$\mathcal{F}_t(t_a, \boldsymbol{n}_a, t_b, \boldsymbol{n}_b) = \bigwedge_{\forall \boldsymbol{v} \in ext(t_a), \forall \boldsymbol{u} \in ext(t_b)} \mathcal{F}_p(\boldsymbol{v}, \boldsymbol{n}_a, \boldsymbol{u}, \boldsymbol{n}_b)$$

The proof has two parts. The first part is the ICR test between a point and a segment. Then, the second part generalizes the ICR test between two triangles.

Theorem 4 A contact point $(\boldsymbol{p}_r, \boldsymbol{n}_r)$ achieves force-closure with any contact point $\boldsymbol{p}_x$ on the segment $\overline{\boldsymbol{p}_s \boldsymbol{p}_t}$ with normal $\boldsymbol{n}_x$ if $(\boldsymbol{p}_r, \boldsymbol{n}_r)$ achieves force-closure with both endpoints of the segment, $(\boldsymbol{p}_s, \boldsymbol{n}_x)$ and $(\boldsymbol{p}_t, \boldsymbol{n}_x)$.

Proof Without loss of generality, we only consider a scenario where contact points lie strictly inside friction cones since it can prove the inverse-cone case in the same manner. Let $\mathcal{C}(\boldsymbol{p}, \boldsymbol{n})$ be a cone originated at point $\boldsymbol{p}$, its axis aligns with $\boldsymbol{n}$ and the aperture is 2θ. A force-closure test between $(\boldsymbol{p}_a, \boldsymbol{n}_a)$ and $(\boldsymbol{p}_b, \boldsymbol{n}_b)$ can be written as two point-and-cone intersection tests, $\boldsymbol{p}_a \in \mathcal{C}(\boldsymbol{p}_b, \boldsymbol{n}_b) \wedge \boldsymbol{p}_b \in \mathcal{C}(\boldsymbol{p}_a, \boldsymbol{n}_a)$. Using geometrical deduction, it is trivial to show that a force-closure test is equivalent to $\boldsymbol{p}_a \in \mathcal{C}(\boldsymbol{p}_b, \boldsymbol{n}_b) \cap \mathcal{C}(\boldsymbol{p}_b, -\boldsymbol{n}_a)$. Let shorten the right-hand side of the previous statement to $\mathcal{C}_{ba} = \mathcal{C}(\boldsymbol{p}_b, \boldsymbol{n}_b) \cap \mathcal{C}(\boldsymbol{p}_b, -\boldsymbol{n}_a)$. Since a cone is convex, an intersection between two cones is also convex. By the definition of convexity, it is sufficient to show that both endpoints of the segment, $\boldsymbol{p}_s$ and $\boldsymbol{p}_t$, lie strictly in an intersection between two

cones ($\boldsymbol{p_s} \in \mathcal{C}_{rs}$ and $\boldsymbol{p_t} \in \mathcal{C}_{rt}$, respectively) to proof that any points $\boldsymbol{p_x}$ in the segment $\overline{\boldsymbol{p_s p_t}}$ lie strictly in $\mathcal{C}_{rx}$. Hence, a contact point $(\boldsymbol{p_r}, \boldsymbol{n_r})$ achieves force-closure with any contact point $\boldsymbol{p_x}$ on the segment $\overline{\boldsymbol{p_s p_t}}$ with normal $\boldsymbol{n_x}$ if $(\boldsymbol{p_r}, \boldsymbol{n_r})$ achieves force-closure with both endpoints of the segment, $(\boldsymbol{p_s}, \boldsymbol{n_x})$ and $(\boldsymbol{p_t}, \boldsymbol{n_x})$.

Corollary 5 A triangle $\boldsymbol{t_a}$ with contact normal $\boldsymbol{n_a}$ form an independent contact region with a triangle $\boldsymbol{t_b}$ with contact normal $\boldsymbol{n_b}$ if and only if all pairs of extreme points between two triangles achieve force-closure.

Proof Let $\boldsymbol{p_x}$ be any points inside $\boldsymbol{t_a}$ and P be a set of points that lies on the edges of $\boldsymbol{t_a}$. Similarly, let $\boldsymbol{p_y}$ and Q be counterparts of $\boldsymbol{p_x}$ and P in $\boldsymbol{t_b}$. It is obvious that every $\boldsymbol{p_x}$ lies on a segment which has both endpoints in P and any point in P lies on segment which has extreme points of $\boldsymbol{t_a}$ as its endpoint. Those statements also apply to $\boldsymbol{p_y}$ and Q in $\boldsymbol{t_b}$. *Theorem 4* implies that every possible pair of $\boldsymbol{p_x}$ and $\boldsymbol{p_y}$ achieve force-closure if every pair of points in P and Q, one from each set, achieve force-closure. Furthermore, every pair of points between P and Q achieve force-closure if every pair of their endpoints, the extreme points of each triangle, achieve force-closure. In other words, every possible pair of $\boldsymbol{p_x}$ and $\boldsymbol{p_y}$ achieve force-closure and triangle $\boldsymbol{t_a}$ form an independent contact region with $\boldsymbol{t_b}$ if all pairs of extreme points between $\boldsymbol{t_a}$ and $\boldsymbol{t_b}$ achieve force-closure.

C. The pseudo code of iterative ICRs algorithm

```
1  function IterativeIcr(int n, Mat A1, Mat S0)
2   int h ← 0
3   Mat A ← I
4   Mat S ← S0
5   (int, int, int)[] ans ← empty array
6   bool finished ← false
7   while not finished do
8    finished ← true
9    for int i ← 0 to n – 2 do
10    for int j ← i + 1 to n – 1 do
11     if is_subset_of(A[i, *], S[j, *]) is true then
12      finished ← false
13      ans[] ← (i, j, h)
14     end if
15    end for
16   end for
17   if finished is true then
18    return ans
19   end if
20   Mat newA ← A
21   Mat newS ← S
22   for int i ← 0 to n – 1 do
23    for int j ← 0 to n – 1 do
24     if A1[i, j] is true then
25      newA[i, *] ← union(newA[i, *], A[j, *])
26      newS[i, *] ← intersect(newS[i, *], S[j, *])
27     end if
28    end for
29   end for
30   h ← h + 1
31   A ← newA
32   S ← newS
33
34  end while
    end function
```


Explanation of symbols and variables:

- **n** is the number of triangles in an object mesh.
- **Mat** is a $n \times n$ boolean matrix that represent a collection of an adjacency set or a stable set. the value at row i and column j of a matrix, M[i, j], is **true** if a triangle t_i is a member of the set M_j, otherwise, it is **false**. M[i, *] is the i^{th} row of matrix M which represents the set M_i.
- A1 is a matrix of A^1, i.e., adjacency matrix of undirected graph G. Note that each row of this matrix contains at most 3 members because, by definition, two triangles are adjacent only if they share the same edge.
- S0 is a matrix of S^0. By the definition of the stable set, S0[i, j] is **true** if the pair of triangles, t_i and t_j, forms a valid ICRs, otherwise, it is **false**.
- I is an identity matrix such that I[i, j] is **true** if i equals to j. Otherwise, it is **false**.
- **h** is the current adjacency distance h starting from zero and increases by one at the end of each iteration of the algorithm.
- A and S are boolean matrices that represent collections of adjacency sets and stable sets at adjacency distance **h**. In short, they denote A^h and S^h, respectively.
- **ans** is an array that contains all ICRs found in our algorithm. We denote ICRs as an integer triplet (x, y, z) that represents a pair of adjacency sets (A_x^z, A_y^z). Note that the operation at line 13 in the pseudo code is appending a new triplet into this array.
- There are three set operations in this algorithm: **union**, **intersect** and **is_subset_of**. All three operations receive two sets (in this context, two rows of boolean matrices) as input. The first two operations return a set while the last one returns a boolean value. Those operations can be written in pseudo-code as follow:

```
 1  function union(bool[n] left, bool[n] right)
 2    bool[n] result
 3    for int i ← 0 to n - 1 do
 4      result[i] ← left[i] or right[i]
 5    end for
 6    return result
 7  end function
 8
 9  function intersect(bool[n] left, bool[n] right)
10    bool[n] result
11    for int i ← 0 to n - 1 do
12      result[i] ← left[i] and right[i]
13    end for
14    return result
15  end function
16
17  function is_subset_of(bool[n] left, bool[n] right)
18    for int i ← 0 to n - 1 do
19      if left[i] is true and right[i] is false then
20        return false
21      end if
22    end for
23    return true
24  end function
```


- The time complexity is $O(h_{max}n^3)$ where h_{max} is the maximum value adjacency distance h, i.e., number of iteration of outer-most while loop.
- The memory complexity is $O(n^2)$ which is the size of a boolean matrix.

D. Synthetic data for training grasp quality

Grasps generated for the synthetic dataset can be classified into two types as zero-score and non-zero-score. The zero-score grasps are the ones which have both contact points on the same edge of a polygonal object. We introduce non-zero-score grasps for only five simple objects: circle, square, diamond, hConcaveSmall and vConcaveSmall. We use different heuristic methods to calculate scores based on the position of grasp's contact points on some parts of each object, for example, grasps for square and diamond are only synthesized when their contact points are on the opposing edges.

- Circle

The precondition of non-zero-score grasps for the circle is the angle between contact normal must be higher than 90 degrees. The method generates the grasps for circle because we want neural networks to recognize grasp quality invariant to rotation transform. In our opinion, an ideal grasp on our circle, which is a 16-sided polygon, should have contact points at the center of opposing edges. So, the algorithm calculates the score from the position of contact points on the circle's edge and the angle between contact normals. A grasp at the center of circle's edge is better than a grasp near the corner of circle's edge. Since the circle is convex, a grasp with the higher angle between contact normals is better than the ones with a lower angle. So, the algorithm averages the final score from two factors: the angle between contact normal ($score_{angle}$) and the distance from contact points to the end points of their edge ($score_{edge}$).

$$score_{angle} = -\boldsymbol{n}_a \cdot \boldsymbol{n}_b$$
$$score_{ea} = \min(\|\boldsymbol{e}_{a1} - \boldsymbol{p}_a\|, \|\boldsymbol{e}_{a2} - \boldsymbol{p}_a\|)$$
$$score_{eb} = \min(\|\boldsymbol{e}_{b1} - \boldsymbol{p}_b\|, \|\boldsymbol{e}_{b2} - \boldsymbol{p}_b\|)$$
$$score_{edge} = \frac{score_{ea} + score_{eb}}{L_e}$$
$$score = 0.5score_{angle} + 0.5score_{edge}$$

$\boldsymbol{p}_x$ and $\boldsymbol{n}_x$ are position and normal of a contact point $\boldsymbol{x}$. $\boldsymbol{e}_{x1}$and $\boldsymbol{e}_{x2}$ are the end points of the edge touched by a contact point $\boldsymbol{x}$. L_e is the edge length of each side of a polygon.

- Diamond and Square

The precondition for synthetic grasps of diamond and square is the position of their contact points must be on the opposing edges. The best grasp for square is precisely the one having contact points at the center of opposing edges while the worst one has contact points at the opposing diagonal corner. We want a neural network to learn that the grasps placed on the opposing edge should have similar quality if the position of their contact points is relatively close. The algorithm averages the score from two types of normalized distance: distance between contact points ($score_{dist}$) and distance between a contact point to the center of an object ($score_{center}$) at the origin (0, 0). L_e is edge length and L_d is the diagonal length of an object. In this case, both square and diamond have the same values: 20 units for edge length and $\sqrt{800}$ for diagonal length.

$$score_{dist} = 1 - \frac{\|\boldsymbol{p}_a - \boldsymbol{p}_b\| - L_e}{L_d - L_e}$$

$$score_{center} = 1 - \frac{\|\boldsymbol{p}_a\| + \|\boldsymbol{p}_b\| - L_e}{L_d - L_e}$$

$$score = 0.5score_{dist} + 0.5score_{center}$$

- hConcaveSmall and vConcaveSmall

Similar to the previous objects, the precondition for synthetic grasps of hConcaveSmall and vConcaveSmall is the position of their contact points must be on the opposing parallel edges and the opposing concave parts of an object. The calculation of grasp's score on opposing parallel edges is the same as the one used for square and diamond. For concave parts, the algorithm calculates

the score from the distance from the contact points to the main axis, (x-axis for hConcaveSmall and y-axis for vConcaveSmall). Since grasping the objects on its concave parts is preferred and more stable due to its ability to cage an object (i.e., pregrasping cage).

$$score_{dist} = 1 - \frac{|d_{sa}||d_{sb}|}{L_e^2}$$

$$score = 0.3 + 0.7score_{dist}$$

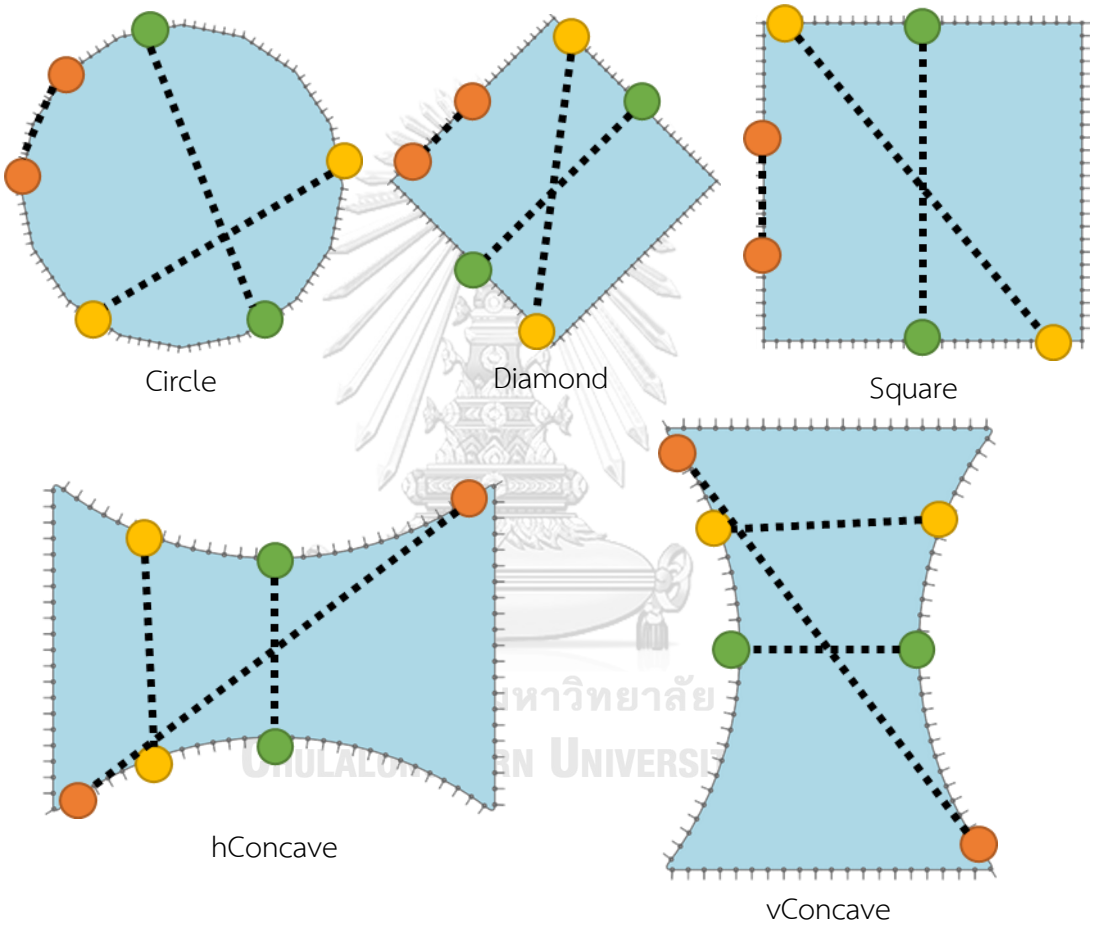


Figure 41 Example of labeled grasps generated from heuristic methods. Grasp quality of each grasp is visualized as its color starting from lowest (orange), medium (yellow) to highest (green).

REFERENCES

[1] E. Rimon, and J. Burdick, "On force and form closure for multiple finger grasps," *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, 1996, pp. 1795–1800.

[2] A. Bicchi, "On the closure properties of robotic grasping," *The International Journal of Robotics Research,* vol. 14, no. 4, pp. 319–334, 1995.

[3] B. Mishra, J. T. Schwartz, and M. Sharir, "On the existence and synthesis of multifinger positive grips," *Algorithmica,* vol. 2, no. 1, pp. 541–558, 1987.

[4] Y. H. Liu, "Qualitative test and force optimization of 3-D frictional form-closure grasps using linear programming," *Ieee Transactions on Robotics and Automation,* vol. 15, no. 1, pp. 163-173, Feb, 1999.

[5] X. Zhu, H. Ding, and S. K. Tso, "A Pseudodistance Function and its Applications," *IEEE Transactions on Robotics and Automation,* vol. 20, no. 2, pp. 344-352, 2004.

[6] D. Kirkpatrick, B. Mishra, and C.-K. Yap, "Quantitative Steinitz's theorems with applications to multifingered grasping," *Discrete & Computational Geometry,* vol. 7, no. 3, pp. 295-318, 1992/03/01, 1992.

[7] C. Ferrari, and J. Canny, "Planning optimal grasps," *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, IEEE, 1992, pp. 2290–2295.

[8] Y. Zheng, "An Efficient Algorithm for a Grasp Quality Measure," *IEEE Transactions on Robotics,* vol. 29, no. 2, pp. 579-585, 2013.

[9] V.-D. Nguyen, "The Synthesis of Stable Force-Closure Grasps," MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1986.

[10] T. Chao-Ping, and A. C. Kak, "Fast construction of force-closure grasps," *IEEE Transactions on Robotics and Automation,* vol. 12, no. 4, pp. 615-626, 1996.

[11] Z. Xiangyang, D. Han, and L. Hanxiong, "A quantitative measure for multi-fingered grasps." pp. 213-219 vol.1.

[12] X. Zhu, and J. Wang, "Synthesis of force-closure grasps on 3-d objects based on the Q distance," *IEEE Transactions on Robotics and Automation,* vol. 19, no. 4, pp. 669-679, 2003.

[13] J. Shi, and G. S. Koonjul, "Real-Time Grasping Planning for Robotic Bin-Picking and Kitting Applications," *IEEE Transactions on Automation Science and Engineering,* vol. 14, no. 2, pp. 809-819, 2017.

[14] K. Byoung-Ho, O. Sang-Rok, Y. Byung-Ju, and S. Il Hong, "Optimal grasping based on non-dimensionalized performance indices." pp. 949-956 vol.2.

[15] E. Chinellato, A. Morales, R. B. Fisher, and A. P. d. Pobil, "Visual quality measures for Characterizing Planar robot grasps," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 35, no. 1, pp. 30-41, 2005.

[16] D. Dan, L. Yun-Hui, and W. Shuguo, "Computation of 3-D form-closure grasps," *IEEE Transactions on Robotics and Automation,* vol. 17, no. 4, pp. 515-522, 2001.

[17] J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet, "On Computing Four-Finger Equilibrium and Force-Closure Grasps of Polyhedral Objects," *The International Journal of Robotics Research,* vol. 16, no. 1, pp. 11-35, 1997.

[18] A. Bicchi, and V. Kumar, "Robotic grasping and contact: A review," *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, IEEE, 2000, pp. 348–353.

[19] R. Suárez, J. Cornellà, and M. R. Garzón, *Grasp quality measures*: Institut d'Organització i Control de Sistemes Industrials, 2006.

[20] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Robotics and Autonomous Systems,* vol. 60, no. 3, pp. 326-336, 2012.

[21] M. A. Roa, and R. Suárez, “Grasp quality measures: review and performance,” *Autonomous Robots,* vol. 38, no. 1, pp. 65-88, 2015/01/01, 2015.

[22] V.-D. Nguyen, “Constructing force-closure grasps,” *International Journal of Robotics Research,* vol. 7, no. 3, pp. 3-16, 1988.

[23] N. S. Pollard, “Closure and quality equivalence for efficient synthesis of grasps from examples,” *International Journal of Robotics Research,* vol. 23, no. 6, pp. 595-613, Jun, 2004.

[24] M. A. Roa, and R. Suarez, "Geometrical approach for grasp synthesis on discretized 3d objects." pp. 3283-3288.

[25] M. A. Roa, and R. Suárez, "Determination of independent contact regions on discretized 3d objects," *Assembly and Manufacturing, 2007. ISAM'07. IEEE International Symposium on*, IEEE, 2007, pp. 191–196.

[26] M. A. Roa, and R. Suárez, "Independent contact regions for frictional grasps on 3D objects," *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, IEEE, 2008, pp. 1622–1627.

[27] M. A. Roa, and R. Suarez, “Computation of Independent Contact Regions for Grasping 3-D Objects,” *IEEE Transactions on Robotics,* vol. 25, no. 4, pp. 839-850, 2009.

[28] R. Krug, D. Dimitrov, K. Charusta, and B. Iliev, "On the efficient computation of independent contact regions for force closure grasps," *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, pp. 586–591.

[29] B. A. Dang-Vu, M. A. Roa, and C. Borst, "Extended independent contact regions for grasping applications." pp. 3527-3534.

[30] R. Krug, D. Dimitrov, K. Charusta, and B. Iliev, "Prioritized independent contact regions for form closure grasps." pp. 1797-1803.

[31] T. Phoka, P. Vongmasa, C. Nilwatchararang, P. Pipattanasomporn, and A. Sudsang, "Planning optimal independent contact regions for two-fingered force-

closure grasp of a polygon," *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 1175–1180.

[32] T. Phoka, P. Vongmasa, C. Nilwatchararang, P. Pipattanasomporn, and A. Sudsang, "Optimal independent contact regions for two-fingered grasping of polygon," *Robotica,* vol. 30, pp. 879-889, Oct, 2012.

[33] C. Rosales, L. Ros, J. M. Porta, and R. Suarez, "Synthesizing grasp configurations with specified contact regions," *International Journal of Robotics Research,* vol. 30, no. 4, pp. 431-443, Apr, 2011.

[34] H. Jeong, and J. Cheong, "In-hand rolling motion planning using independent contact region (ICR) with guaranteed grasp quality margin." pp. 3239-3244.

[35] J. Fontanals, B. A. Dang-Vu, O. Porges, J. Rosell, and M. A. Roa, "Integrated grasp and motion planning using independent contact regions." pp. 887-893.

[36] N. Alvarado, R. Suárez, and M. A. Roa, "Determining independent contacts regions to immobilize 2D articulated objects." pp. 4292-4297.

[37] W. Kuperberg, "Problems on polytopes and convex sets," *DIMACS Workshop on polytopes*, 1990, pp. 584–589.

[38] A. Sudsang, "A sufficient condition for capturing an object in the plane with disc-shaped robots," *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, IEEE, 2002, pp. 682–687.

[39] P. Vongmasa, and A. Sudsang, "Coverage diameters of polygons," *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006, pp. 4036–4041.

[40] P. Pipattanasomporn, and A. Sudsang, "Two-finger caging of concave polygon," *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, IEEE, 2006, pp. 2137–2142.

[41] P. Pipattanasomporn, P. Vongmasa, and A. Sudsang, "Two-finger squeezing caging of polygonal and polyhedral object," *Robotics and Automation, 2007 IEEE International Conference on*, IEEE, 2007, pp. 205–210.

[42] P. Pipattanasomporn, and A. Sudsang, “Two-Finger Caging of Nonconvex Polytopes,” *IEEE Transactions on Robotics,* vol. 27, no. 2, pp. 324-333, 2011.

[43] T. Allen, J. Burdick, and E. Rimon, "Two-fingered caging of polygons via contact-space graph search," *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 4183–4189.

[44] J. Erickson, S. Thite, F. Rothganger, and J. Ponce, “Capturing a Convex Object With Three Discs,” *IEEE Transactions on Robotics,* vol. 23, no. 6, pp. 1133-1140, 2007.

[45] M. Vahedi, and A. F. v. d. Stappen, "Geometric Properties and Computation of Three-Finger Caging Grasps of Convex Polygons." pp. 404-411.

[46] W. Wan, R. Fukui, M. Shimosaka, T. Sato, and Y. Kuniyoshi, "On the caging region of a third finger with object boundary clouds and two given contact positions," *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 4154–4161.

[47] P. Pipattanasomporn, P. Vongmasa, and A. Sudsang, "Caging rigid polytopes via finger dispersion control," *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 1181–1186.

[48] P. Pipattanasomporn, "Caging of Rigid Polytope via Dispersion Control of Point Fingers," Chulalongkorn University, 2011.

[49] N. Suarod, N. Boonpinon, and A. Sudsang, "A heuristic method for computing caging formation of polygonal object." pp. 823-828.

[50] T. Makapunyo, T. Phoka, P. Pipattanasomporn, N. Niparnan, and A. Sudsang, "Measurement framework of partial cage quality," *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, IEEE, 2012, pp. 1812–1816.

[51] T. Makapunyo, T. Phoka, P. Pipattanasomporn, N. Niparnan, and A. Sudsang, "Measurement framework of partial cage quality based on probabilistic motion planning," *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 1574–1579.

[52] S. Makita, and K. Nagata, “Evaluation of quality of partial caging by a planar two-fingered hand,” *Advanced Robotics,* vol. 30, no. 3, pp. 178-189, 2016/02/01, 2016.

[53] J. Mahler, F. T. Pokorny, Z. McCarthy, A. F. v. d. Stappen, and K. Goldberg, “Energy-Bounded Caging: Formal Definition and 2-D Energy Lower Bound Algorithm Based on Weighted Alpha Shapes,” *IEEE Robotics and Automation Letters,* vol. 1, no. 1, pp. 508-515, 2016.

[54] G. A. Pereira, M. F. Campos, and V. Kumar, “Decentralized algorithms for multi-robot manipulation via caging,” *The International Journal of Robotics Research,* vol. 23, no. 7, pp. 783–795, 2004.

[55] A. Sudsang, and J. Ponce, "A new approach to motion planning for disc-shaped robots manipulating a polygonal object in the plane," *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, IEEE, 2000, pp. 1068–1075.

[56] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-robot manipulation via caging in environments with obstacles," *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 1471–1476.

[57] D. J. Cappelleri, M. Fatovic, and U. Shah, "Caging micromanipulation for automated microassembly," *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3145-3150.

[58] J. Chen, M. Gauci, W. Li, A. Kolling, and R. Groß, “Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots,” *IEEE Transactions on Robotics,* vol. 31, no. 2, pp. 307-321, April, 2015.

[59] G. A. S. Pereira, V. Kumar, J. R. Spletzer, C. J. Taylor, and M. F. M. Campos, "Cooperative Transport of Planar Objects by Multiple Mobile Robots Using Object Closure." pp. 287-296.

[60] R. Diankov, S. S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning with caging grasps." pp. 285-292.

[61] A. Jain, and C. C. Kemp, "Pulling open doors and drawers: Coordinating an omni-directional base and a compliant arm with equilibrium point control," *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, pp. 1807–1814.

[62] M. Vahedi, and A. F. v. d. Stappen, "Caging Polygons with Two and Three Fingers," *The International Journal of Robotics Research,* vol. 27, no. 11-12, pp. 1308-1324, 2008.

[63] S. Makita, and W. Wan, "A survey of robotic caging and its applications," *Advanced Robotics,* vol. 31, no. 19-20, pp. 1071-1085, 2017/10/18, 2017.

[64] K. G. Gopalakrishnan, and K. Goldberg, "D-space and Deform Closure Grasps of Deformable Parts," *The International Journal of Robotics Research,* vol. 24, no. 11, pp. 899-910, 2005/11/01, 2005.

[65] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *The International Journal of Robotics Research,* vol. 31, no. 7, pp. 886–900, 2012.

[66] A. Rodriguez, and M. T. Mason, *Two finger caging: squeezing and stretching*: Springer, 2009.

[67] Y. Maeda, N. Kodera, and T. Egawa, "Caging-based grasping by a robot hand with rigid and soft parts," *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 5150–5155.

[68] W. Wan, R. Fukui, M. Shimosaka, T. Sato, and Y. Kuniyoshi, "A new "grasping by caging" solution by using eigen-shapes and space mapping." pp. 1566-1573.

[69] W. Wan, R. Fukui, M. Shimosaka, T. Sato, and Y. Kuniyoshi, "Grasping by caging: A promising tool to deal with uncertainty," *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 5142–5149.

[70] C. Borst, M. Fischer, and G. Hirzinger, "Grasping the dice by dicing the grasp," *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 2003, pp. 3692–3697.

[71] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum volume bounding box decomposition for shape approximation in robot grasping." pp. 1628-1633.

[72] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives." pp. 1824-1829 vol.2.

[73] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp Planning via Decomposition Trees." pp. 4679-4684.

[74] S. El-Khoury, and A. Sahbani, "A new strategy combining empirical and analytical approaches for grasping unknown 3D objects," *Robotics and Autonomous Systems,* vol. 58, no. 5, pp. 497-507, 2010/05/31/, 2010.

[75] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng, "Robotic grasping of novel objects," *Advances in neural information processing systems*, pp. 1209-1216, 2007.

[76] A. Saxena, J. Driemeyer, J. Kearns, C. Osondu, and A. Y. Ng, "Learning to Grasp Novel Objects Using Vision," *Experimental Robotics: The 10th International Symposium on Experimental Robotics*, O. Khatib, V. Kumar and D. Rus, eds., pp. 33-42, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[77] R. Detry, D. Kraft, A. G. Buch, N. Krüger, and J. Piater, "Refining grasp affordance models by experience." pp. 2287-2293.

[78] S. El-Khoury, and A. Sahbani, "On computing robust n-finger force-closure grasps of 3D objects." pp. 2480-2486.

[79] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergström, D. Kragic, and A. Morales, "Mind the gap - robotic grasping under incomplete observation." pp. 686-693.

[80] E. Klingbeil, D. Rao, B. Carpenter, V. Ganapathi, A. Y. Ng, and O. Khatib, "Grasping with application to an autonomous checkout robot." pp. 2837-2844.

[81] G. Kootstra, M. Popović, J. A. Jørgensen, K. Kuklinski, K. Miatliuk, D. Kragic, and N. Krüger, "Enabling grasping of unknown objects through a synergistic use of

edge and surface information," *The International Journal of Robotics Research,* vol. 31, no. 10, pp. 1190-1213, 2012.

[82] D. Fischinger, and M. Vincze, "Empty the basket - a shape based learning approach for grasping piles of unknown objects." pp. 2051-2057.

[83] H. Dang, and P. K. Allen, "Semantic grasping: planning task-specific stable robotic grasps," *Autonomous Robots,* vol. 37, no. 3, pp. 301-316, October 01, 2014.

[84] M. T. Ciocarlie, and P. K. Allen, "Hand Posture Subspaces for Dexterous Robotic Grasping," *The International Journal of Robotics Research,* vol. 28, no. 7, pp. 851-867, 2009/07/01, 2009.

[85] L. Sergey, P. Peter, K. Alex, I. Julian, and Q. Deirdre, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, pp. 0278364917710318, 2017.

[86] D. Guo, F. Sun, H. Liu, T. Kong, B. Fang, and N. Xi, "A hybrid deep architecture for robotic grasp detection." pp. 1609-1614.

[87] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-Driven Grasp Synthesis - A Survey," *IEEE Transactions on Robotics,* vol. 30, no. 2, pp. 289-309, 2014.

[88] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a Multi-Armed Bandit model with correlated rewards." pp. 1957-1964.

[89] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view Convolutional Neural Networks for 3D Shape Recognition," in Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 945-953.

[90] R. Goetschalckx, P. Poupart, and J. Hoey, "Continuous correlated beta processes," in Proceedings of the Twenty-Second international joint

conference on Artificial Intelligence - Volume Volume Two, Barcelona, Catalonia, Spain, 2011, pp. 1269-1274.

[91] S. Pandey, D. Chakrabarti, and D. Agarwal, “Multi-armed bandit problems with dependent arms,” in Proceedings of the 24th international conference on Machine learning, Corvalis, Oregon, USA, 2007, pp. 721-728.

[92] J. Weisz, and P. K. Allen, "Pose error robust grasping from contact wrench space metrics." pp. 557-562.

[93] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. Aparicio Ojea, and K. Goldberg, “Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics,” *ArXiv e-prints*, 2017, 2017.

[94] J. Mahler, M. Matl, X. Liu, A. Li, D. V. Gealy, and K. Goldberg, “Dex-Net 3.0: Computing Robust Robot Suction Grasp Targets in Point Clouds using a New Analytic Model and Deep Learning,” *CoRR,* vol. abs/1709.06670, 2017.

[95] N. Tian, M. Matl, J. Mahler, Y. X. Zhou, S. Staszak, C. Correa, S. Zheng, Q. Li, R. Zhang, and K. Goldberg, "A cloud robot system using the dexterity network and berkeley robotics and automation as a service (Brass)." pp. 1615-1622.

[96] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *Robotics and Automation, IEEE Transactions on,* vol. 12, no. 4, pp. 566–580, 1996.

[97] G. Sánchez, and J.-C. Latombe, "A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking," *Robotics Research*, Springer Berlin Heidelberg, 2003, pp. 403-417.

[98] J. J. Kuffner Jr, and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, 2000, pp. 995–1001.

[99] J. Cornella, and R. Suárez, "Fast and flexible determination of force-closure independent regions to grasp polygonal objects," *Robotics and Automation,*

2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, IEEE, 2005, pp. 766–771.

[100] OpenMP Architecture Review Board. "OpenMP C and C++ Application Program Interface Version 2.0," http://www.openmp.org/mp-documents/cspec20.pdf.

[101] Nvidia. "CUDA Toolkit Documentation," http://docs.nvidia.com/cuda/.

[102] Nvidia. "CUDA C Programming Guide," http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.

[103] Nvidia. "CUDA Occupancy Calculator," http://developer.download.nvidia.com/compute/cuda/CUDA_Occupancy_calculator.xls.

[104] Blender Foundation. "Blender - a 3D modelling and rendering package," http://blender.org/.

[105] A. Kasper, Z. Xue, and R. Dillmann, "The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics," *The International Journal of Robotics Research,* vol. 31, no. 8, pp. 927-934, 2012.

[106] Artec Group. "Gallery of Scanned 3D Models Using Artec 3D Scanners," https://www.artec3d.com/3d-models.

[107] P. Cignoni, M. Corsini, and G. Ranzuglia, "Meshlab: an open-source 3d mesh processing system," *Ercim news,* vol. 73, pp. 45–46, 2008.

[108] A. Gessler, T. Schulze, K. Kulling, and D. Nadlinger. "Open Asset Import Library," http://www.assimp.org/.

[109] W. Schroeder, K. Martin, and B. Lorensen, *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, 4 ed.: Kitware, 2006.

[110] M. Akimaliev, and M. F. Demirci, "Shape Abstraction through Multiple Optimal Solutions." pp. 588-596.

[111] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[112] R. Lopez, and Artelnics, "Open Neural Networks (OpenNN)," Artelnics, 2017.

APPENDIX

VITA

Teesit Makapunyo was born in Bangkok, Thailand, in October 1988. He received a bachelor's degree in engineering from Chulalongkorn University since 2010. His field of study is computer engineering. His research interest includes robotic grasp planning, machine learning, parallel computing and computer vision.