DEXTEROUS GRASPING OF NOVEL OBJECTS FROM A SINGLE VIEW

BY

CHAO ZHAO

A thesis submitted to
The University of Birmingham
for the degree of
MSC BY RESEARCH

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
MAY 2019

Abstract

In this thesis, a novel generative-evaluative method was proposed to solve the problem of dexterous grasping of the novel object with a single view. The generative model is learned from the human demonstration. The grasps generated by the generative model is used to train the evaluative model. Two novel evaluative network architectures are proposed. The evaluative model is a deep evaluative network that is trained in the simulation. The generative-evaluative method is tested in a real grasp dataset with 49 previously unseen challenging objects. The generative-evaluative method achieves a success rate of 78% that outperforms the purely generative method, that has a success rate of 57%. The thesis provides insights into the strengths and weaknesses of the generative-evaluative method by comparing different deep network architectures.

Acknowledgement

First of all, special mention goes to my supervisor Prof. Jeremy Wyatt and Prof. Ales Leonardis. My MSc by Research has been an amazing experience, and I thank them all great support and suggestions for my research and life. I also thank Rusen Aktas and Marek Kopicki. This thesis will not be presented without their work and support. I am also hugely appreciative of Prof. David Parker who is my thesis group member to examine my research progress and giving me the useful suggestion. Finally, but by no means least, thanks to my mother, father and fiancee. They are very important people for me, and I dedicate this thesis to them.

Sincerely,
Chao Zhao

Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Motivation

Among mammalian hands, human hands are distinct. Humans can use their hands with dexterity, which is key to human culture and technological progress[1]. Dexterous human hands and human strategic planning are the most critical components of these attainments. Robotic grasping has been an active research area for many years and has made steady progress through the development of a number of efficient and robust algorithms and of increasingly human-like hands[2]. In robotic grasping, dexterous grasping is one of the most exciting areas[3]. The goal of dexterous grasping is to achieve functions which are similar to those involved in human grasping. If this goal were to be achieved, dexterous grasping could be applied to a number of situations, such as those where it would be dangerous or impossible to use a human agent.

This thesis focuses on the problem of the dexterous grasping of novel objects when only a single view of the object is available. There are three requirements here. The first requirement is dexterous grasping. To enable robots to work in a human-centric environment, dexterous grasping is essential. However, due to its inherent complexity and the inevitable uncertainty of the models relating to it, dexterous grasping remains a challenge. In order to carry out dexterous grasping, multi-fingered or specialized robotic hands must be used, which allows for precise control of force and motion. Multi-fingered robotic hands usually have around ten times more parameters compared to a simple parallel gripper. The second requirement is that the grasping needs to be performed on novel objects. Grasping novel objects means grasping a previously unknown object. Thus, the robot should not have any prior experience of grasping novel objects. The third requirement is that the robot needs to grasp an object which it can only see from one single viewpoint. However, the viewpoint in question may be from above or from one side or another, and the robot always has no access to a complete image of the object‘s shape. The combination of these three requirements makes the grasp planning problem particularly challenging.

For example, an unknown object, such as a mug, and a depth map of this object from a

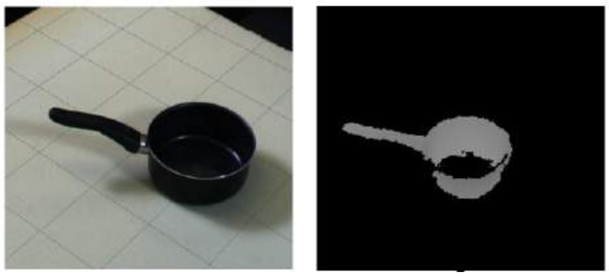


Figure 1.1: Object and its depth image.

single view-point are given. The purpose is to find a grasp configuration for a multi-finger hand to grasp the mug. However, the depth map from a single view means that only part of the object‘s surface will be reconstructed, so it is hard to fit a known object model. Therefore, given an incomplete reconstruction of the surface of the object, a large number of hand parameters and an uncertain object mass, friction coefficient, etc., it is hard to employ classical analytical methods based on the laws of mechanics to predict grasp configuration or quality. Deep learning is a possible solution to this. It could be used for predicting the quality of the planned grasp. For example, some research has used deep learning for two-finger grippers [4, 5, 6, 7, 8, 9], and a small number of studies have explored the use of deep learning for dexterous grasping [10, 11, 12, 13, 14]. While steady progress in this area has been made, and using deep learning for grasp planning looks promising, currently, dexterous grasping which is driven by deep learning works only in simulation or incompletely in real-world robot tests. This indicated that the use of deep learning for dexterous grasping is an area well worth exploring.

In summary, the uncertainties during grasping (object shape, weight and friction etc.) and the different kinds of planning strategies resulting from these uncertainties are still major challenges in robotic grasping and manipulation. The central part of this thesis attempts to use an evaluative model driven by deep learning to evaluate grasps generated by a generative model learned from demonstrations. The evaluative model is trained in simulation using grasps generated by the generative model. This thesis demonstrates how it is possible to use the benefits of deep networks — via a learning-based dexterous grasps generator working in a generative-evaluative learning architecture.

1.2 Contributions

This thesis proposes a generative-evaluative method for dexterous grasping that incorporates the model of uncertainty for grasping during the evaluative model training. The experiment results show that the generative-evaluative model improves the robustness and adaptability of grasping. This thesis makes the following contributions in the robot grasping area:

1. This thesis presents the first generative-evaluative method through which the generative model and the evaluative model are learned for the dexterous grasping of novel objects. Section 3.2, the generative model, was conducted by Rusen Aktas (a graduated PhD student supervised by Prof. Jeremy Wyatt and Prof. Ales Leonardis). Marek Kopicki (Research Fellow in School of Computer Science, University of Birmingham), who finished the real robot test as shown in Section 3.5.3.

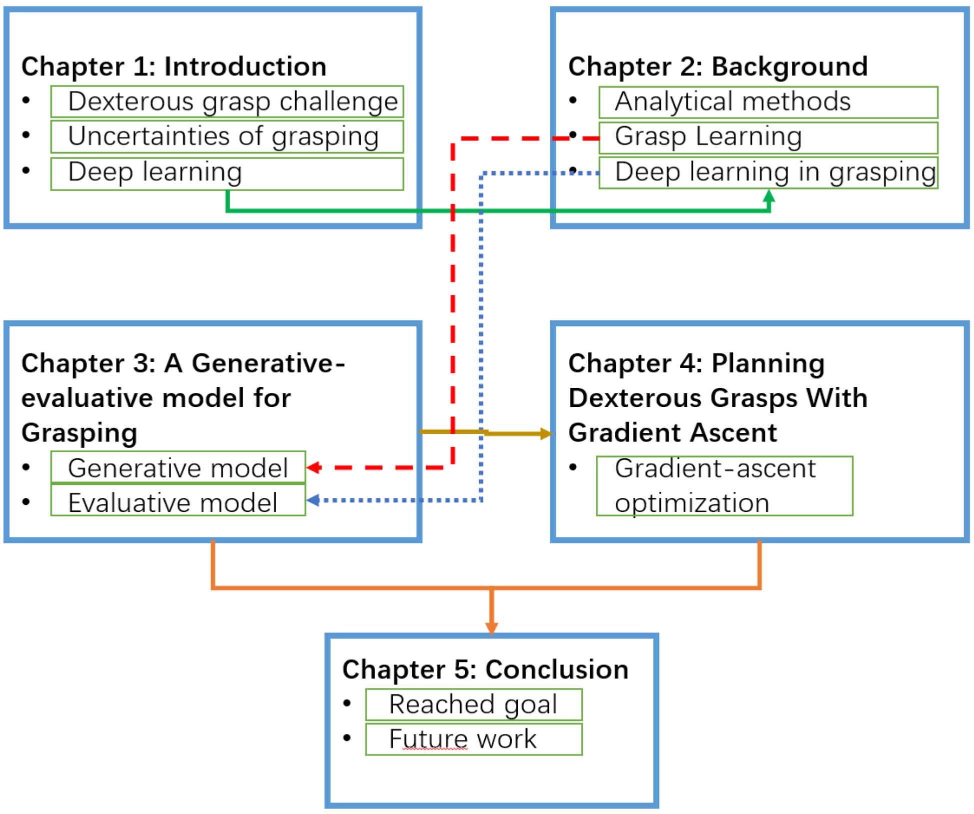
2. Two novel evaluative network architectures are proposed. Both evaluative networks take robot hand parameters and a depth image as network input and predict the grasp success probability. The dataset for training network captures the uncertainty in unobservable variables, such as mass and friction. The experiment results of the evaluative network show a robust evaluation.

3. Gradient ascent is used to optimise the grasp parameters by finding the grasp parameters which maximise the probability of success of the grasp has been tested. Throughout the experiment, results show that the method is not helpful for improving grasp quality. However, some promising approaches for improving this optimisation method are proposed according to the analysis of experimental results.

The code and the dataset will be released on Github before 2019 September.

1.3 Thesis outline

In this section, I briefly summarize the content of each chapter of the thesis and its structure, as illustrated in Fig 1.2. Lines represent the connection between chapters.



Chapter 1: Introduction
Dexterous grasp challenge
Uncertainties of grasping
Deep learning
Chapter 2: Background
Analytical methods
Grasp Learning
Deep learning in grasping
Chapter 3: A Generative-evaluative model for Grasping
Generative model
Evaluative model
Chapter 4: Planning Dexterous Grasps With Gradient Ascent
Gradient-ascent optimization
Chapter 5: Conclusion
Reached goal
Future work

Figure 1.2: Thesis structure with key points.

- **Chapter 2 - Background**

  In this chapter, a review of robotic grasping in terms of analytical methods, learning methods and deep learning methods is presented. The terms "dexterous grasp" and "deep leaning in dexterous grasping" are defined and analysed here.

- **Chapter 3 - A Generative-evaluative model for Grasping**

  In this chapter, two novel evaluative network architectures are presented, and both demonstrate the capabilities of the convolutional neural network (CNN) in predicting the grasp quality of dexterous grasps. First, a method based on [15] is used to generate candidate dexterous grasps in the simulation. Second, the evaluative model based on deep learning methods is trained by grasp-image pairs. Finally, for a novel object in the real-word, candidate grasps are generated by [15], and then the learned evaluative model assesses each grasp and executes the best grasp which has been presented — where each is ranked by the probability of grasp success that the network has predicted for them.

- **Chapter 4 - Planning Dexterous Grasps with Gradient Ascent**

  In this chapter, it is described how, for tuning dexterous grasps, a gradient ascent method is integrated into the generative-evaluative model that was presented in the last chapter. In the course of the evaluation process, the grasp parameters are optimised by finding the hand parameters which maximise the probability of success of a given grasp in relation to a given depth image. This approach is expected to help avoid some of the limitations of the candidate dexterous grasps generated by [15].

- **Chapter 5 - Conclusion, Summary and Discussion**

  In the last chapter, first, the work which has been achieved in this research is summarised. Then, possible directions for future work, along with the deep learning research route, and the prospects for robotic grasping, generally, are discussed.

Chapter 2: Background

Grasping is one of the most commonly encountered problems in robotic manipulation. Napier, in his pioneering work [16], distinguishes power grasps from precision grasps. Power grasps impose a large area of contact between the fingers and a target object so that the object is held firmly in the palm of the hand. In robotic grasping, precise grasps are a necessity when the robot has hands which mimic human hands; such hands usually have more degrees of freedom (DOF) than the parallel gripper.

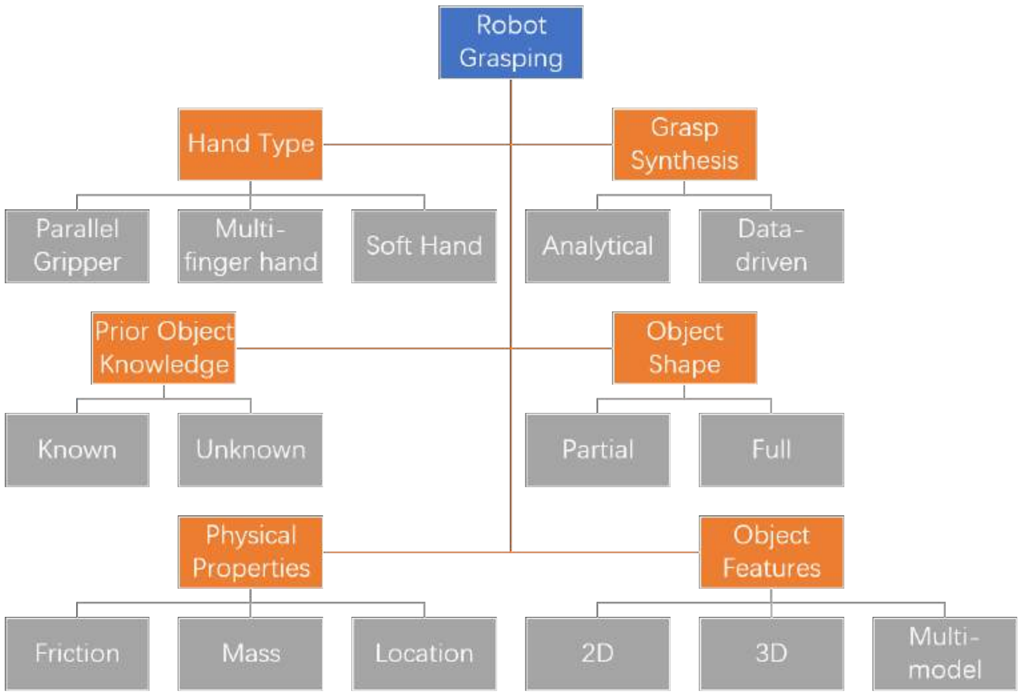


Figure 2.1: Robot grasping. A number of aspects that influence how a grasp is planned for a given object.

As shown in Fig. 2.1, lots of different considerations are involved in the grasping process, such as the robot hand, grasp synthesis method and prior object knowledge. It has emerged that, for the best grasp, all of these aspects need to be considered. However, this is difficult to achieve due to the complexity of these aspects, and include algorithm and hardware limitations. Therefore, research in this area has often addressed only some of these factors.

2.1 Introduction of robot grasping foundations

In this section, the fundamentals of robot grasping are presented. A human grasp could be defined as the making of a seizing motion. In robot grasping, the robotic hand constrains the movement of an object by establishing a set of contacts on the surface of the object and thus resisting any movements of it (such as falling). As a result of the grasp action, the object transitions from an initial steady state to an unstable state, and eventually returns to a stable state in the robot hand, which occurs at the end of the grasp action[17, 18, 19]. For any frictional object, using too small a grasping force will mean that the object may slip, but excessive force should be avoided. In addition, exactly how to make the force act on the object is an important element of a grasp. In order to deal with these issues, a contact model is generated and then used by the grasping system to determine what the forces exerted by the manipulator on the contact areas should be. After the contact model is established, a grasp can be generated based on the grasp analysis methods or grasp synthesis methods. In the next subsection, some basic contact kinematics and the nature of a contact model are described.

Contact Kinematics and model

Looking at the situation where a robot has grasped an object, there are N contact points between the hand and the object as shown in Fig. 2.2. Mason [20] defines a contact as the bond between the finger and the object. When the robot hand‘s finger applies a force f_i at a contact point c_i on an object, there is a generalised force w_i. w_i can be represented by Eq. 2.1 :

$$w_i = \begin{pmatrix} f_i \\ \eta_i \end{pmatrix} = \begin{pmatrix} f_i \\ (c_i - p) \times f_i \end{pmatrix} \tag{2.1}$$

where f_i is the force applied to the object and η_i the resulting moment with force and torque components at the object position p.

In order to keep the object stable when the robot is grasping, all forces w_o are applied to all contact points and all other external forces w_{ex}, such as gravity, taken together, must offset each other, as shown in Eq. 2.2.

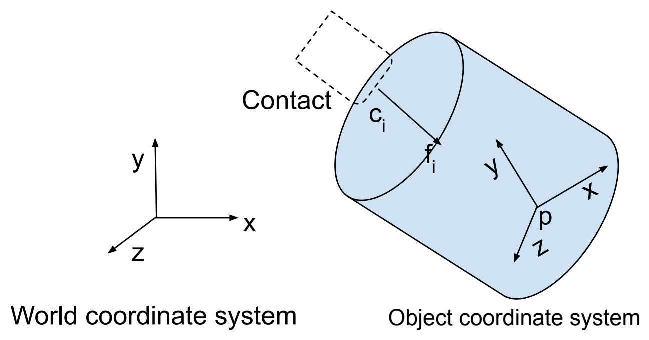


Figure 2.2: An object in contact with a robot finger. p is the object position, c_i is the contact point, f_i is the force applied to the object.

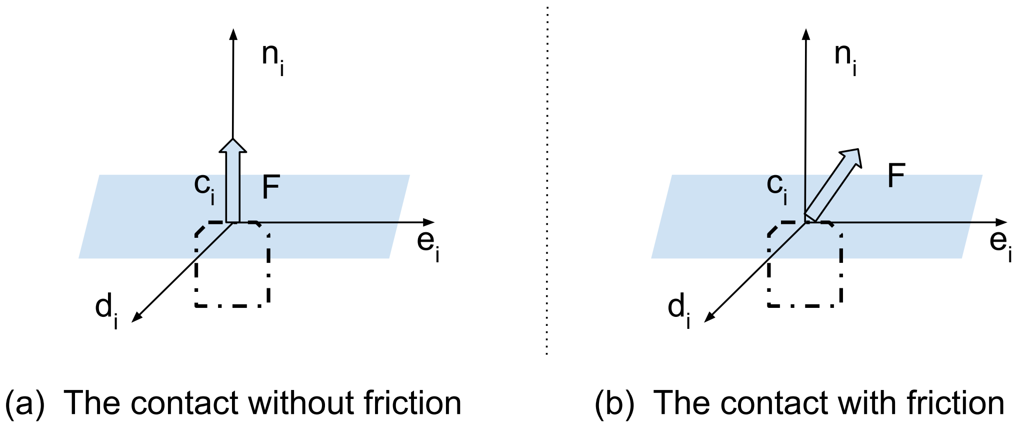
$$w_o + w_{ex} = 0 \tag{2.2}$$

where w_o is a total set of w_i, as shown in Eq. 2.3. n_c is the number of contact points.

$$w_o = \sum_{i=1}^{n_c} w_i \tag{2.3}$$

As described above, the joint between the finger and the object constitutes the contact. The characteristics of the contact vary with the different shapes of the contact surfaces and the friction characteristics of the object. To mirror the different characteristics of contacts, various different kind of contact models are proposed. Contact models are constructed to represent the relationships between the forces exerted by the fingers and the resultant wrenches w_i of the object. In robotic grasping, the most common type of contact model used is the point contact model [21]. Another common type of contact model is the soft finger contact model [22, 23, 24], which is used for soft robotic hands, but this is not within the scope of the present research. The point contact model assumes that the robot hand and the object are the rigid body, and according to whether or not friction is represented at the contact point between the hand and the object, a point contact model can be further classified as a point contact model with friction or a friction-less point contact model, as shown in Fig. 2.3.

Friction-less point contact models are based on an idealised environment wherein the contact surface is very small and the surface frictions of the hand and object are minimal or non-existent. In this case, it is assumed that the contact point (without friction) can only apply a force to the



n_i
c_i
F
e_i
d_i
(a) The contact without friction
(b) The contact with friction

Figure 2.3: Friction and friction-less point contact models in robotics. (a): the contact without friction. (b): the contact with friction. c_i is contact point i. d_i,e_i are two unit vectors that orthogonal and lie in the tangent plane of the contact. n_i is unit normal to the contact tangent plane directed toward the object.

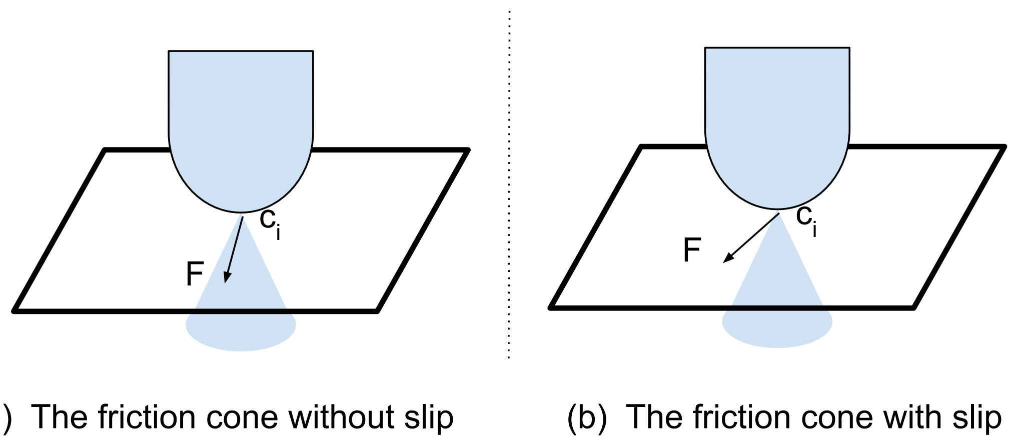
object along the normal direction n_i at the point of contact, as shown in Fig. 2.3. However, this idealised situation can only exist in simulation environments and is far from the situation encountered by robots in the real-world [25, 26, 27]. Due to the lack of robustness of the friction-less point contact model, more studies choose to use point contact models with friction to assist with grasping, especially when the research needs to be tested in the real world.

Point contact models with friction are used to represent contact points which experience friction. In these cases, the contact point with friction can apply a force to the object along the normal direction and also along the tangential direction at the point of contact, as shown in Fig. 2.3. Friction is a complex nonlinear physical phenomenon which occurs between contact surfaces that have relative motion. Many friction models have been proposed [28, 29]. Coulomb friction [30] is a classic and widely used model. It proposes that friction is proportional to the normal force, opposite to the direction of motion and independent of the contact area. As shown in Eq. 2.4.

$$f_t \leqslant \mu f_n \tag{2.4}$$

where μ is coefficient of friction. f_n is normal force. f_t is coulomb friction.

In the point contact model with friction, due to the frictional limit caused by the static coefficient of friction, there will be a total set of forces resembling vertebral bodies called a friction cone, as shown in Fig. 2.4. If the contact force is in the friction cone, the finger



c_i
F
(a) The friction cone without slip
c_i
F
(b) The friction cone with slip

Figure 2.4: Friction cone. (a) A point contact remains in the fixed contact mode while its contact force lies inside the friction cone, (b) A point contact slips while its contact force lies outside the friction cone. c_i is contact point i, F is the contact force.

at the contact point will not slip, as shown in Fig. 2.4. In an ideal case, that is, when the force is applied to the contact point, the friction coefficient is constant. The friction cone can be regarded as a circular cone. Under this condition, the force therein satisfies the following equation 2.5:

$$\sqrt{f_{t_i}^2 + f_{o_i}^2} \leqslant \mu f_{n_i} \tag{2.5}$$

Grasp analysis and grasp synthesis

When the contact model is established, the force or torque applied by the robotic hand on the contact area can be analysed. Most work assumes that the contact is a point contact, and that the large area contacts and line contacts can be considered as a collection of multiple point contacts. Grasp analysis and grasp synthesis are two different ways to obtain a grasp configuration. Grasp analysis methods [31, 32, 33] aim at finding a stable grasp of an object using a set of contacts. Force-closure[34] and form-closure [35] are wildly used. The form-closure and force-closure properties of robotic grasping is the capability of the robot to inhibit the motion of the object with external force applied; these are evaluated with the purpose of making the robot select a stable grasp. In contrast to grasp analysis methods, grasp synthesis [9, 10, 36] usually aims at finding a set of contacts on the object surface and then determining an hand configuration by looking at the geometry of the object.

In Section 2.2, the grasp analysis method is presented in detail. In Section 2.4, data-driven grasp synthesis methods are described and analysed. Following this, the data-driven grasp synthesis method is compared to the grasp analysis method given in Section 2.3.

2.2 Analytical methods for robot grasping

Most early studies of grasping focus on finding the optimal contact points and the force closure [37], which is used to measure grasp quality [34, 37, 38, 39]. These approaches tried to model the physical processes that occur at the surface of the object and the robot hand and can be divided into two phases. In the first phase, approaches try to determine the contacts which allow the robot hand to grasp an object and keep it stable. Most early research focuses on this phase. In the second phase, approaches need to calculate the possible robot hand pose that achieves the grasp. Though the hand kinematics is essential for a grasp planning, it was often solved by crude heuristics. Recently, the hand kinematics are considered [40, 41] when estimating the possible robot hand configuration for the contact points of grasping. For many other analytical approaches for grasp planning, those proposed methods are only tested in a simulated environment, which has an accurate object model and accurate hand kinematics. In practice, the simulation engine always has a big gap compared to the real world, due to the inherent systematic and random errors in robot systems and noisy sensors which make it difficult to find an accurate placement for the fingertips.

When considering robustness in robot grasping, some methods use dynamic simulation to estimate what the robot grasp would be like in the real-world environment [42]. Other methods search for stable graspable regions on the object [43, 44]. Some studies in this direction [45] explore providing robust finger placement by computing Independent Contact Regions (ICRs), as proposed by Nguyen [46]. If the fingertips' positions are in an ICR area, then any grasp should achieve a force-closure. Another line of research towards pre-grasping is the cage method which attempts to forestall inaccurate finger positioning. Rodriguez et al. [47] found that some cages are suitable as waypoints in the process of grasping an object. According to the definition of the process [47], starting from a pre-grasping cage, there is a path leading to a cage that object is caged in robot hand. Once the robot hand is in this configuration, a successful grasp is ensured without the need for more precise finger positioning. These studies solved uncertainty in grasp planning only in a partial manner, and they were usually conducted at the contact level; before these methods were enacted, it was usually necessary to find a reliable hand configuration and grasping points within an ICR, or a pre-grasping cage.

2.3 Analytical methods vs learning methods

Notwithstanding these achievements of the analytical method given in Section 2.2, when analytical methods are applied to real robotic grasping, some significant shortcomings are exposed. First, analytical methods often need some assumptions to be made for grasp analysis [48]. For example, the physical properties of an object, such as friction, mass and location, must be precisely known, and contacts are often simplified as point contacts with friction. But in the real world, not all of the geometric and physical properties of an object can be obtained, and often, even when they are, they cannot be determined precisely. Further, in order to make the calculations easier to handle, fixed contact surfaces and static friction parameters [48, 49] are often used in the analytical methods. Under these assumptions, a given grasp can be analysed, and the grasp can be planned geometrically using the concept of force closure [48]. However, in the real world, these assumptions are either invalid or must be represented by a mass of detail. Moreover, such information may be unavailable or very difficult to obtain with certainty, especially in unstructured environments. Therefore, the planned grasps may end up being unstable or not feasible even though they possess a very high grasp quality in terms of force closure. Particle filters [50] that incorporate tactile sensors have been developed and used to estimate the physical properties of an object. A particle filter can track the object as it moves, even when the object is blocked by the hand. However, this is an expensive and non-universal method and only partially relaxes some of these assumptions. In addition, analytical methods are not good for grasping unknown objects. For an unknown object, analytical methods focus on using various approaches for estimating the shape of an object. Dune et al. [51] proposed one approach to estimate the rough shape of an object by determining the quadric that best approximates its shape, using multi-view measurements; this estimate was used to guide the wrist pose. Marton et al. [52] used a single 3D snapshot to build an object model for use in grasping by fast shape estimation and probabilistic model fitting. The errors in estimating the shape of the object and the assumptions made about the physical properties of the object further make the analytical method unable to adapt to the constantly changing real-world environment. Moreover, the two previously mentioned downsides lead to a third, which is that the performance of the analytical method is much poorer in the real world situation than it is in the virtual environment [53]. This is because most of the precise physical properties and shapes of the objects on which the analytical method relies can only be accurately obtained in simulation.

In contrast to the analytical methods, learning methods don't pay so much attention to the specific parameterisation of the grasp and the precise physical model [54]. The learning methods place more weight on feature representation, similarity metrics, pose estimation and perception. Grasps are then generated, or searched for from samples or knowledge bases, using the resulting data[53]. For example, when the robot hand is a parallel gripper, the grasping problem can be reduced to searching potential positions that fit the gripper's aperture - from a 2D image or 3D point cloud [55, 56, 57]. The learning method also differs in terms of how it evaluates the quality of the grasp and how it transfers the grasp to an object that has not been seen before. Learning methods usually measure grasp quality based on human demonstration, the perception of information or on semantics [53]. In the next section, the detail and analysis of the learning method for grasping are described. Learning-based methods have performed significantly better and more robustly in real-world grasp execution processes, compared to analytical methods.

2.4 Learning methods for grasping

Until the year 2000, the robotic grasping area was dominated by analytical methods, although some researchers had started exploring learning methods in grasp planning [58]. Grasp learning allows robots to use learning methods to learn grasp strategies. As mentioned above, detailed object models and full pose estimation are hard to obtain in real-world for robot grasping. The advantage of the learning methods promises to solve these problems. In 2004, the presence of Graspit! [59] and other simulators promoted the popularity of learning approaches [55, 60, 61, 62, 63, 64]. These studies generated candidate grasps, not from an infinite space of possibilities, but by learning to search for shapes that meet human-made rules between object features and grasp features. Klingbeil et al. [55] observed that if a robot grasp is stable, the contact surface area between the robot hand and the object surface is maximised. Therefore, they searched for an area on the object‘s surface that resembles the shape of the robot hand's interior surface, using range data. Miller et al. [60] associated predefined grasp primitives with shape primitives that represented objects. Fischinger and Vincze [64] emptied a basket filled with objects by the use of their proposed Height Accumulated Features (HAF) method, which calculated grasps related to feature values. By utilising human defined rules or features, these studies work well with grippers, but such rules, which are limited to basic shapes and simple grasp primitives, are often unable to cope with all possible conditions. For example, grasp stability and a large finger contact area are not a sufficient requirement [55]. Thus, these methods are not able to execute the grasp on a set of specific contact locations or generate good candidate grasps for multi-fingered hands. To address these issues, some studies have explored learning the mapping from visual features to grasp parameters. Saxena et al. [65] learned features from a 2D image and a 3D point cloud to calculate grasp features and apply them to a supervised learning algorithm with a three-fingered 4-DoF hand. Another approach, proposed by Saxena et al. [66], pushed learning methods for grasp planning even further. A logistic regressor was trained to predict suitable points on a target object for grasping, based on synthetic, labelled data. No analytic principles were used. Kroemer et al. [67] proposed a non-parametric representation of an object‘s surface for use in calculating the similarity between two subparts of objects; this allows the robot to learn the mapping from the point clouds to hand motor actions directly. Kopicki et al. [15] proposed a method which allowed learning of a generative model of the whole grasping process, including contact models and hand configuration models between object features and fingers. Thus, mapping from local 3D shaped features to the robot hand fingers and its pose was established. They showed the method can be generalised to new objects for a given grasp type.

Instead of generating grasp hypotheses by learning the rules directly, a different method is to transfer the grasp from human demonstration to the robot. This is called imitation learning. The correspondence problem is the main issue in imitation learning [31]. For example, in terms of kinematic structures, due to the difference between human and robotic hands, no direct connection can be made between the human hands and the robot hands. Another method is not to learn a direct mapping between the human hand and the robot hand but to simplify these by learning the grasp primitives or intentions from human demonstration [68, 69]. Ekvall and Kragic [70] used a magnetic tracking device to collect the relevant information from grasp demonstrations in which humans grasped a target object. These human demonstrations were used to model a set of hand pre-shapes with respect to objects. The object and grasp types were first recognised by the system and then performed through a fixed schema using a mapping. In summary, these methods demonstrate the power of the learning methods in relation to generating good quality candidate grasps, not only for low DoF hands [55, 60, 64] but also for high DoF hands [15, 65].

For a single object, there are a number of different ways to grasp it. Therefore, it is important to evaluate the quality of a grasp. When candidate grasps are generated, each grasp needs to be evaluated in order to obtain a predicted grasp quality measure. For candidate grasp ranking, some methods use mechanically informed reasoning based on analytic formulations. For example, the ϵ-metric method proposed by Ferrari and Canny [71]. Saxena et al. [65] set up a probabilistic model containing a classifier that predicted the probability that a grasp will succeed by using the hand configuration and a point cloud. Pelossof et al. [72] set a regression mapping between object, hand configuration and grasp quality by the Support Vector Machine(SVM). Detry et al. [32] proposed object grasp affordances which were represented by grasp densities. They assessed whether a given relative object-robot hand configuration will produce a stable grasp or not.

2.5 Comparison of deep learning methods in grasping with previous research

Recently, a number of dexterous grasp methods have achieved state of the art performance based on deep learning, such as object detection [73, 74], natural language processing [75], etc. Deep learning methods have the advantage of using multiple layers to learn representations of data with multiple levels of abstraction [76]. Further, deep learning can handle large datasets through back-propagation algorithms and the extracting of useful abstract information from them. Due to these advantages, deep learning is now widely used in robotic grasping [77]. As a branch of learning-method, deep learning solves some of the shortcomings in previous learning-based grasping methods. It is a good supplement to some established learning methods and frameworks and has achieved some improvements[4, 5, 6, 7, 8, 33, 36, 78].

As discussed in Section 2.4, when applying a learning-based method to grasping, there are still some bottlenecks. First, human-made rules will not satisfy all the situations encountered involving the different types of hands and objects. Second, some studies design hand-engineered feature vectors from sensor data using human experts, which is cumbersome and time-consuming. Third, though some methods have been proposed for evaluating grasp, there is still room for improvement. Fourth, although the learning-based method reduces the gap between the simulated environments and the real-world, transferring grasps from simulation to the real world is still a challenge.

Deep learning methods first demonstrated their advantages when they were applied to evaluate grasp quality [4, 5, 6, 7, 8, 33]. The ways used to evaluate grasp quality in learning-based methods, such as the ϵ-metric method proposed by Ferrari [71], are often only suitable for a part of the object-grasp pairs due to their inherent limitations. Deep learning methods have considerable advantages when used for classification through training by labelled data [79]. Studies have either focused on evaluating the quality of grasps [10, 12, 14] or on evaluating the quality of contact points for grasping [5, 8]. Deep learning methods have also demonstrated advantages in relation to transferring grasps from the simulation to the real world. For example, Bousmalis et al. [80] proposed a method called GraspGAN. Using synthetic data and domain adaptation and a combination of simulated data and real-world samples, these researchers reduced the number of real-world samples needed by up to 50 times. Similar work was undertaken by Zhang et al. [81], Fang et al. [82]. With the development of deep learning and the drive of robotics, some novel end-to-end grasping frameworks and methods based on deep learning have been proposed. These studies aim to generate grasp parameters directly from deep neural networks [36, 78]. Deep learning has been rapidly adopted in robotics and has revolutionised many aspects of robot grasping. In the next section, the detail and analysis of deep learning methods for grasping are described.

2.6 Deep learning in grasping

Deep learning methods for grasping can be divided into two categories, based on whether the deep learning method is used to evaluate grasps or to generate grasps. One class predicts the success of the grasp by learning a mapping from grasp to grasp quality, which uses a point cloud or an image from the object [4, 5, 6, 7, 8, 33]. The other class directly generates a hand configuration for performing a grasp, using a regression network with an image [36, 78]. Lenz et al. [4] first applied deep learning to robot grasping with a two-fingered parallel gripper. Given an image of an object, a two-stage neural network that is trained on the Cornell grasping dataset was used to evaluate the best grasp position on the object. In the first stage, a small neural network was used to detect several positions that could be used for grasping. In the second stage, a large neural network was used to evaluate each candidate position. Thus, a single optimal grasp was produced for each given object. Pinto and Gupta [5] further simplified the evaluation of grasp positions by evaluating the grasp orientation for each given object. Their proposed system used up to 50,000 grasps on a real robot as data to train the required neural network. One shortcoming was that this required calibration between the robot and the camera manually during training. Similar work was proposed by Levine et al. [6]. These researchers attempted to grasp objects from a tray of objects; such a scene is more complex and therefore generates a larger amount of data (More than 800,000 grasps executed by 14 real robot arms over two months).

One drawback of these methods is that the manpower and material resources used to collect training data from real-world robots far exceeds the data-collection requirements of the other methods (i.e., those using simulations). However, for the grasp problem, using a simulation environment instead of experiments in the real world will not work in most cases. At present, the best simulation engine still exhibits a large gap between its own representations and the situations in the real-world which it is designed to simulate. The longer the simulation time, the more obvious the difference becomes. This leads to algorithms that work perfectly in simulations, but which deviate markedly from correct working in real-world experiments. To address this issue, Mahler et al. [8] made two contributions which reduced dependence on physical experiments. First, they manually introduced modelled error into their simulations in order to make the data fit the real-world situation more efficiently. Second, their neural network only needs to perform an evaluation function to score a given grasp position. Thus, their method can be divided into three phases. First, pick grasp positions randomly from the object point cloud. Second, according to a trained neural network, score each grasp position. Third, select the highest grasp score position in order to execute it. Instead of evaluating the grasp position, some studies have explored predicting the best grasp configuration for the object directly [36, 78]. One common step in the systems proposed by these studies is to use a deep convolutional neural network (CNN) to extract features and then pass these through fully connected layers in order to regress the grasp parameters.

2.6.1 Deep learning in dexterous grasping

Introduction of dexterous grasping

Dexterous grasping or multi-finger grasping is a research area of robotic manipulation. The main differences between dexterous grasping and the parallel gripper grasping that has been widely used in industry are the degrees of freedom of the robot hand and its means of grasping. As shown in Fig 2.5, the gripper‘s finger is a rigid component with one degree of freedom.

These characteristics of the parallel gripper determine that the grasping method is unique, that is, it first selects the potential position on the object that can fit within the space between two fingers and then grasps the object by the closure of these two jointless fingers. In contrast, a large

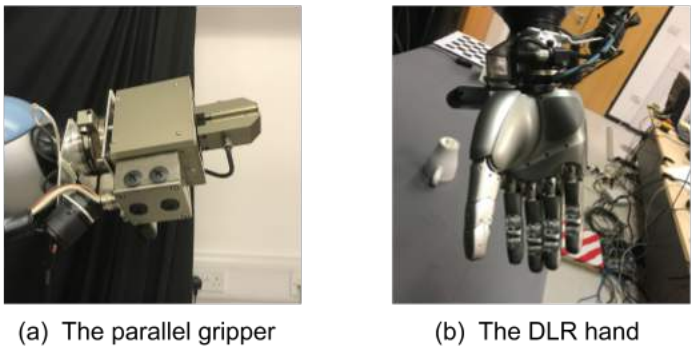


Figure 2.5: Robot hand example. (a): A parallel gripper. (b): The DLR HAND used in the experiment of this thesis.

number of robotic hands designed for dexterous grasping are anthropomorphic in design, as shown in Fig 2.5. Thus, for the same grasp task, as shown in Fig 2.6, dexterous grasping exhibits a higher level of complexity. In a typical dexterous grasp task, the core issue is to handle the relationship between the fingertip joints, the contacts, and the object in the hand-object system. Dexterous grasping is based on the object to be grasped to determine how each finger should behave and what force should be applied to the object. Dexterous grasping requires precise control of force and movement, which the simple robotic gripper does not provide. Specialised hands, usually with multi-jointed fingers, must be used.

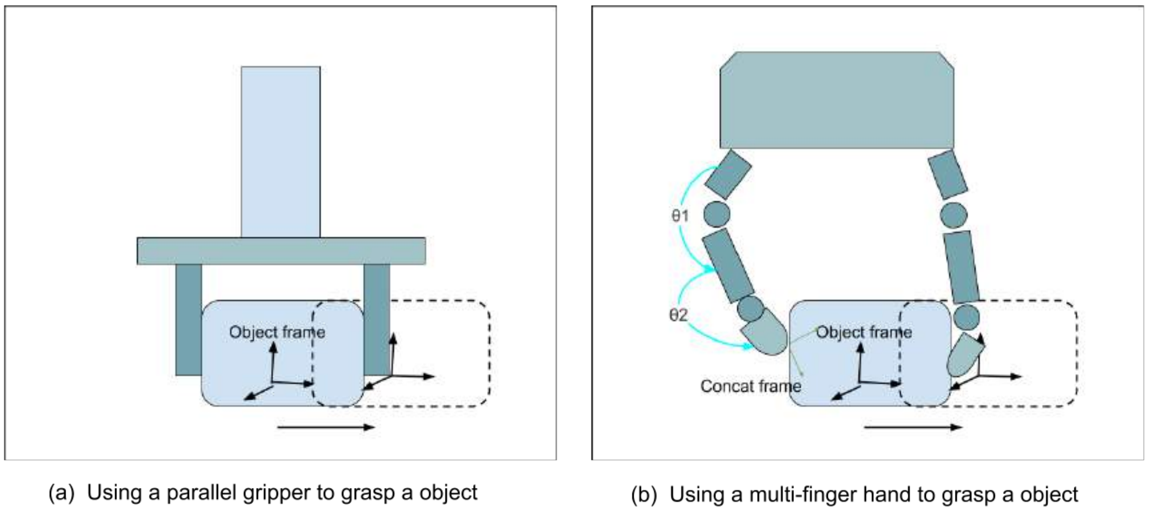

Object frame
θ1
θ2
Object frame
Concat frame
(a) Using a parallel gripper to grasp a object
(b) Using a multi-finger hand to grasp a object

Figure 2.6: A grasping task: grasping an object to another position handled by a parallel gripper(a) and multi-finger hand(b).

Dexterous grasping with deep learning methods

In the studies previously discussed, these deep learning approaches for grasping have shown an impressive performance for the parallel gripper, while a small number of papers have explored deep learning as a method for the more difficult problem of dexterous grasping [10, 11, 12, 13, 14]. All of these methods use simulation to generate training data for learning. Kappler et al. [10] suggest the ability of CNN to learn the mapping from dexterous grasps to grasp quality. They created a template, which is a feature representation of the local object shape from the whole point cloud of the object, and a trained network to predict if a given palm pose applied at this template will succeed. While only a number of sampled grasp poses are changed with a fixed hand pre-shape for an object. Varley et al. [12] push this further by predicting grasp quality under various hand pre-shapes from only a single RGB-D image for dexterous grasps. Similar works are also proposed by Zhou and Hauser [14] and Lu et al. [11]. Lu et al. [11] not only learned an evaluative model but also performs grasps directly by optimising the wrist pose and hand pre-shape through the learned network via gradient descent. In contrast, Veres et al. [13] presented a conditional variational auto-encoder (CVAE) regression network to predict the contact locations and contact normals for a dexterous grasp with an RGBD image of an object. However, the assumption of this regression approach is that for a given RGBD image of an object view, there exists a unique best grasp, and this assumption is not always correct.

As shown in Table 2.1, despite all the progress which has been made, generalising learned grasps across a large number of different hands and objects under various task constraints is still, to a large extent, somewhat unknown. The current method is far less than human prediction and grasp capability. The critical challenges towards efficient robust grasping are: (1) perception, how to use a general predictive model of learning environment evolution and its response to hand is an open problem worthy of exploration; (2) feedback, for example, when you have already grasped a cup, and someone adds water to that cup, your hand will gradually increase its grip in order to balance the weight without using excessive force to crush the cup this process is very delicate and difficult to simulate in robot system, so it is essential to build a model from previous grasp experience which can select proper corrective actions based on perception (visual perception, tactile perception, etc.); (3) active learning, the goal of active learning is to

Table 2.1: Approaches for Grasping

	Prior Object Knowledge		Multi-Fingered Hand	Object Features			Grasp Synthesis				Data	
	Known /Familiar	Unknown		2D	3D	Multi -model	Heuristic	Demonstration	Trial Trial	Labeled Data	Simulation	Real -Data
Kamon et al. [58]	Y			Y					Y			Y
Miller et al. [60]	Y		Y		Y		Y				Y	
Goldfeder et al. [61]	Y		Y		Y		Y				Y	
Borst et al. [62]	Y		Y		Y		Y				Y	
Ciocarlie and Allen [63]	Y		Y		Y		Y				Y	
Fischinger and Vincze [64]	Y				Y					Y		Y
Klingbeil et al. [55]		Y			Y		Y					Y
Saxena et al. [65]		Y			Y		Y					Y
Kroemer et al. [67]	Y		Y		Y			Y	Y			Y
Kopicki et al. [15]		Y	Y	Y				Y				Y
Ekvall and Kragic [70]	Y		Y			Y		Y				Y
Pelossof et al. [72]	Y		Y		Y					Y	Y	
Saxena et al. [66]	Y			Y						Y		Y
Zhou and Hauser [14]		Y	Y	Y						Y	Y	
Kumra and Kanan [36]		Y		Y						Y		Y
Veres et al. [13]		Y	Y	Y						Y	Y	
Varley et al. [12]		Y	Y	Y						Y		Y
Lu et al. [11]		Y	Y	Y			Y					Y
Mahler et al. [8]		Y		Y						Y		Y
Gualtieri et al. [33]		Y		Y						Y		Y
Johns et al. [7]		Y		Y						Y		Y
Levine et al. [6]		Y		Y						Y		Y
Pinto and Gupta [5]		Y		Y						Y		Y
Lenz et al. [4]		Y			Y					Y		Y

build a method which enables robots to use previously learned knowledge and transfer it to new contexts.

To contribute to the resolution of these issues, this research in this thesis develops several networks for evaluating dexterous grasps with a learning-based generative approach. By taking advantage of the network prediction, which is the result of a robust training process to bridge the evaluation gap left by the generative approach, the robot hand is able to adapt adequately to various uncertainties.

Chapter 3: A Generative-evaluative model for Grasping

3.1 Introduction

In this chapter, an evaluative model for evaluating dexterous grasping is presented and a dexterous grasps generator for this evaluative model is described. The approach, shown in Fig 3.1, consists of:

1. Grasp generator based on Kopicki et al. [15] which is learned from 10 demonstrations and then used to generate candidate dexterous grasps for training and testing deep network.
2. A deep neural network is used for the evaluative model. After training, the learned evaluative model evaluates each candidate grasps that are generated by the generator and executes the top one, which is ranked by the probability of grasp success as predicted by the network.
3. For collecting enough data quickly for training the evaluative model, the grasp generator is used to generate grasps and collect data in the simulation. In order to make the network which was trained using virtual data applicable to reality, some measures, as described in Section 3.5.1, are taken to reduce the gap between simulation and reality.

Following the grasping process, the generative model is described in Section 3.2. The evaluative model is presented in Section 3.3 and the training process is described in Section 3.4. The dataset and experiment results analysis is shown in Section 3.5.

3.2 Generative model for grasp generator

In this section, the generative model is briefly described. Though this part is not the contribution of this thesis, it is included to keep the manuscript self-sufficient. The symbols and expressions

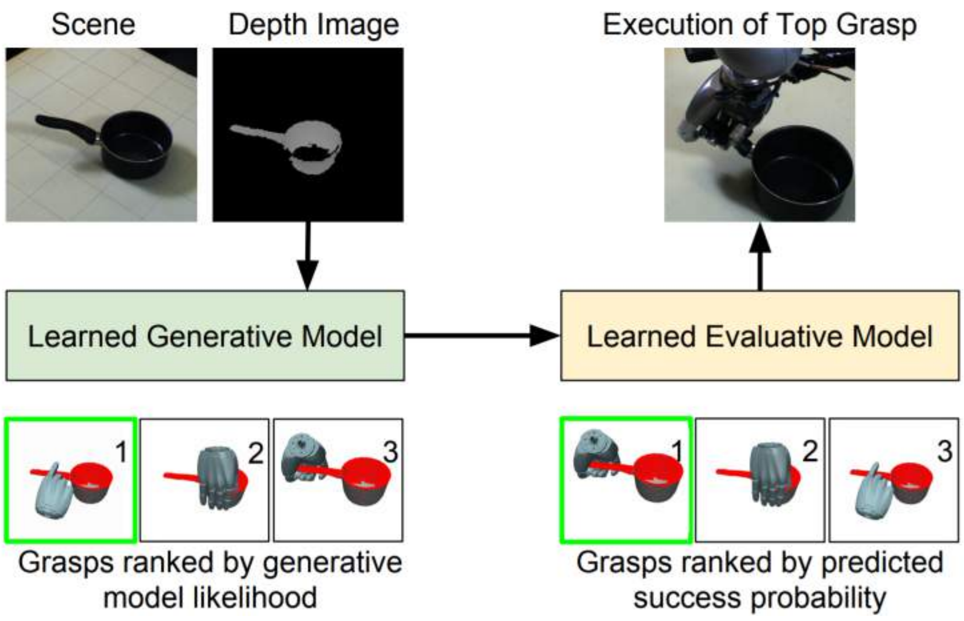


Figure 3.1: The overall process when presented with a novel object.

here follow the original paper by Kopicki et al. [15]. A more detailed description is included in [15].

In addition to the quality of grasps of the method itself, this method is chosen as a grasp generator for two reasons: (1) this thesis is able to be compared with Kopicki et al. [15] in real-world experiments by using the same objects and environment. (2) Though some studies have shown the ability of Convolutional Neural Networks (CNN) to predict grasp quality for multi-fingered grasps, it is worth exploring if the deep network can also achieve a better performance based on a learning method that has achieved decent performance already. The generative model builds on the paper of [15]. First, the model needs to learn from the demonstration. For this thesis, the ten demonstration grasps are provided as shown in Fig. 3.2. Then it needs to transfer to new objects. Finally, candidate grasps are generated by the learned generative model.

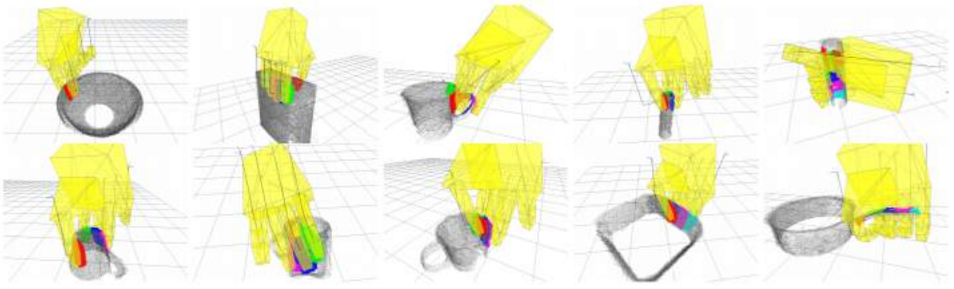
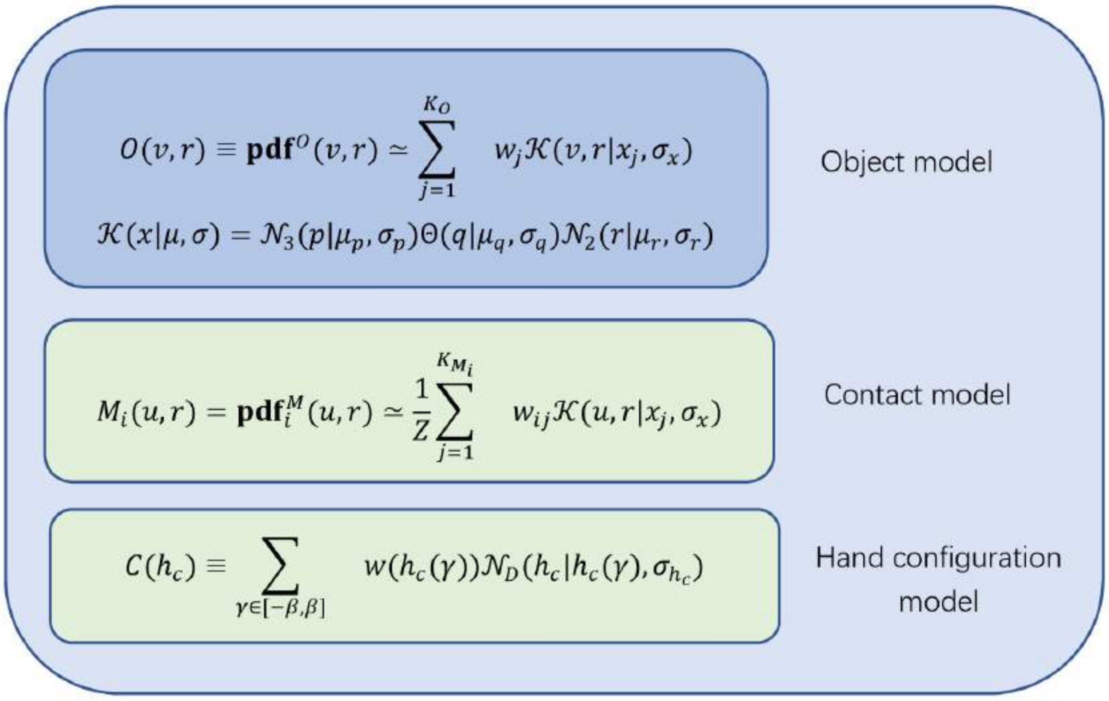


Figure 3.2: The ten training grasps for the generative model.

A. Model learning



$O(v,r) \equiv \mathbf{pdf}^{O}(v,r) \simeq \sum_{j=1}^{K_O} w_j \mathcal{K}(v,r|x_j,\sigma_x)$
$\mathcal{K}(x|\mu,\sigma) = \mathcal{N}_3(p|\mu_p,\sigma_p)\Theta(q|\mu_q,\sigma_q)\mathcal{N}_2(r|\mu_r,\sigma_r)$
Object model
$M_i(u,r) = \mathbf{pdf}_i^{M}(u,r) \simeq \frac{1}{Z}\sum_{j=1}^{K_{M_i}} w_{ij}\mathcal{K}(u,r|x_j,\sigma_x)$
Contact model
$C(h_c) \equiv \sum_{\gamma\in[-\beta,\beta]} w(h_c(\gamma))\mathcal{N}_D(h_c|h_c(\gamma),\sigma_{h_c})$
Hand configuration model

Figure 3.3: The model learning process (The symbols and expressions here follow the original paper by Kopicki et al. [15]). (1) O is short for $\mathbf{pdf}^{O}$, bandwidth $\sigma_x = (\sigma_p, \sigma_q, \sigma_r)$, the set of features x_j in the object model is K_O, all weights are equal $w_j = 1/K_O$, μ represents the kernel mean point, and the kernel bandwidth is σ, $\mathcal{N}_n$ is an n-variate isotropic Gaussian kernel. (2)u is the pose of L_i relative to the pose v_j of the j^{th} surface feature, K_{M_i} is the number of surface features in the neighbourhood of link L_i, Z is the normalising constant, and w_{ij} is a weight that falls off exponentially as the distance between the feature x_j. (3)γ is a parameter that interpolates between the beginning (h_c^t) and end (h_c^g) points on the trajectory,

In the model learning step, as shown in Fig. 3.3, there are three models shown: object model, contact model, and hand configuration model. For an object, a depth camera is used to generate point cloud, where the object model contains the surface features from a given point cloud. When a grasp is in the demonstration, the final hand pose is used for contact model M_i to build a relationship between surface feature x_j of contact model and the pose of each finger link L. The hand configuration model encodes a set of configurations of the hand joints to make the generated grasp similar to the training demonstration in order to improve the reliability of the grasp.

B. Grasp transfer and generation

For grasping a novel object, the learned model must be able to apply to new object. For a new object, the contact models learned in the last section need be transferred to the new object. A partial point cloud of the new object is acquired and recast as a density, O_{new}. The transfer of each contact model M_i is achieved by convolving M_i with O_{new}. The Monte-Carlo procedure produces sampled poses to link L_i with the new object. The j^{th} sample is $\hat{s}_{ij} = (\hat{p}_{ij}, \hat{q}_{ij})$. For each sample, $\hat{s}_{ij}$ the Monte-Carlo procedure produces a weight w_{ij} equal to its likelihood. These samples are used to build what is termed the query density estimate:

$$Q_i(s) \simeq \sum_{j=1}^{K_{Q_i}} w_{ij} \mathcal{N}_3(p|\hat{p}_{ij}, \sigma_p)\Theta(q|\hat{q}_{ij}, \sigma_q) \tag{3.1}$$

where all the weights are normalised, $\sum_j w_{ij} = 1$. Every contact model and the new object construct with a query density. Then, these query densities are used to generate grasps with the hand configuration model.

In Kopicki et al. [15], candidate grasps can be generated by the following steps. First, a query density k is selected and a sample $s_k \sim Q_k$ is taken. Following this, a sample $h_c \sim C$ from the hand configuration model is taken. This pair of samples together defines, via the hand kinematics, a complete grasp $h = (h_w, h_c)$, where h_w is the pose of the wrist and h_c is the configuration of the hand. Finally, the initial grasp is then improved by stochastic hill-climbing on a product of experts:

$$\underset{(h_w, h_c)}{\operatorname{argmax}}\, C(h_c) \prod_{Q_i \in Q} Q_i \left(k_i^{\text{for}}(h_w, h_c)\right) \tag{3.2}$$

This generates an process which can be run many times, thus enabling the generation of many candidate grasps. After that, in Kopicki et al. [15], they choose the top grasp in all likelihood to execute, according to the learned models. For the training data of the evaluative model, many grasps were generated by repeating this process.

3.3 The evaluative model

The grasp generator has been described in the last section. The grasp generator learns by ten real grasps from human demonstration. The generator uses only a single view depth image as an input to generate a number of plausible candidate grasps. While a single view depth image leads to partial object shape, to a large extent, the grasp generator only considers the features from the hand model and the partial object shape. As discussed in Sec. 2.3, the success of a grasp depends on factors such as the object shape, mass and friction etc. The grasp generator model is not able to learn the impact of these factors on the grasping. Thus, the evaluative model is used to indirectly learn the impact of these factors. The evaluative model has two inputs: a colourised depth image of the object, and the sequence of wrist poses and hand configurations for a grasp, expressed in the camera frame. The evaluative model outputs a grasp success probability for image-grasp pair. After the grasp generator has learned, it is used to generate grasps for the evaluative model. In this section, which is the main work of this thesis, two novel evaluative architectures EM1 and EM2 are proposed. The overall process of the two frameworks is similar and shown in Fig 3.4.

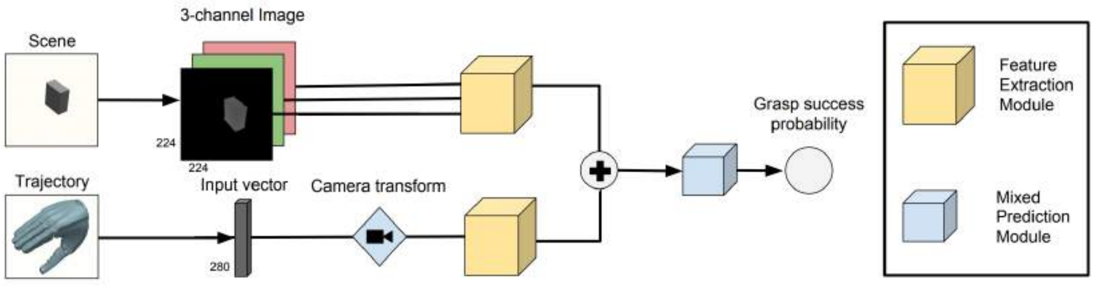


Figure 3.4: The overall evaluative model process

Overall, the evaluative model is a deep neural network, it takes hand parameters and depth image as network inputs and predicts the grasp success probability. Both inputs are passed through their Feature Extraction Module (FEM) to get abstract features respectively. The pre-trained VGG-16 network is used for image input, and various other network details will be described in the following paragraph. Features from two inputs are combined, which generates a concatenation of the hand parameters and depth image. Then it passes through the Mixed Prediction Module (MPM) to get the final grasp success probability.

The FEM for image branch extracting features from different poses of the thermos cup, as shown in Fig. 3.5. As shown in Fig. 3.6, the FEM for hand branch is shown extracting features from different grasp parameters. The red dot representing the successful grasp and the green fork representing the failed grasp are initially mixed (left coordinate system), and it is difficult for classification. In the middle coordinate system of Fig. 3.6, after the processing of

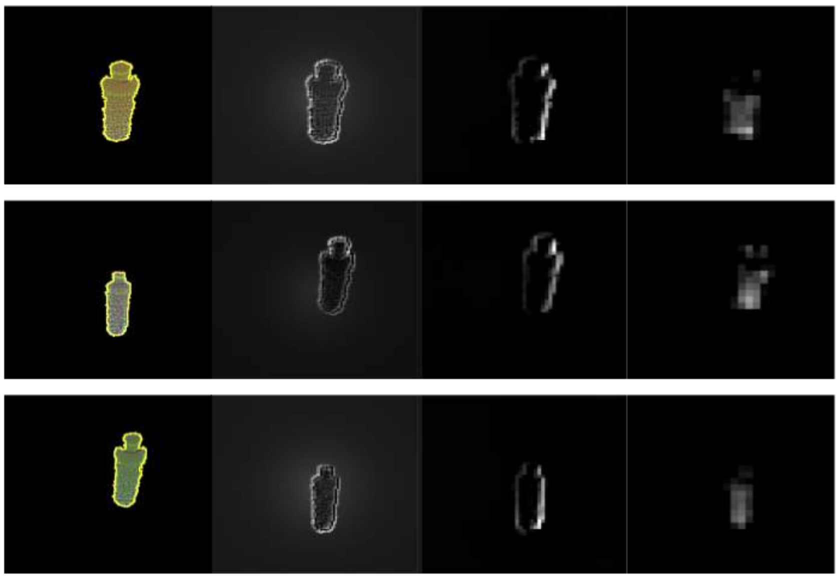


Figure 3.5: Convolutional network inside of evaluative model. First column: colorized depth images (Each channel is normalized in 0-255, the three channels are depth image, mean curvature and Gaussian curvature, respectively.). Second column: feature maps from the early convolutional layer. Third column: feature maps from the middle convolutional layer. Fourth column: feature maps from the late convolutional layer.

the FEM of hand branch, the distance between the data points of successful grasp and the failed grasp is increased, which means that the characteristics of the data are more abundant, and the difficulty of classification is reduced. The right coordinate system in Fig. 3.6 shows the vector processed by MPM. When the information of the two branches overlaps and is processed by the MPM, the concentration of different categories is significantly improved, resulting in the precise prediction of grasp outcome. The grasp-object pair data provided to the evaluative network is collected from the random combination of the parameters (object pose, mass and friction etc.). The function of the FEM and the MPM make learning these different combinations possible. Further, the network is supervised with the grasp result signal (success or failure). In the following narrative, this thesis shows how to learn a robust evaluative model for grasps across different conditions.

3.3.1 Preprocessing for depth image and hand trajectory

As mentioned above, the evaluative model takes a 3-channel image to the pre-trained VGG-16 network. While the initial depth image is a single channel, depth image colourisation is employed. The depth image colorization takes a 1-channel depth image I_t^{depth} and output a

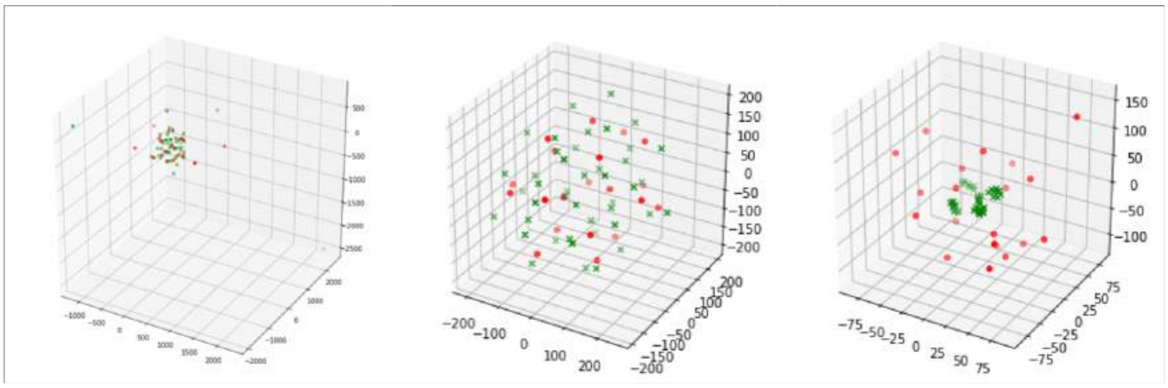


Figure 3.6: How neural networks classify the grasp visualized by PCA (each axis represents one principal component.). The red dot represents the successful grasp, the green cross represents the failed grasp. Left: origin hand parameters. Middle: the final output vector from hand parameter branch. Right: the last fully connected layer of MPM which combines the output of images branch and hand parameters branch.

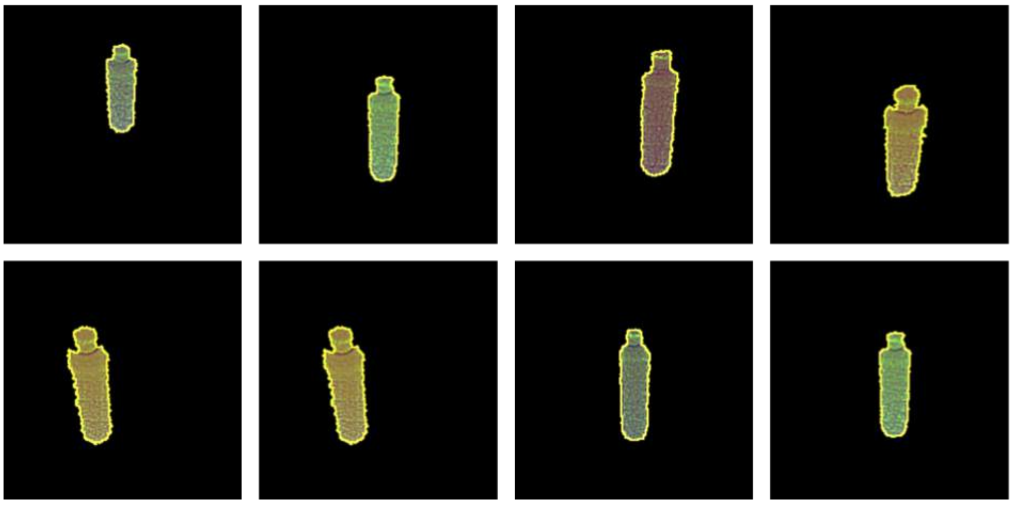


Figure 3.7: Colourised depth images. (Each channel is normalized in 0-255, the three channels are depth image, mean curvature and Gaussian curvature, respectively.)

3-channel image I_t. First, the 640×480 depth image is cropped by a square window of size 460×460 and down-sampled to an image of size 224×224. The colourisation process creates a 224×224 3-channel image, the first channel is mean curvature and the second channel is Gaussian curvature calculated by [83]. The formula of mean curvature is shown in Equation 3.3:

$$h = gr_{xx} + gr_{yy} \tag{3.3}$$

where gr_{xx} is the second gradient in the horizontal direction in a 1×3 window, and gr_{yy} is its vertical counterpart. Similarly, Gaussian curvature is calculated by Equation 3.4

$$k = gr_{xx} \times gr_{yy} - (gr_{xy})^2 \tag{3.4}$$

where gr_{xy} is the second-order gradients with respect to both directions.

After a colourised depth image is generated, each channel of an image is processed by a pixel level zero-mean standardisation as shown in Equation 3.5.

$$new\_I_{ch}(i, j) = (I_{ch}(i, j) - I_{ch\_mean}(i, j))/I_{ch\_std}(i, j) \tag{3.5}$$

where ch represents one of the three RGB channels in colourised depth image. i,j are the coordinates on the image, $new\_I_{ch}(i, j)$ is the result of point i(0-243), j(0-243) of channel ch after processing. $I_{ch}(i, j)$ is origin point of channel ch in colourised depth image, and $I_{mean}(i, j)$ and $I_{std}(i, j)$ are average and variance of channel ch across all the colourised depth images in the training set at the i, j point, respectively. The same prepossessing method is applied to the hand parameters and is also shown in Equation 3.6.

$$new\_h(i) = (h(i) - h_{mean}(i))/h_{std}(i) \tag{3.6}$$

where i is the index of hand parameters, $new\_h(i)$ is the result of point i (0-279) after processing. $h(i)$ is origin value of hand parameter, and $h_{mean}(i)$ and $h_{std}(i)$ are average and variance of across all the hand parameters in the training set at the i point, respectively.

3.3.2 Evaluative model EM1

The first network architecture is based on Levine et al. [6], but it is trained on a single depth image and hand parameters instead of motor commands. The network takes the colourised image as input. The input image is processed by 16 convolutional layers from the pre-trained VGG-16 network, following a 1024 fully connected layer. After this, the hand trajectory is provided as an input to the network. Hand trajectory comprises ten way-points, each of 28 dimensions (20 for the finger joint angles, 1 for the grasp type, and 7 for the wrist pose), giving a total of 280. This 280 vector is transformed to a 1024 vector by using a 1024 fully connected layer. The outputs of these two fully connected layers are added. After this concatenation, a further four fully connected layers are applied, as shown in Fig 3.8, followed by a Softmax layer. This results in the network output of the probability of success for a given grasp with depth image of the object.

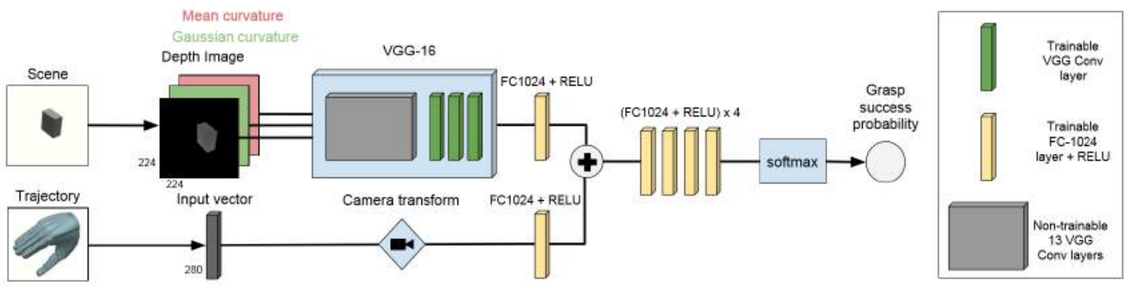


Figure 3.8: EM1 architecture: The first 13 layers of the VGG network have been frozen. The network output is the predicted probability of grasp success.

3.3.3 Evaluative model EM2

As an improvement to the EM1, a more complicated and effective network architecture $f(\cdot)$ as shown in Fig 3.9 is proposed. The feature extraction module of image branch is reserved. Inspired by the VGG-16 network, a new feature extraction module for hand trajectory is proposed. I also redesigned the Mixed Prediction Module with more effective architecture by discarding unnecessary layers.

The modified network details are shown in Table 3.1. In EM1 configuration, the hand trajectory is provided as an input to the network with only a single 1024 fully connected layer to transform hand parameters and then add to the output of feature extraction module of image branch. For the new feature extraction module for hand trajectory in EM2, hand trajectory will pass through some customised convolutional layers. EM2 also discards full connected layers in both feature extraction modules. Instead, EM2 uses Global Average Pooling which directly obtains feature vectors from feature maps of the final convolutional layer. Global Average Pooling not only reduces the amount of computation, but also provides a global feature for the prediction network. Then, the output of GAP operation is a 512 length vector, after that two

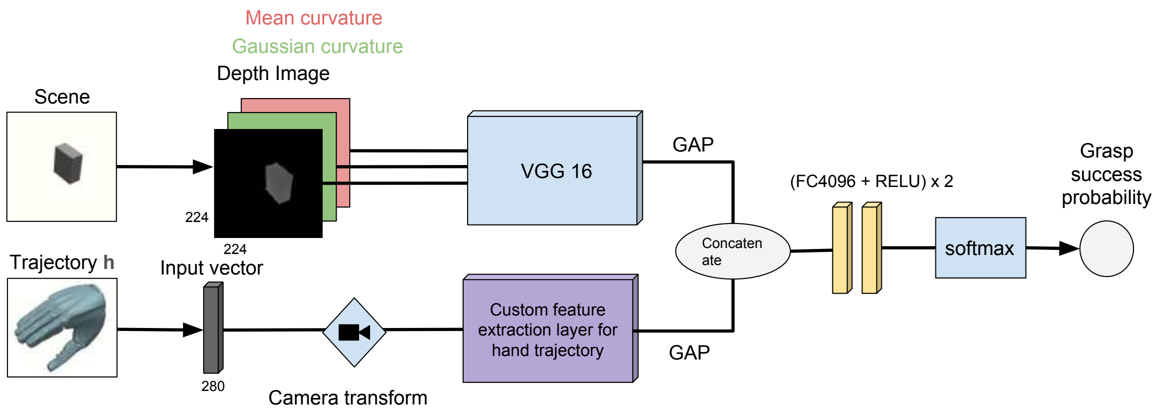


Figure 3.9: EM2 architecture. The output of the feature extraction module of image branch and hand branch are concatenate. They are then passed through 2 fully connected layers. The detail of the custom feature extraction layer is shown in 3.1.

512 length vectors are concatenated. EM2 reduces the number of fully connected layers from four to two, and increases the number of neurons per layer from 1024 to 2048. Thus, EM2 have reduced the total of six fully connected layers, which reduces the amount of calculation and avoids over-fitting to some extent.

3.4 Training

Both networks have the same training process. The network outputs a discrete probability distribution, $p = (p_0, p_1)$, over 2 categories. In one network, the two categories are success grasps and failed grasps. As usual, p is computed by SoftMax over the outputs of fully connected layers. For training the network, the cross-entropy loss is used, as shown in Equation 3.7.

$$H_{y'}(y) := -\sum_i (y'_i \log(y_i) + (1 - y'_i)\log(1 - y_i)) \tag{3.7}$$

where $y = f(I_i, h_i)$ is the predicted grasp success of grasp h_i, and I_i is the associated colourised depth image of grasp h_i. and y_i defines the ground truth grasp label as 1 if successful and 0 if failed. SGD is used as optimiser with mini-batches of size 64 for 300,000 iterations and learning rate is 0.001. Dropout is applied after each fully connected layer to prevent over-fitting, thus keeping weights with a probability of 0.5 and apply Xavier initialisation [84] for the weights. TensorFlow (http://tensorflow.org/.) is used to implement and train all of neural

Table 3.1: EM1 and EM2 network architecture (shown in columns). The convolutional layer parameters are denoted as <Number of layers> conv <number of channels> <receptive field size>. The fully connected layer parameters are denoted as <Number of layers> FC <number of channels>. The ReLU activation function is not shown for brevity. The input 1 is 224×224 colourised depth image, the input 2 is 280×1 hand parameters. GAP means Global Average Pooling.

Network Architecture				
EM2 Network Configuration			EM1 Network Configuration	
input 2	input 1			input 1
2*conv-64-3*1	2*conv-64			2*conv-64
maxpool	maxpool			maxpool
2*conv-128-3*1	2*conv-128			2*conv-128
maxpool	maxpool			maxpool
3*conv-256-3*1	3*conv-256			3*conv-256
maxpool	maxpool			maxpool
3*conv-512-3*1	3*conv-512			3*conv-512
maxpool	maxpool			maxpool
3*conv-512-3*1	3*conv-512			3*conv-512
GAP	GAP		input 2	maxpool
Concatenate			FC-1024	FC-1024
2*FC-4096			Add	
			4*FC-1024	
Softmax			Softmax	
Output (The grasp success probability)			Output (The grasp success probability)	

network models .

Additionally, Exponential Moving Average (EMA) is used with decay 0.999 to make the evaluative model more robust and effective. When training a model, the moving averages of the evaluative network parameters are maintained. During the evaluation process, the network that used averaged parameters showed significantly better results, compared to the network that used final trained values. Exponential decay is used to compute the moving averages. The shadow variables have the same initial values as the trained variables. When training the network and maintaining the moving averages, each shadow variable is updated by the formula 3.8:

$$s\_v = decay \times s\_v + (1 - decay) \times v \tag{3.8}$$

where s_v is shadow variable and $_v$ is variable.

3.5 Experiment

3.5.1 Dataset

In this section, the composition of the dataset and dataset collection processes are described. The generative model is worked with a simulated depth image of containing a single object and grasps are generated and executed in simulation. The success or failure of each simulated grasp is recorded. As described in Section 3.1, the uncertainties during grasping are still the major challenge in robotic grasping. Thus, the critical problem is that the training dataset must consider the uncertainty. The dataset generation setting is described below:

MuJoCo [85] is selected as a physics simulator to generate and execute grasps in simulation. The DLR-II hand is used in the real robot experiment, thus the 3D model of the DLR-II hand is used in the simulation. There is no kinematic constraint on the hand to grasp the object except colliding with the virtual table. All 3D object models in the dataset are decomposed into convex parts, because MuJoCo acquires objects to consist of convex parts for collision checks. Figure 3.10 shows the approximates decompositions of some objects in our dataset, using the V-HACD algorithm [86].

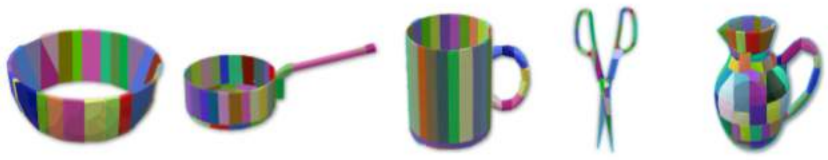


Figure 3.10: Approximate convex decomposition of some objects in our dataset. Best viewed in colour.

The total 3D object model number is 294 from 20 categories in simulation. The 294 object models include: bottles, bowls, cans, boxes, kitchen utensils, mugs, cups, pans, tennis balls and dustpans, as shown in Fig 3.11. 250 of the 294 objects are used for training the network, and the other 44 objects are included in the test set. The 294 objects have a similar distribution of mass, size and frictional coefficient to the real-world objects, and the size of the objects in the simulation are chosen to be within the grasp affordance limits of the DLR-II hand. Each object is assigned a weight that is uniformly sampled from its class range, as given in Table 3.2 in gram units. The friction coefficients of each object are sampled from a range of $[0.5, 1]$

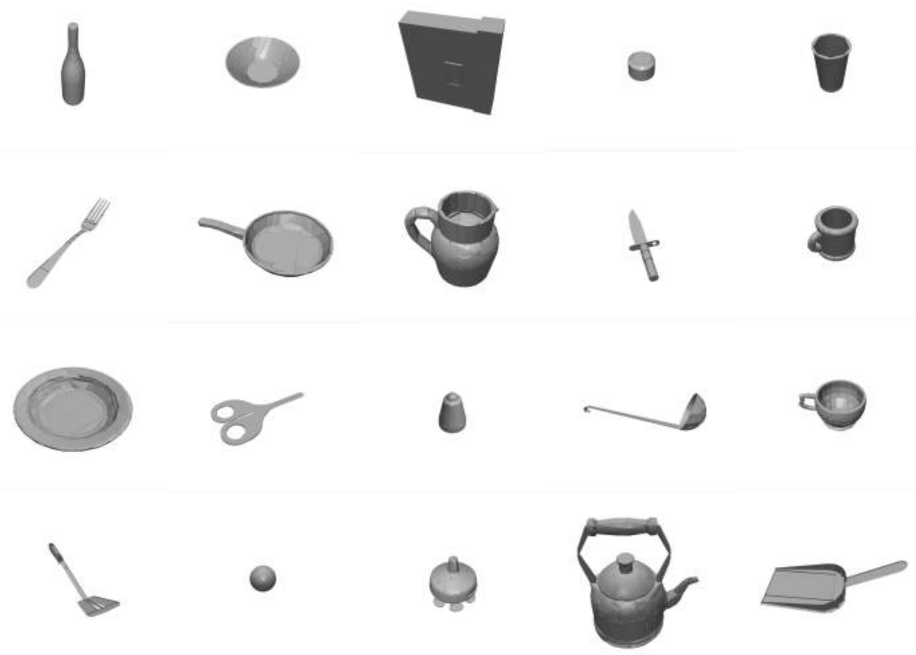


Figure 3.11: Sample objects from all classes in the 3D model dataset.

in MuJoCo default units. In this way, the training dataset can contain a variety of different instances that help the evaluative model make a robust evaluation both in simulation and the real-world experiment.

Bottle	Bowl	Box	Can	Cup	Pan	Fork
30-70	50-400	50-500	200-400	30-330	150-450	40-80
Teapot	Scissors	Spoon	Mug	Shaker	Teacup	Plate
500-800	50-150	40-80	250-350	100-160	150-250	40-80
Dustpan	Knife	Spatula	Funnel	Ball	Jug	
100-150	50-150	40-80	40-80	50-70	80-200	

Table 3.2: Weight ranges(g) for different object category.

B. Data Collection

All training data were collected in simulation. During the collection process, a single object is placed on the virtual desktop in the simulator, and then the grasp generator generates grasps to grasp the object. These grasps and the object are called a scene, and due to the process and time of data collection, the number of grasps per scene is different. Each scene contains 10-100 grasps, and each scene has at least one successful grasp.

The time flow of data collection is introduced in the following section. The data collection

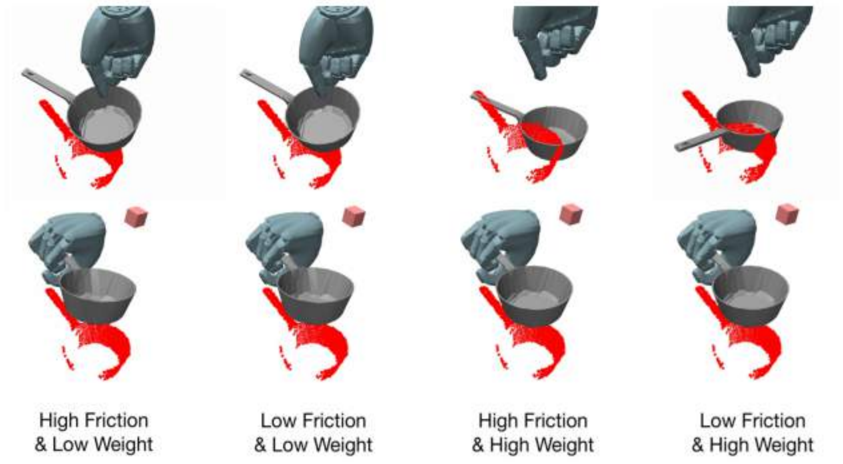

High Friction & Low Weight
Low Friction & Low Weight
High Friction & High Weight
Low Friction & High Weight

Figure 3.12: For a richer dataset. From left to right column: the same pot with varying friction and mass parameters. Top row: The same pinch grasp executed. Bottom row: A more robust grasp, executed under the same object and conditions.

process can be divided into the following four phases: First, an object from the 294 3D models is generated and placed on a virtual table. In order to collect different data from limited amount object models, every time this step is performed, object position, orientation, scale, weight and friction coefficients are changed randomly within the limits. Fig 3.12 shows the effect of these changes. Second, a depth camera takes a depth image I_0 of this scene and converts it into a point cloud. The elevations of the camera viewpoint vary between $30 - 57$ degrees, whereas the azimuths are sampled from $[0, 2\pi]$. Other simulated depth images are taken from different views around the object. Third, the grasp generator by Kopicki et al. [15], as described in Section 3.2, is used to generate candidate grasps with the first depth image I_0. Fourth, in each scene, other simulated depth images, as described in the second step, are taken. Up to 20 distinct views are generated in each scene and about 100 grasps are performed in each scene. Each depth image is associated with each grasp created in the third step randomly. Adding a different view for each grasp instead of just using the image I_0 ensures that there are more changes in the viewpoint and it helps to build a richer dataset. Additionally, a grasp is considered as successful if an object is lifted 1 meter above the table, and the object is held there for 2 seconds. If the object slips from the hand during lifting or holding, the grasp is considered as a failure grasp.

After these four phases, data collection is completed. Grasp result, grasp information and depth image are stored as results for training and testing network. In total, 1081332 grasps in 20025 scenes were collected for the training dataset, consisting of equal numbers of successes and failures. 99521 grasps in 1539 scenes were taken for the test dataset, containing 51711 successful and 47810 failed grasps.

3.5.2 Experiment in simulation

In this section, an analysis of two evaluative model architectures in a simulation test is presented. Both evaluative networks are trained on the same training dataset that has 1081332 grasps on 20025 scenes. Once the network is trained, it is used to predict the success probability of the grasps in the test set, which consists of 99521 grasps in 1539 scenes, containing 51711 successful and 47810 failed grasps. Those objects in the test set are unseen objects.

As shown in Table 3.3, the grasp success prediction accuracy of generative-evaluative architecture EM1 (GEA_EM1) is 77.28% for the test set, and the grasp success prediction accuracy of the top grasp in each scene is 87.72%. The grasp success prediction accuracy of generative-evaluative architecture EM2 (GEA_EM2) is 78.66% for the test set, and the grasp success prediction accuracy for the top grasp in each scene is 89.73%. Generative model architecture (GM) is ranking by Kopicki et al. [15] origin method and the proposed generative-evaluative learning architectures (GEA) are ranked by evaluative network prediction. According to the Table 3.3, the results of two generative-evaluative models on the overall accuracy and Top-1 accuracy are significantly better than GM only. In addition, the results on the overall accuracy and top-1 accuracy of the GEA_EM2 are better than the GEA_EM1. The comparative success of the GEA_EM2 shows that the custom feature extraction model for the grasp parameters allows the network to learn a more useful feature representation compared to a single layer.

Table 3.3: Overall grasp success prediction accuracy for two generative-evaluative models on test set. Fail(Fail) represent the network predict a failed grasp as a failed grasp. Fail(Suc) represent the network predict a failed grasp as a successful grasp. Suc(Fail) represent the network predict a successful grasp as a failed grasp. Suc(Suc) represent the network predict a successful grasp as a successful grasp.

	#				%	%
	Fail(Fail)	Suc(Suc)	Fail(Suc)	Suc(Fail)	Overall acc	Top-1 acc
GM	-	-	-	-	48.50	69.30
GEA_EM1	38316	38590	9494	13121	77.28	87.72
GEA_EM2	40271	38015	10069	11166	78.66	89.73

Further analysis of Top-1 grasps in each scene is provided in Table 3.4. MFEF represents Top-1 grasp selected by GM fail and Top1 grasp selected by evaluative model fail in the same scene. MSES represents Top-1 grasp selected by GM success and Top1 grasp selected by evaluative model success in the same scene. MFES represents Top-1 grasp selected by GM fail and Top1 grasp selected by evaluative model success in the same scene. MSEF represents Top-1 grasp selected by GM success and Top1 grasp selected by evaluative model fail in the same scene. The results show more MFES than MSEF, which indicated that the evaluative model works well for selecting Top-1 grasps, which is important to the success of grasping an object.

Table 3.4: Further analysis of Top-1 grasps in each scene.

	# MFEF	# MSES	# MFES	# MSEF
EM1	139	1020	330	50
EM2	115	1027	354	43

In order to make a comparison with Kopicki et al. [15] in simulation, the success probability of the same ranking position is shown in Fig 3.13. From the graph, grasp success probability from the two GEA models falls monotonically, which means for each scene, the grasp success probability of top k grasp from our network prediction is higher than the grasp success probability of top $k + 1$ grasp. This result proves that the evaluative network is valid and reasonably evaluates different grasps quality. On the other hand, the likelihood-based ranking of GM results in many good grasps being low-ranked.

3.5.3 Experiment in real robot

An ideal situation is to test EM1 and EM2 with the real robot, but the real robot test is time-consuming and expensive. Further, hardware abrasion is a problem, so only EM1 is tested using the real robot. For making a comparison in the real-world experiment with Kopicki et al. [15], a new test set is created that contains more challenging objects and places them in challenging poses. The test set in the real world contains 40 test objects and another six training objects, as shown in Fig 3.14. Humans use these training objects to demonstrate ten sample grasps as shown in Fig 3.2. 35 of the 40 objects are included in the category of the simulation environment, 5 of the 40 objects are not included in the category of the simulation environment. The 40 test objects were used to generate 49 object-pose pairs.

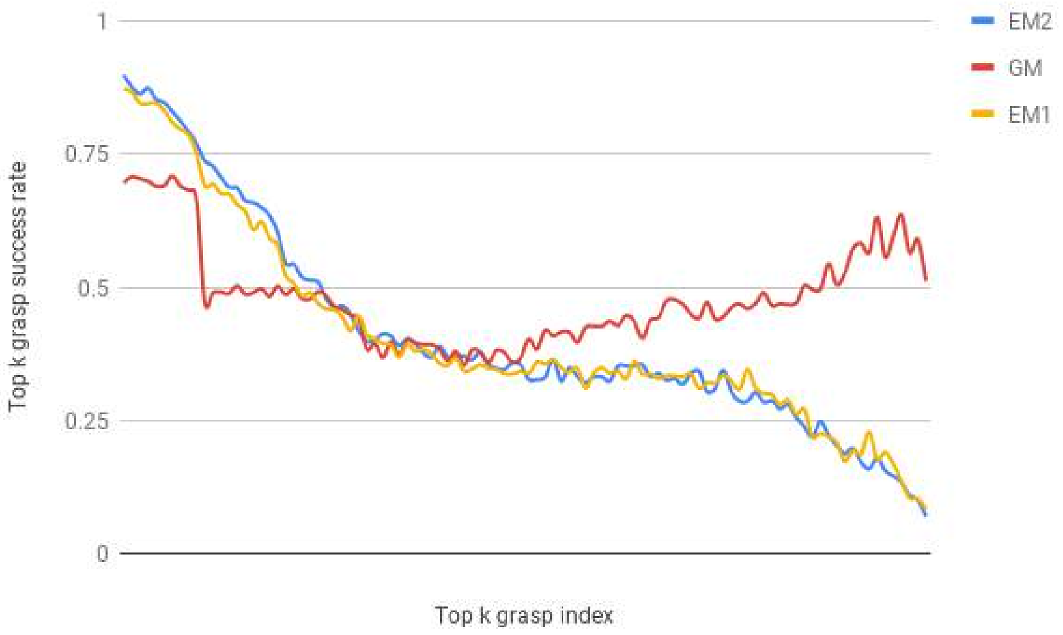


Figure 3.13: Grasp success probability vs. grasp ranking in simulation.

All 49 object-pose pairs are evaluated by model GM and model EM1 using a paired trials methodology. The highest ranked grasp was executed after the model evaluated and generated a ranked list of grasps. If the object is stable in the robot hand for five seconds before it is automatically released after five seconds, the grasp is considered as a successful grasp. As shown in Table 3.5, the GM algorithm achieved a grasp success rate 59.2% (29/49) on the test set, and the EM1 algorithm achieved a grasp success rate 79.6% (39/49) on the test set. The difference between the two algorithms has a p-value of 0.0442, and so is statistically significant. A selection of grasps where the two methods performed differently are shown in Fig. 3.15

Table 3.5: Results of the real robot.

		GM	
		# succs	# fails
EM1	# succs	23	15
	# fails	5	6

3.6 Discussion and summary

In this chapter, two novel evaluative network architectures for dexterous grasp under uncertainties of the grasped objects with a generative model are proposed. First, a grasp generator

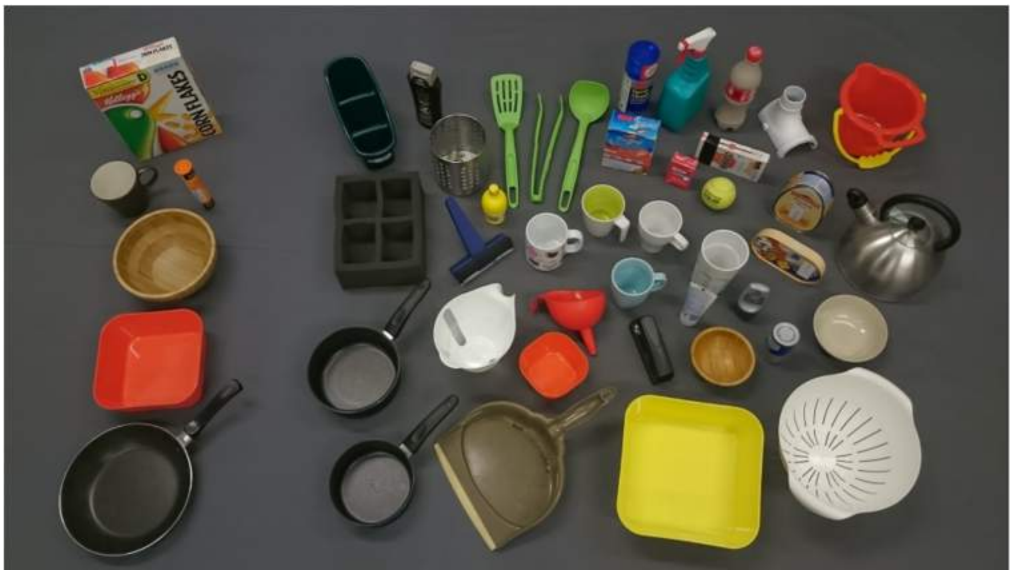


Figure 3.14: The real objects. The training objects are on the left, testing objects are on the right.

is learned, based on Kopicki et al. [15], and became part of the generative-evaluative model. Then, the grasp generator is used to generate grasps in simulation, the results of which are used to train a deep neural network. These two parts construct the generative-evaluative model.

According to the analysis of the performance of the proposed method, the deep evaluative network shows robustness to unobservable variation. The results showed that our evaluative model could evaluate a multi-fingered grasp that applied to a novel object with strict restrictions and increased the grasp success rate on real novel objects by achieving a better grasp ranking.

Finally, some limitations and future directions will be discussed. First, the evaluative model is learning from partial shape information, this makes it dependent on the angle at which the object is placed. Although the noise in depth images in training data can help this problem to some extent, there is still room for improvement. Second, the proposed generative-evaluative model in this chapter goes one-way, from the generating grasps to evaluating grasps, and there is no feedback from the evaluative model. Therefore, as a solution, in the next chapter, a method that planning grasps as probabilistic inference in the evaluative network will be presented.

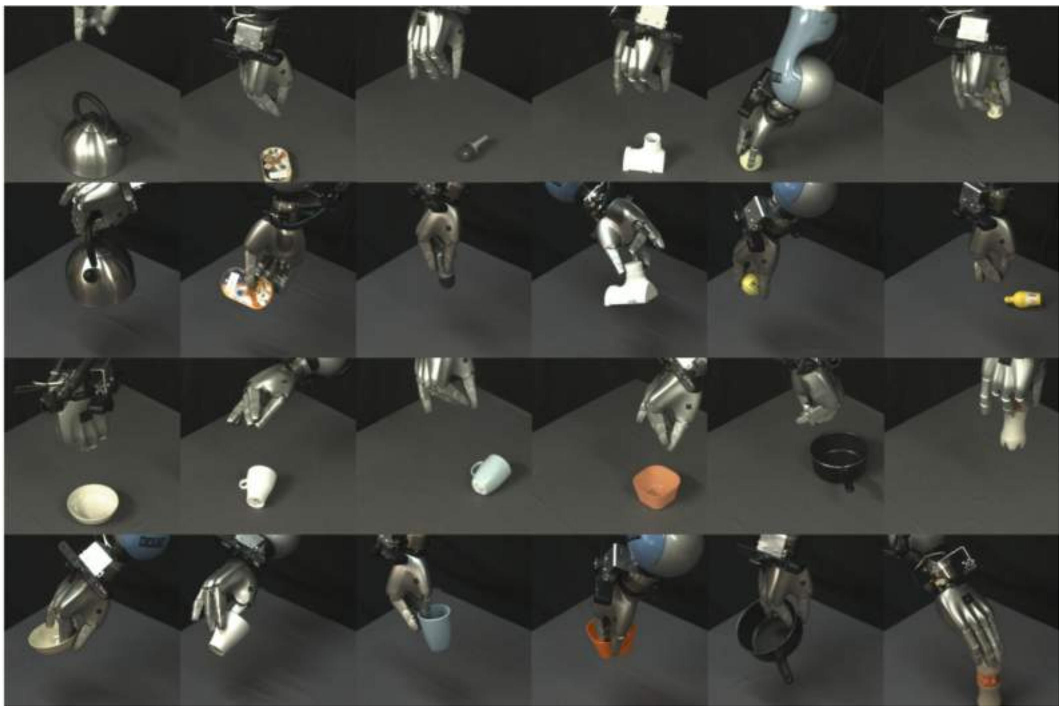


Figure 3.15: Examples of grasps generated by the generative model (GM, 1st and 3rd row) and the generative-evaluative model (GEA, 2nd and 4th row) on paired trials. The first five columns show some of the 15 cases where the GEA model succeeds but GM fails. The far right-hand column shows 2 of the 5 converse cases.

Chapter 4: Planning dexterous grasps with gradient ascent

4.1 Introduction

In the Chapter 3, the challenges involved in the dexterous grasping of novel objects given only a single viewpoint were discussed. The present research proposes that the way to meet these challenges is to use an evaluative model based on a deep learning algorithm to improve the accuracy of the ranking of good quality grasps generated by Kopicki et al. [15]. The evaluative model shows robustness in identifying successful grasps, but after the evaluation of the grasps, the candidate grasps generated by the Kopicki et al. [15] algorithm are not changed, which imposes the limitation that grasping success is entirely dependent on the quality of the generated grasps. To address this issue, inspired by Lu et al. [11], this section attempts to optimise the grasp parameters by finding the hand parameters which maximise the probability of success of a given grasp. The proposed method comprises of two parts:

- A learned generative model for generating candidate grasps.
- A trained evaluative model is used to optimise grasps and then evaluate them.

The EM2 architecture has been chosen as the means for encoding grasp parameters for the planner. As described in Section 3.3, the depth image I_t and grasp parameters h_t are fed into the neural network as inputs. Because the EM2 has already been trained, for a new object o_i, the network can be used to perform inference to maximise the probability of dexterous grasp success $P(Y = 1|o_i, h_t)$. This optimisation process is performed by a gradient-ascent optimisation with constraints via the evaluative network, using a back-propagation algorithm. Whilst this approach did not improve the performance, this study analyses the possible causes of the failure and, at the end of this chapter, presents a promising solution.

4.2 Optimising dexterous grasps with deep network

Unlike Lu et al. [11], for each scene, instead of keeping the image input fixed or moving it as a function of the palm pose of the grasp, simulated depth images are taken from virtual cameras around the object. Therefore, for each scene, the generative model uses the first depth image I_0 in order to generate grasps. Following this, each depth image (each generated from a distinct viewpoint of the object) is associated with generated grasps on a random basis.

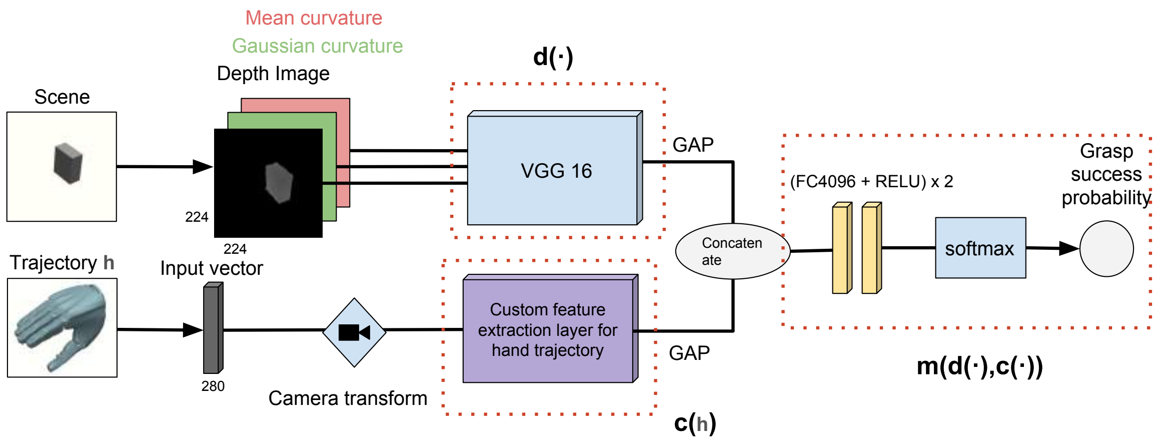


Figure 4.1: EM2 model. Top left: the colourised depth image representation, I_t, and image feature network channel $d(\cdot)$. Bottom: hand configuration network channel $c(\cdot)$. Right: the mixed prediction network and output $m(\cdot)$

As shown in Fig. 4.1, the network EM2 $f(\cdot)$ can be sub-divided into sub-modules $d(\cdot)$, $c(\cdot)$, and $m(\cdot)$, so that $f = m(d(I_t); c(h_t))$. Thus, the gradient with respect to the input h_t is computed by the following equation:

$$\frac{\partial f}{\partial h_t} = \frac{\partial m}{\partial c}\frac{\partial c}{\partial h_t} \tag{4.1}$$

After gradient is calculated, h_t is calculated by gradient-ascent as shown in Eq. 4.2 .

$$op\_h_t^i = h_t^i + \alpha * grad\_h_t^i \tag{4.2}$$

where i is the index of hand parameters and $op\_h_t^i$ is the result of point i after processing. h_t^i is origin value of hand parameter, $grad\_h_t^i$ is the gradient and α is the learning rate or called step size.

4.3 Experiment

The following parameters are used when performing inference. The wrist location is constrained within a $0.01 * 0.01 * 0.01\mathrm{m}$ cube around the palm location, which is given as an initialisation. The number of iterations is from 2 to 20 for the optimisation process. In order to compare the performance of this approach with that of EM2 without optimisation, a test set with 951 scenes is used. Three groups, each with a different configuration, are used for the experiment.

- Group A : 2 times optimisation for all way-points of each grasp.
- Group B : 2 times optimisation for the last 3 way-points of each grasp.
- Group C : 20 times optimisation for the last 3 way-points of each grasp.

Three ranking methods are used.

- GEA ranking: This represents the original ranking made by the generative-evaluative network without optimisation.
- GEA ranking after optimisation: This represents the ranking made by the generative-evaluative network after grasp optimisation.
- GM ranking: This represents the original ranking made by the generative model based on Kopicki et al. [15], without optimisation.

The baseline for the experiment was 48.21% overall grasp success, 69.5% best grasp success (according to GM ranking) and 89.74% top (best) grasp success (according to GEA ranking). GM represents the generative model alone; GEA represents the deep network. None of the experiments across the three groups achieved better results when compared to baseline.

The top (best) grasp success (according to the GM ranking) in Group A was 63.40%, the top grasp success (according to the GEA ranking) in Group A was 77.77% and the overall grasp success of Group A was 42.60%.

The top grasp success (according to the GM ranking) in Group B was 70.14%, the top grasp success (according to the GEA ranking) in Group B was 89.1% and the overall grasp success in Group B was 47.90%.

The top grasp success (according to the GM ranking) in Group C was 69.36%, the top grasp success (according to the GEA ranking) in Group C was 89.02% and the overall grasp success in Group C was 47.50%.

Table 4.1: The optimisation results of three groups.

	Baseline	Group A	Group B	Group C
Overall grasp success	48.21%	42.60%	47.90%	47.50%
Top grasp success (GM ranking)	69.5%	63.40%	70.14%	69.36%
Top grasp success (GEA ranking)	89.74%	77.77%	89.10%	89.02%

For further analysis, it is essential to determine whether gradient ascent optimisation improves the grasp quality across the entire test set. As shown in Table 4.3: 6905 grasps transitioned from success to failure after optimisation, 3300 grasps transitioned from failure to success after optimisation and 182 grasps transitioned to colliding grasps. Table 4.2 shows a similar pattern: 1922 grasps transitioned from success to failure after optimisation, 1572 grasps transitioned from failure to success after optimisation and 61 grasps transitioned to colliding grasps. These results demonstrate that although optimisation makes some of the failed grasps successful, unfortunately, more of the originally successful grasps become failed.

Table 4.2: Grasp analysis: 2 epoch, all way-points

Group A				
		Success	Fail	Collide
Baseline	Success	22550	6905	60
	Fail	3300	27910	122
	Collide	3	3	0

Table 4.3: Grasp analysis: 2 epoch, 3 way-points

Group B				
		Success	Fail	Collide
Baseline	Success	27568	1922	25
	Fail	1572	29724	36
	Collide	3	3	0

The next question to ask is, how does the optimisation process affect the good quality grasps? Hence, the Top-1 grasps from the GM ranking and the GEA ranking in Group A must be analysed. As shown in Table 4.4, in the Top-1 grasps selected by the GM ranking across 951 scenes, there are 104 Top-1 grasps of scenes which transitioned from success to failure and 51 grasps of scenes which transitioned from failure to success. As shown in Table 4.5, in the Top-1 grasps selected by the GEA ranking across the same 951 scenes, there are 15 Top-1 grasps of scenes which transitioned from success to failure and ten grasps of scenes which transitioned from failure to success. The results show that the optimisation process does not improve the quality of the Top-1 grasps —either those selected by the GM ranking or by the GEA ranking.

Table 4.4: Top grasp analysis (GM ranking): all way-points

Group A				
		Success	Fail	Collide
Baseline	Success	552	104	3
	Fail	51	241	0
	Collide	0	0	0

Table 4.5: Top grasp analysis (GEA ranking): all way-points

Group A				
		Success	Fail	Collide
Baseline	Success	838	15	0
	Fail	10	88	0
	Collide	0	0	0

Finally, Table 4.6 verifies whether the evaluation network can effectively evaluate optimised grasps. After optimisation, the network considers that 99% optimised grasps are better than the original grasps. However, in Table 4.6, the Top-1 grasp selected by the GEA ranking after optimisation across every scene predicts success in only 688 scenes, far below the baseline of 853. This indicates that the network, which was trained by the original grasps, does not evaluate the optimised grasps correctly.

Table 4.6: Top grasp analysis (GEA ranking after optimization): all way-points

Group C				
		Success	Fail	Collide
	Success	672	9	2
Baseline	Fail	16	250	1
	Collide	0	0	1

4.4 Discussion and summary

In this section, the method of using gradient ascent to optimise the grasps via a trained evaluative network is described. The experimental result showed that the evaluative network can neither improve the quality of each grasp nor predict the correct optimised grasp ranking. One hypothesis to explain this is that the network search space for grasp parameters is too small, due to the network being trained with grasps that were generated by [15]. Further, the Kopicki et al. [15] model was trained using only ten demonstration grasps. Based on this analysis of experimental results, there are a number of possible ways to improve this optimisation method:

1. From the result, it appears that the fewer way-points which are optimised, the better the performance. 280 dimensions may be too high for network optimisation. So reducing the number of parameter dimensions is one promising direction.

2. The optimisation process clearly improves the predicted probability of success of each grasp but has fairly low confidence in its prediction. One approach to solve this problem is to evaluate the grasping by using a network which has been retrained with optimised grasps.

3. The grasps generated by Kopicki et al. [15] are limited to ones observed from human demonstrations, which may result in a too small a search space. It might be a good option to use other methods of generating grasps, such as heuristic grasp procedures.

Chapter 5: Summary and Discussion

The main work of this thesis is summarised in this chapter. The limitations and potential future working directions will also be discussed.

5.1 Main work

Throughout this thesis, I proposed to address a fundamental problem in robotic grasping: When grasps of an object are generated, how to decide which one to execute? The main work of this thesis encompasses three aspects:

1. First, a generative-evaluative model for dexterous grasping is proposed. The generative model is a dexterous grasps generator based on a learning method, the evaluative model is a deep neural network for evaluating dexterous grasps. The model shows that the deep network has a good ability to evaluate dexterous grasps success probability.

2. Second, two novel evaluative network architectures are proposed. The results show that a custom feature extraction layer in EM2 for hand parameters significantly contributes to the evaluation accuracy. Further, the results show that the evaluative network can transfer the knowledge of grasp evaluation from a simulation environment to the real world.

3. Third, the optimisation method based on the evaluative model for dexterous grasps is presented. From the experimental result, the optimisation method does not show the potential for improving the quality of each grasp or predicting the correct optimised grasp ranking. Three promising approaches are proposed as the future direction to improve this method.

5.2 Future work

Despite the future works that are illustrated in Section 3.6 and Section 4.4, a logical extension beyond the evaluative network is an end to end deep generative-evaluative network. The key challenge of the deep generative-evaluative network is how to use a single image of the object to generate various dexterous grasps. Generative adversarial network[87] is a promising solution. Recently, Firman et al. [88] proposed a novel method to predict some possible answers with a single input which is suitable for grasping as well. Once the policy network for generating grasps is established, this will be a good point to start reinforcement learning by pairing the policy network with the evaluative network, and jointly training those two networks. It may take multiple iterations of interlaced training to tune the whole end to end deep generative-evaluative network.

References

[1] Hanna Yousef, Mehdi Boukallel, and Kaspar Althoefer. Tactile sensing for dexterous in-hand manipulation in roboticsâĂŤa review. *Sensors and Actuators A: physical*, 167(2): 171–187, 2011.

[2] Beatriz León, Antonio Morales, and Joaquin Sancho-Bru. Robot grasping foundations. In *From Robot to Human Grasping Simulation*, pages 15–31. Springer, 2014.

[3] Tsuneo Yoshikawa. Multifingered robot hands: Control for grasping and manipulation. *Annual Reviews in Control*, 34(2):199–208, 2010.

[4] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[5] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3406–3413. IEEE, 2016.

[6] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with large-scale data collection. In *International Symposium on Experimental Robotics*, pages 173–184. Springer, 2016.

[7] Edward Johns, Stefan Leutenegger, and Andrew J Davison. Deep learning a grasp function for grasping under gripper pose uncertainty. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4461–4468. IEEE, 2016.

[8] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[9] Stanimir Dragiev, Marc Toussaint, and Michael Gienger. Gaussian process implicit sur-

faces for shape estimation and grasping. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2845–2850. IEEE, 2011.

[10] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. Leveraging big data for grasp planning. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4304–4311. IEEE, 2015.

[11] Qingkai Lu, Kautilya Chenna, Balakumar Sundaralingam, and Tucker Hermans. Planning multi-fingered grasps as probabilistic inference in a learned deep network. In *International Symposium on Robotics Research*, 2017.

[12] Jacob Varley, Jonathan Weisz, Jared Weiss, and Peter Allen. Generating multi-fingered robotic grasps via deep learning. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4415–4420. IEEE, 2015.

[13] Matthew Veres, Medhat Moussa, and Graham W Taylor. Modeling grasp motor imagery through deep conditional generative models. *IEEE Robotics and Automation Letters*, 2 (2):757–764, 2017.

[14] Yilun Zhou and Kris Hauser. 6dof grasp planning by optimizing a deep learning scoring function. In *Robotics: Science and Systems (RSS) Workshop on Revisiting Contact-Turning a Problem into a Solution*, 2017.

[15] Marek Kopicki, Renaud Detry, Maxime Adjigble, Rustam Stolkin, Ales Leonardis, and Jeremy L Wyatt. One-shot learning and generation of dexterous grasps for novel objects. *The International Journal of Robotics Research*, 35(8):959–976, 2016.

[16] John R Napier. The prehensile movements of the human hand. *The Journal of bone and joint surgery. British volume*, 38(4):902–913, 1956.

[17] Richard M Murray. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[18] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *ICRA*, volume 348, page 353. Citeseer, 2000.

[19] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[20] Matthew T Mason. *Mechanics of robotic manipulation*. MIT press, 2001.

[21] J Kenneth Salisbury and B Roth. Kinematic and force analysis of articulated mechanical hands. *Journal of Mechanisms, Transmissions, and Automation in Design*, 105(1):35–41, 1983.

[22] Matei Ciocarlie, Andrew Miller, and Peter Allen. Grasp analysis using deformable fingers. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 4122–4128. IEEE, 2005.

[23] Matei Ciocarlie, Claire Lackner, and Peter Allen. Soft finger model with adaptive contact geometry for grasping and manipulation tasks. In *EuroHaptics Conference, 2007 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2007. Second Joint*, pages 219–224. IEEE, 2007.

[24] Imin Kao and Fuqian Yang. Stiffness and contact mechanics for soft fingers in grasping and manipulation. *IEEE Transactions on Robotics and Automation*, 20(1):132–135, 2004.

[25] Qiao Lin, Joel W Burdick, and Elon Rimon. A stiffness-based quality measure for compliant grasps and fixtures. *IEEE Transactions on Robotics and Automation*, 16(6): 675–688, 2000.

[26] Antonio Bicchi. On the problem of decomposing grasp and manipulation forces in multiple whole-limb manipulation. *Robotics and Autonomous Systems*, 13(2):127–147, 1994.

[27] Mark R Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on robotics and automation*, 5(3):269–279, 1989.

[28] Henrik Olsson, Karl Johan Åström, Carlos Canudas De Wit, Magnus Gäfvert, and Pablo Lischinsky. Friction models and friction compensation. *Eur. J. Control*, 4(3):176–195, 1998.

[29] V Van Geffen. A study of friction models and friction compensation. *DCT*, 118:24, 2009.

[30] Daniel Charles Drucker. Coulomb friction, plasticity, and limit loads. Technical report, BROWN UNIV PROVIDENCE RI DIV OF APPLIED MATHEMATICS, 1953.

[31] Aris Alissandrakis, Chrystopher L Nehaniv, and Kerstin Dautenhahn. Correspondence mapping induced state and action metrics for robotic imitation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):299–307, 2007.

[32] Renaud Detry, Emre Başeski, Mila Popović, Younes Touati, Norbert Krüger, Oliver Kroemer, Jan Peters, and Justus Piater. Learning continuous grasp affordances by sensorimotor exploration. In *From motor learning to interaction learning in robots*, pages 451–465. Springer, 2010.

[33] Marcus Gualtieri, Andreas ten Pas, Kate Saenko, and Robert Platt. High precision grasp pose detection in dense clutter. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 598–605. IEEE, 2016.

[34] V-D Nguyen. Constructing stable grasps in 3d. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 234–239. IEEE, 1987.

[35] Antonio Bicchi. On the closure properties of robotic grasping. *The International Journal of Robotics Research*, 14(4):319–334, 1995.

[36] Sulabh Kumra and Christopher Kanan. Robotic grasp detection using deep convolutional neural networks. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 769–776. IEEE, 2017.

[37] Carlo Ferrari and John Canny. Planning optimal grasps. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2290–2295. IEEE, 1992.

[38] Brian Mirtich and John Canny. Easily computable optimum grasps in 2-d and 3-d. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 739–747. IEEE, 1994.

[39] Yun-Hui Liu. Computing n-finger form-closure grasps on polygonal objects. *The International journal of robotics research*, 19(2):149–158, 2000.

[40] Ch Borst, Max Fischer, and Gerd Hirzinger. Calculating hand configurations for precision and pinch grasps. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 2, pages 1553–1559. IEEE, 2002.

[41] Carlos Rosales, Lluís Ros, Josep M Porta, and Raúl Suárez. Synthesizing grasp configurations with specified contact regions. *The International Journal of Robotics Research*, 30(4):431–443, 2011.

[42] Junggon Kim, Kunihiro Iwamoto, James J Kuffner, Yasuhiro Ota, and Nancy S Pollard. Physically based grasp quality evaluation under pose uncertainty. *IEEE Transactions on Robotics*, 29(6):1424–1439, 2013.

[43] Yu Zheng and Wen-Han Qian. Coping with the grasping uncertainties in force-closure analysis. *The International Journal of Robotics Research*, 24(4):311–327, 2005.

[44] Vassilios N Christopoulos and Paul Schrater. Handling shape and contact location uncertainty in grasping two-dimensional planar objects. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1557–1563. IEEE, 2007.

[45] Máximo A Roa and Raúl Suárez. Computation of independent contact regions for grasping 3-d objects. *IEEE Transactions on Robotics*, 25(4):839–850, 2009.

[46] Van-Duc Nguyen. Constructing force-closure grasps. *The International Journal of Robotics Research*, 7(3):3–16, 1988.

[47] Alberto Rodriguez, Matthew T Mason, and Steve Ferry. From caging to grasping. *The International Journal of Robotics Research*, 31(7):886–900, 2012.

[48] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 348–353. IEEE, 2000.

[49] Karun B Shimoga. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research*, 15(3):230–266, 1996.

[50] Li Zhang and Jeffrey C Trinkle. The application of particle filtering to grasping acquisition with visual occlusion and tactile sensing. In *Robotics and automation (ICRA), 2012 IEEE international conference on*, pages 3805–3812. IEEE, 2012.

[51] Claire Dune, Eric Marchand, Christophe Collowet, and Christophe Leroux. Active rough shape estimation of unknown objects. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3622–3627. IEEE, 2008.

[52] Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, Jonathan Kleinehellefort, and Michael Beetz. General 3d modelling of novel objects from a single view. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3700–3705. IEEE, 2010.

[53] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesisâĂŤa survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014.

[54] Anis Sahbani, Sahar El-Khoury, and Philippe Bidaud. An overview of 3d object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326–336, 2012.

[55] Ellen Klingbeil, Deepak Rao, Blake Carpenter, Varun Ganapathi, Andrew Y Ng, and Oussama Khatib. Grasping with application to an autonomous checkout robot. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2837–2844. IEEE, 2011.

[56] Mila Popović, Dirk Kraft, Leon Bodenhagen, Emre Başeski, Nicolas Pugeault, Danica Kragic, Tamim Asfour, and Norbert Krüger. A strategy for grasping unknown objects based on co-planarity and colour information. *Robotics and Autonomous Systems*, 58(5): 551–565, 2010.

[57] Mario Richtsfeld and Michael Zillich. Grasping unknown objects based on 21/2d range data. In *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pages 691–696. IEEE, 2008.

[58] Ishay Kamon, Tamar Flash, and Shimon Edelman. Learning to grasp using visual information. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 3, pages 2470–2476. IEEE, 1996.

[59] Andrew T Miller and Peter K Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004.

[60] Andrew T Miller, Steffen Knoop, Henrik I Christensen, and Peter K Allen. Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1824–1829. IEEE, 2003.

[61] Corey Goldfeder, Peter K Allen, Claire Lackner, and Raphael Pelossof. Grasp planning via decomposition trees. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4679–4684. IEEE, 2007.

[62] Christoph Borst, Max Fischer, and Gerd Hirzinger. Grasping the dice by dicing the grasp. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 4, pages 3692–3697. IEEE, 2003.

[63] Matei T Ciocarlie and Peter K Allen. Hand posture subspaces for dexterous robotic grasping. *The International Journal of Robotics Research*, 28(7):851–867, 2009.

[64] David Fischinger and Markus Vincze. Empty the basket-a shape based learning approach for grasping piles of unknown objects. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2051–2057. IEEE, 2012.

[65] Ashutosh Saxena, Lawson LS Wong, and Andrew Y Ng. Learning grasp strategies with partial shape information. In *AAAI*, volume 3, pages 1491–1494, 2008.

[66] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008.

[67] Oliver Kroemer, Emre Ugur, Erhan Oztop, and Jan Peters. A kernel-based approach to direct action perception. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2605–2610. IEEE, 2012.

[68] Staffan Ekvall and Danica Kragic. Interactive grasp learning based on human demonstration. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3519–3524. IEEE, 2004.

[69] Sing Bing Kang. *Robot instruction by human demonstration*. PhD thesis, Carnegie Mellon University Pittsburgh, PA, 1994.

[70] Staffan Ekvall and Danica Kragic. Learning and evaluation of the approach vector for automatic grasp generation and planning. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4715–4720. IEEE, 2007.

[71] Carlo Ferrari and John Canny. Planning optimal grasps. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2290–2295. IEEE, 1992.

[72] Raphael Pelossof, Andrew Miller, Peter Allen, and Tony Jebara. An svm learning approach to robotic grasping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3512–3518. IEEE, 2004.

[73] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[74] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[75] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[76] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436, 2015.

[77] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and

potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[78] Joseph Redmon and Anelia Angelova. Real-time grasp detection using convolutional neural networks. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1316–1322. IEEE, 2015.

[79] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[80] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *arXiv preprint arXiv:1709.07857*, 2017.

[81] Fangyi Zhang, Jürgen Leitner, Zongyuan Ge, Michael Milford, and Peter Corke. Adversarial discriminative sim-to-real transfer of visuo-motor policies.

[82] Kuan Fang, Yunfei Bai, Stefan Hinterstoisser, and Mrinal Kalakrishnan. Multi-task domain adaptation for deep learning of instance grasping from simulation. *arXiv preprint arXiv:1710.06422*, 2017.

[83] Changsheng Zhao, Dongming Zhao, and Yubao Chen. Simplified gaussian and mean curvatures to range image segmentation. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 2, pages 427–431. IEEE, 1996.

[84] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feed-forward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[85] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[86] Khaled Mamou and Faouzi Ghorbel. A simple and efficient approach for 3d mesh approximate convex decomposition. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 3501–3504. IEEE, 2009.

[87] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[88] Michael Firman, Neill DF Campbell, Lourdes Agapito, and Gabriel J Brostow. Diversenet: When one right answer is not enough. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5598–5607, 2018.