Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Multimodal Perception for Robotic Grasping and Pouring

Dissertation
with the aim of achieving a doctoral degree at the
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Universität Hamburg

Hongzhuo Liang
Hamburg, 2022

Day of oral defense: 05.05.2022

Abstract

Programming a modern service robot to implement daily tasks requires a thorough understanding of the environment. Inspired by everyday human experience and the multimodal processing mechanism of the human brain, engineers have developed various sensors to build robots with a near-human sensory system. However, extracting valuable data from noisy inputs and efficiently integrating the multiple signals is still challenging. This dissertation investigates the use of multimodal sensory perception for two of the most fundamental service robot tasks, grasping and pouring.

Regarding two-fingered grasping, several effective improvements, *i.e.*, attentional sampling, three-dimensional grid search, and invalid candidate correction, are added to a state-of-the-art grasp candidate generation algorithm. Then a novel deep-learning-based grasp evaluation algorithm *PointNetGPD* is proposed. To the best of our knowledge, *PointNetGPD* is the first work that uses 3D point clouds directly as input for grasp pose evaluation. In parallel, a self-built grasp dataset labels 350K parallel-jaw grasps by meticulous scores based on force-closure quality and friction coefficient values.

To endow robots with the same dexterity as human hands, a closed-loop multifingered grasping framework based on multimodal reinforcement learning is presented. A dexterous grasping simulation environment is built to train a multifingered grasping agent. This agent uses fingertip tactile sensing, joint torques, and hand proprioception as observation and outputs the joint actions for a multifingered hand. To reduce the dimension of the target space, a PCA-based hand synergy is calculated based on a self-collected dataset with pairwise human hand and robot hand motions. In the real robot experiments, the improved grasp generation method and the *PointNetGPD* model are used to determine the initial grasp poses. Furthermore, real robot experiments show that the trained agent can be applied in the real world even if the model is trained purely in simulation.

To tackle the two major challenges, generalization and precision, in the perception for robotic pouring, two neural networks *AP-Net* and *MP-Net*, are proposed. The recurrent neural network *AP-Net* utilizes the audio vibration to estimate liquid height and generalizes well to different experiment settings (*e.g.*, different target containers, different initial liquid heights, different liquid types) while the precision can still be guaranteed. However, the performance of the audio-only *AP-Net* model is limited in noisy environments. Therefore, the novel audio-haptic recurrent deep *MP-Net* is proposed to predict liquid height in real-time and is robust to different levels and types of noise. Moreover, a multimodal pouring dataset including audio-frequency recordings, liquid real-time weight, force-torque data, video of the pouring motion, and source container trajectories during the pouring is collected.

The system assessment and comparison across network evaluation and various robotic experiments show that the proposed multimodal neural networks can successfully solve the grasping and pouring related perception problem. Combining the proposed perception networks learned for grasping and pouring, a service robot can accomplish daily tasks like grasping, pouring, and serving a drink to human users.

Zusammenfassung

Das Programmieren moderner Serviceroboter zur Erfüllung von alltäglichen Aufgaben setzt ein sehr gutes Verständnis der jeweiligen Umgebung voraus. Inspiriert von der Leistungsfähigkeit des menschlichen Gehirns zur multimodalen Verarbeitung und seiner immensen Erfahrung im Alltagsleben haben Forscher verschiedeneste Sensoren entwickelt, um Roboter mit möglichst menschlicher Perzeption auszustatten. Allerdings bleibt die Verarbeitung dieser verrauschten Signale und die effiziente Integration von Signalen verschiedener Quellen eine immense Herausforderung. Diese Doktorarbeit beschäftigt sich mit der Nutzung multimodaler Wahrnehmung für zwei grundsätzliche Aufgaben der Servicerobotik: Das Greifen von Objekten und das Eingießen von Flüssigkeiten.

Zunächst wird ein moderner Algorithmus zur Generierung von Greifhypothesen zum Greifen mittels zweier Finger um mehrere effektive Verbesserungen erweitert. Im Speziellen wird das Generieren unter Aufmerksamkeitsfokus, dreidimensionale Rastersuche und die Korrektur ungültiger Hypothesen eingeführt. Darauf aufbauend wird ein neuer Deep-Learning-basierter Algorithmus *PointNetGPD* zur Evaluation von Greifhypothesen vorgeschlagen. Unseres Wissens nach stellt *PointNetGPD* das erste System dar, das 3D-Punktwolken direkt als Eingabe zur Bewertung von Greifhypothesen nutzt. In diesem Rahmen wird auch ein neuer Datensatz von 350.000 Griffen eines Parallelgreifers mit Qualitätswertung nach Kraftschluss und Reibungskoeffizienten bereitgestellt.

Um Roboter mit der Fingerfertigkeit menschlicher Hände zu versehen, wird ein geschlossenes Regelungssystem zum Mehrfingergreifen eingeführt, das auf multimodalen Daten mittels Verstärkungslernen trainiert wird. Zu diesem Zweck wird eine Simulationsumgebung für Greifexperimente mit Mehrfingergreifern modelliert. Die simulierten Agenten nutzen taktile Daten in den Fingerspitzen, Drehmomente der Gelenke sowie Eigenwahrnehmung der Gelenkpositionen als Eingabe und generieren Bewegungsaktionen für die Gelenke des Greifers. Um die Dimensionalität des Zielraumes zu reduzieren, wird eine Hauptkomponentenanalyse von Synergien verschiedener Handstellungen genutzt. Der hierfür notwendige Datensatz enthält selbstaufgenommene Paare aus menschlichen Handbewegungen und passenden Handbewegungen der Roboterhand. Im realen Roboterexperiment werden die vorgeschlagenen Verbesserungen des Griffgenerators sowie *PointNetGPD* genutzt, um die initiale Griffpose zu bestimmen. Die Experimente demonstrieren, dass der ausschließlich in der Simulation trainierte Agent in der physischen Welt erfolgreich agieren kann.

Die Generalisierbarkeit und die Genauigkeit von Perzeptionssystemen stellen zwei große Hürden in der Umsetzung von Eingießbewegungen dar. Dazu werden zwei neuronale Netze *AP-Net* und *MP-Net* vorgeschlagen, die geeignete Eigenschaften zeigen. Das rekurrente Netz *AP-Net* nutzt Audiovibrationen, um die Füllhohe von Gefäßen während der Gießbewegung zu schätzen. Es generalisiert über verschiedene Experimentalbedingungen, inklusive verschiedener Gefäße, Flüssigkeiten und initialer Füllhöhe, mit zufriedenstellender Präzision. In Umgebungen mit vielen Nebengeräuschen hingegen ist dies nicht mehr der Fall. Um hier robustere Ergebnisse zu erzielen, wird das audio-haptische *MP-Net* entwickelt. Dieses schätzt in Echtzeit die aktuelle Füllhöhe und ist dabei robust gegen verschiedene Arten und Stärken von Störsignalen. Zum Training der Netze wird ein neuer Datensatz mit Eingießbewegungen vorgestellt, der Audiodaten, das aktuelle Füllgewicht des Zielgefäßes, Kraft und Drehmoment am Gießbehälter, sowie Bewegungstrajektorien und Videoaufnahmen enthält.

Die Bewertung und der Vergleich der vorgestellten Systeme über verschiedene physische Experimente hinweg demonstriert den Nutzen multimodaler neuronaler Netze zur Bewältigung von Perzeptionsproblemen für Greif- und Gießaufgaben. Durch Kombination der vorgestellten Lösungsansätze kann ein Serviceroboter Alltagsaufgaben wie das Greifen und Aufheben von Objekten, das Eingießen von Flüssigkeiten oder das Servieren von Getränken für Menschen übernehmen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

To build a robot that can deal with daily tasks to support elderly people is a crucial goal in the service robotic community. This goal can be divided into many subtasks and thus formulate many research areas, among which grasping and manipulation are the most commonly used skills. Unlike most industrial applications that require robots capable of dealing with high-precision duplicated tasks in structured environments, the service robot always works in an unstructured environment. The home environment constantly changes during human activities. Therefore, it is impossible to program every single motion manually. In particular, grasping is fundamental for all manipulation, and pouring is one of the most commonly performed manipulation tasks in the kitchen environment. Thus, a system that programs the robot to accomplish tasks autonomously is required. A promising solution is to perceive the environment via sensory inputs, especially multimodal sensory, and to extract useful information from high-dimensional inputs. Then several general types of robotic software are needed, such as trajectory generation, path planning, and dynamics compensation. Finally, motor commands are sent to the low-level controller, and motors move to the desired position within the desired time limit.

Imagine you are sitting on your sofa, and you are thirsty. Now you would like a service robot to serve you a drink. What kind of skills should the robot possess to achieve this goal? First of all, the robot needs to understand human commands, which can be given by standard computer input/output devices like a keyboard or mouse. Recently, due to the improvement of Natural Language Processing (NLP), using human language input is becoming more and more popular [80]. To simplify the use of the robot, the task commands given to the robot should ideally be the only user input. Then the robot has to complete the rest of the task independently. First, it should autonomously move to where the cups are stored, *e.g.*, the kitchen. Autonomous driving requires the robot to solve the Simultaneous Localization and Mapping (SLAM) problem that updates the environment map and the robot's position as it moves. After it arrives in the kitchen, a **manipulation** skill is needed to open the drawer or the cupboard, then a **grasping** skill is needed to grasp the cup and place it on the table. Then the **grasping** skill is needed again to grasp a source container (with the liquid that the human user wants). With proper planning of the **pouring** motion, the robot will pour the desired amount of liquid with a **pouring height perception** algorithm. In the end, the robot **grasps** the cup, moves towards the human, and hands the drink over.

The tasks mentioned above are complicated for a robot, and every step is a substantial individual research topic in the robotic community. However, these grasping and manipulation tasks are straightforward for humans, even animals. As shown in Figure 1.1, after several days' self-learning, a Sumatran orangutan can grasp a twig and use it as a tool to pry food from an acrylic box. Far beyond the very best systems in artificial intelligence and robotics, the human brain is an extraordinarily dynamic system that robustly integrates vast amounts of information from different and noisy sensory channels. From this immense quantity of raw data, the brain forms a unified and cohesive view of its universe and quickly makes decisions for the specific task.

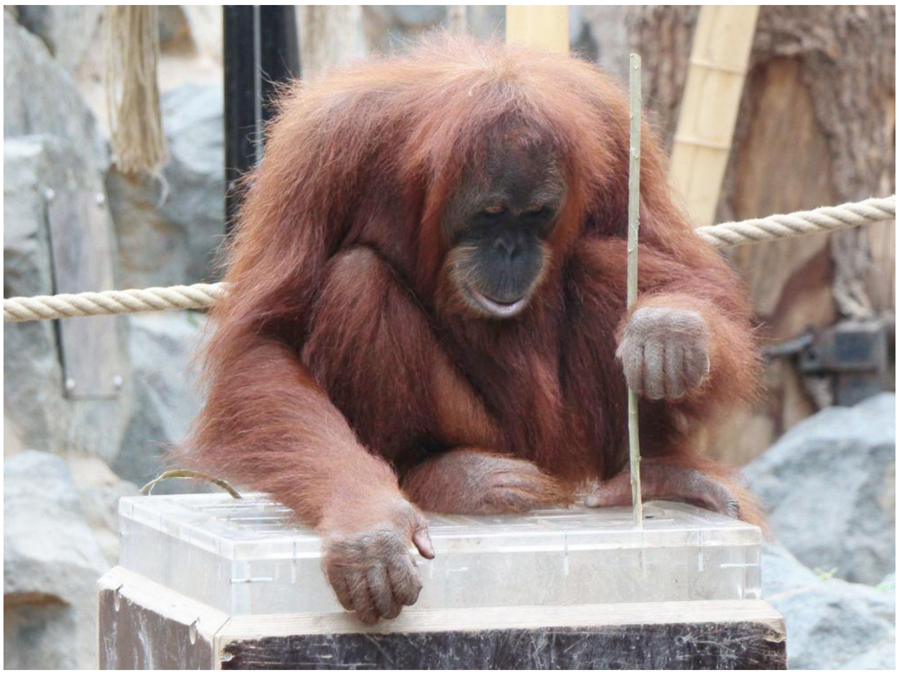


Figure 1.1: A Sumatran orangutan is using a twig as tool to pry food from an acrylic box. Photo was taken by the thesis author at Hagenbecks Tierpark zoo in 2021.

Regarding the grasping task, humans utilize their visual system to locate the objects and generate grasping poses. When the objects are grasped, they take advantage of their hands, containing full-palm tactile sensory. With the tactile sensory, humans can evaluate the grasp poses and adjust finger motions in real-time based on object texture and slippery detection. In addition, the force sensing of human joints helps to estimate the weights and the stiffness of the objects. The thermal sensation and auditory perception are all essential in the manipulation tasks. Regarding the pouring task, humans perform incredible pouring height estimations when they visualize the pouring scenario completely. However, when pouring hot liquid on a cold and dark night, as the temperature difference between the environment and the liquid is high, much steam will be generated, thus occluding the vision-based perception pipeline. Weadon [121] has reported that a blind person can use sound to estimate how near a liquid is to the top of its container. Similarly, normal people can achieve this goal by concentrating on the pouring sounds. Furthermore, the changing weights of the pouring container is another helpful clue in the pouring task.

Inspired by the multi-inputs processing mechanism of the human brain and the human experience, we learn that:

- Visual is usually the dominant modality for grasping and pouring tasks.
- When one dominant modality is occluded or hardly used, other modalities can become the dominant sensations and complement the whole perception system. For example, suppose the environment is dark and noisy during a pouring task. In that case, the tactile, force/torque, and proprioception senses complement the vision and audio sensing system and provide more valuable information.
- It is essential to equip the robot with a similar multisensory system as humans. Humans use multisensory information to accomplish daily tasks, *e.g.*, vision, tactile, proprioception, force/torque, and audio.
- To comprehend the manipulation scenario, the skill learning algorithms for the robots should make use of multiple modalities and be able to extract and integrate valuable data from noisy inputs robustly.

Deep learning and deep neural networks are systems that try to mimic biological neural network architectures. We want such a system to automatically solve problems that are tedious for humans. With the recent development of deep neural networks, image processing [60] and audio processing [47] has achieved exciting results. Compared to traditional methods, deep neural networks can better represent the input data where the feature of the input data can be abstracted automatically during the training. Thus, using a deep neural network to solve robotic perception problems is a good choice.

1.2 Aim of this Thesis

This thesis aims to develop service robot software that are based on multisensory input and achieve everyday tasks such as stable grasping and pouring a specific amount of liquid accurately, as shown in Figure 1.2. A PR2[1] service robot is serving drinks to the human. The scientific problems that this thesis tries to solve are:

- A novel two-fingered robotic grasp pipeline. The grasp pose generation method should generate high-quality grasp candidates based on visual inputs. Moreover, the grasp evaluation algorithm should determine the quality of each grasp candidate.
- A multifingered grasping pipeline that endows robots with the same dexterity as human hands. The multifingered grasping strategy can generate the finger motions of the robotic hand based on multiple sensing modalities.

[1]`https://robots.ieee.org/robots/pr2`

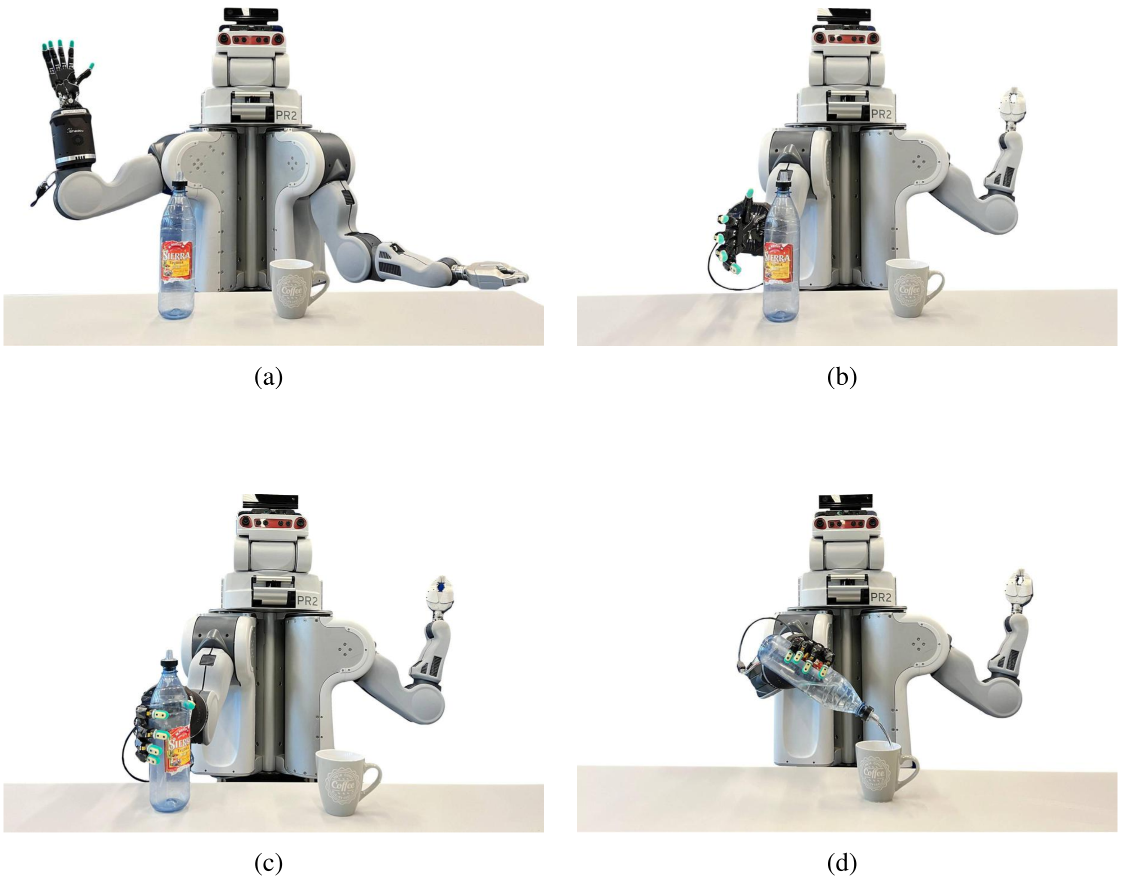


Figure 1.2: An example of the tasks this thesis attempts to solve. (a) PR2 is ready to carry out tasks. (b) and (c) PR2 is grasping a liquid bottle. (d) PR2 is pouring the desired liquid into a mug.

- Estimating real-time pouring height using auditory perception when visual signals are not accessible.
- A robust robotic pouring height estimation algorithm using multimodal inputs. The method should be robust to environmental noise, unknown containers, and other environmental variables.

1.3 Novelty and Contribution of this Thesis

The main contributions of this thesis are listed below:

- **Two-fingered Robotic Grasping Pose Detection using Point Cloud:** We improved the state-of-the-art two-fingered grasping approaches from two aspects: grasp candidate generation and the grasp candidate evaluation. In the grasp candidate generation stage, we propose several improvements [70] to the popular grasp pose generation algorithm [114]. In the grasp evaluation stage, we propose the novel grasp evaluation algorithm PointNetGPD [72] based on point cloud input using PointNet [89] architecture. To the best of our knowledge, this is the first work that uses 3D point clouds directly as input for a grasp evaluation network. Furthermore, we propose a novel grasp dataset generation method that labels the grasp candidates using both the force-closure-based quality value and the friction coefficient value.

- **Multifingered Robotic Grasping Using Proprioception, Tactile, and Torque:** We propose a multifingered grasping framework that takes the output of PointNet-GPD as the initial grasp and then uses a multimodal Reinforcement Learning (RL) framework for multifingered grasping [69]. Fingertip tactile sensing, joint torques, and hand proprioception are fused as the training observation, and the hand synergy principle reduces the target action space. The proposed framework generates robust grasp poses in simulation and the real world with the proper design of multisensory inputs and training methods.
- **Accurate Robotic Pouring Perception using Audio:** We have designed a robotic audio-based pouring system that makes sense of audio perception for robotic pouring height perception to estimate the length of the air column [71]. To the best of our knowledge, we are the first that use audio for liquid height perception. Moreover, we built a large-scale multimodal human pouring dataset containing audio-frequency recordings, liquid real-time weight, force/torque data, video of the pouring motion, and source container trajectories during the pouring.
- **Robust Robotic Pouring Perception using Audio and Force:** While audio can give us a high accurate pouring height perception, a noisy environment will influence the sensing accuracy. We discuss using a multimodal network that could help improve the robustness of the liquid height estimation network, allowing it against noise and against changes in different tasks and varying environments [73]. In addition, the proposed multimodal network is able to reconstruct the shape of the target containers.

1.4 Structure of this Thesis

This thesis contains two main parts, robotic grasping and robotic pouring. The structure of this thesis is shown in Figure 1.3. The rest of this section will introduce and give an abstract of each chapter.

Related Work

- Chapter 2: Related Work. This chapter will introduce the related work on robotic grasping and pouring. We will illustrate the commonly used grasp quality metrics available, model-based grasp detection and model-free grasp detection. Furthermore, the robotic pouring trajectory generation and the liquid height perception methods will be introduced for accurate robotic pouring. The commonly used sensor modalities include vision, audio, force/torque and multimodal fusion.

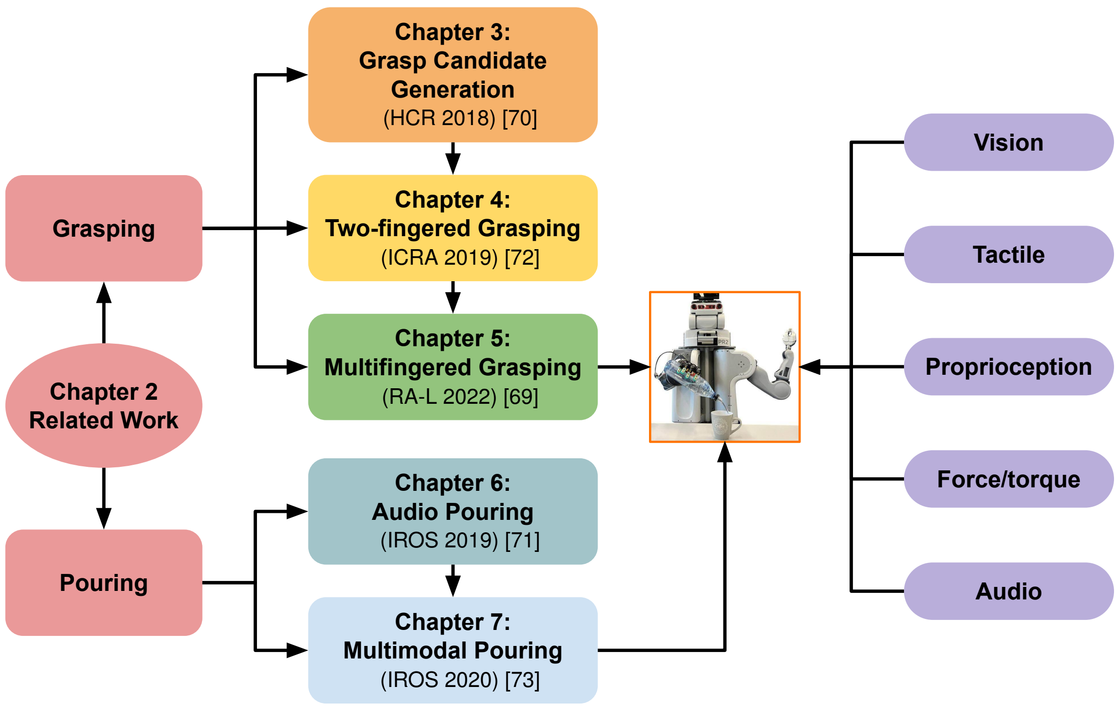


Figure 1.3: Overview of this thesis. The left column shows the two main robotic topics discussed in this thesis. The middle column shows the five main chapters that discuss the main contributions by the author. Chapter 3 [70], Chapter 4 [72], Chapter 5 [69], Chapter 6 [71], and Chapter 7 [73] are based on five papers by the author. The right column shows the sensor modalities used in this thesis.

Robotic Grasping

- Chapter 3: Grasp Candidate Generation. This chapter will introduce the grasp pose generation method GPG [114] based on point cloud input with our modifications to improve grasp candidate generation. This chapter is based on the paper [70].

- Chapter 4: Two-fingered Grasping Using Point Clouds. Based on the grasp candidate generation method in the previous chapter, we propose the two-fingered grasp evaluation method PointNetGPD. It takes point cloud as input, and outputs the grasp evaluation scores. This chapter is based on the paper [72].

- Chapter 5: Multifingered Grasping Based on Multimodal Reinforcement Learning. To carry out useful tasks with the multifingered hand, this chapter solves the basic multifingered grasping problem via RL method training in simulation and adapting to the real world. This chapter is based on the paper [69].

Robotic Pouring

- Chapter 6: Making Sense of Audio Vibration for Robotic Pouring. To pour liquid into a target container, precise perception is needed. This chapter discusses the technique that takes advantage of audio sensing to solve the liquid height perception problem. This chapter is based on the paper [71].

- Chapter 7: Robust Robotic Pouring using Audition and Haptics. The previous chapter discussed liquid height perception using only audio sensor input. This chapter builds upon that and discusses the benefits of multimodal sensory input that combines audio and force/torque input. This chapter is based on the paper [73].

Conclusion

- Chapter 8: Conclusion and Future Work. The final chapter concludes this thesis and lists the findings and outcomes of the Ph.D. study. In the end, potential future work is also proposed to overcome current shortcomings and extend the current state of my work.

Chapter 2

Related Work

As discussed in Chapter 1, achieving a robot serve drinks contains several subtasks, *i.e.*, grasping the bottle, pouring liquid into a mug, grasping the mug, and handing it over to the human. All subtasks needed in this scenario can be abstracted into two topics: grasping and pouring. As the grasping and pouring tasks are the most commonly required, the robotic community is also committed to building versatile service robots with these capabilities. However, in contrast to humans, these straightforward tasks are pretty challenging for robots, from understanding the environment to executing motions. Therefore, many sophisticated algorithms are designed in robotics based on human brains and daily experiences. Many neuroscience and physiology researchers have found that humans always do tasks according to multisensory inputs and condition different sensory inputs accordingly. Multisensory integration is key to how humans process these signals, *i.e.*, vision, sound, touch, smell, and even taste. In grasping motion generation, according to the research on human grasping behavior, we found that human grasping can be sorted into several grasping types [34]. Sensory data like vision, tactile, and proprioception usually work together for grasping detection and motion planning. This chapter will introduce the state-of-the-art research progress on robotic grasping.

Besides grasping, humans are also experts at handling liquids. After training, humans can precisely implement the tasks mentioned above without spilling any liquid. For example, Chinese tea being poured from impressively high up by performers using a long-mounted pot, or cocktails made in fantastic shows by bartenders. This is because humans can precisely condition visual, audio, and force sensing to control the pouring motion. Applying the same sensor modalities to a robot requires the processing of vision sensing, audio sensing, force/torque sensing, and proprioception.

For reference, Section 2.1 gives a brief introduction to robotic grasping basics, including friction cones, force-closure, and grasp quality measures. Section 2.2 introduces the grasp detection method in both model-based and model-free methods for a two-fingered gripper. Section 2.3 introduces the current state-of-the-art of multifingered grasping methods. Section 2.3.1 introduces multifingered grasp detection methods. As the multifingered hand usually has a large action space, it is often efficient to introduce dimension reduction into controlling such a hand. Section 2.3.2 introduces several examples of work that uses dimensional reduction techniques in robotic control, especially in robotic grasping. Robotic pouring has usually been implemented through generating motion trajectories or estimating specified features of the liquids or the containers as the guidance for pouring tasks. As a result, Section 2.4 introduces robotic pouring motion generation methods for a robotic arm. Section 2.5 discusses robot pouring algorithms depending on different modalities, such as vision, audio, and force/torque.

2.1 Grasp Basics

The fundamental of robotic grasping is the static analysis of a grasp. Which is the research field that finds the relationship between the target grasp object and the grasp force generated during the grasp. This kind of research aims to find an efficient method to evaluate whether a grasp is good or not before executing it in the real world. There are three main ways to identify a good grasp:

- Using the geometry method to judge whether a grasp is force-closure or form closure.
- Using a physics simulator to test a grasp to see whether the object falls from the gripper.
- Using a data-driven method to collect a large dataset of grasps containing both positive and negative samples and using a deep neural network to score a grasp candidate.

The geometry method is the fundamental method for evaluating a grasp. This section will introduce this in more detail. The basis of the geometry method is to assume that the hand and graspable objects are rigid objects and use force-closure measures to score a grasp. To make the complicated force-closure theory easier to understand, we discuss the 2D grasp situation and explain what a force-closure grasp is and how to use Grasp Wrench Space (GWS) [58] to quantify the grasp quality.

2.1.1 Friction Cone

We first assume the grasp is a so-called precision grasp where all the contact points with the objects are fingertips. Then the fingers will exert pressure on the object, and the object will be against the fingers with forces in opposite directions. As the finger and object have friction, all possible forces from object to finger lie in a cone around the normal surface. This 3D shape is called a friction cone. If the grasp is simplified to a 2D scenario, the friction cone can be collapsed into a triangle as shown in Figure 2.1. The friction angle γ is the angle that forms the friction cone. It can be defined as $\gamma = \tan^{-1} \mu$ where μ is the friction coefficient.

2.1.2 Force-Closure Grasp

Every contact of a grasp can be described as a 3-dimensional force and 3-dimensional torque, which is called a *wrench*. A grasp with n contact points can be described as:

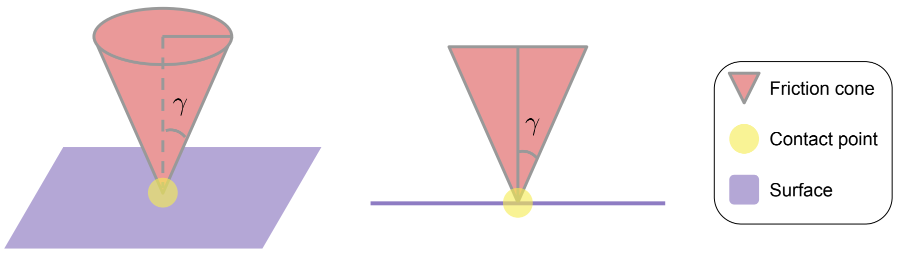


Figure 2.1: Schematic diagram of the friction cone. (left) 3D friction cone. (right) 2D friction triangle.

$w_1, w_2, \cdots, w_n$. Then a stable grasp with some external disturbance w_{ext} must satisfy the following formulation:

$$\sum_{i=1}^{n} w_i + w_{ext} = 0 \tag{2.1}$$

If a grasp with all contact forces within its friction cone and with all contact wrenches combined can balance the external disturbance, we can call it a force-closure grasp [85]. To determine whether a grasp is force-closure by a geometrical method is quite straightforward, as shown in Figure 2.2. The first row shows grasp forces using a black arrow, and the yellow dot is the contact point. The grasp analysis graph is then shown in the second row. The object to grasp is simplified to an origin point (black point in the figure), and the red triangles illustrate the friction cone of the contacts. Then, we can draw a convex hull of the grasp by connecting all of the friction cone's outliers. If the origin is inside the convex hull as shown in Figure 2.2(a)(c), then the grasp is force-closure. Otherwise, the grasp is not force-closure, as visualized in Figure 2.2(b).

2.1.3 Grasp Quality Measure

To evaluate the quality of a grasp, many analytic approaches physically analyze the geometry of the gripper configuration and the object to evaluate the quality of a grasp. Force-closure and GWS analysis are two mainstream grasp quality metrics. The force-closure methods take the friction between the gripper and object into consideration, while GWS works on friction-less cases.

The grasp quality commonly used in the analytical evaluation of a grasp is called grasp wrench space. This concept is formed using the space spanned by all contacts. More specifically, the GWS is the largest inscribing ball of a grasp's convex hull, as the green circle shown in Figure 2.2(a) and (c) indicates. The dimension of this circle (or a ball in a 3D situation) can be taken as the grasp quality. Figure 2.2(c) has a larger circle dimension than (a), so the quality of grasp (c) is higher than that of grasp (a).

However, these analytic methods can only provide reliable grasp quality measurements when a precise object model is available. As a result, they cannot handle raw sensor inputs like camera images and point clouds.

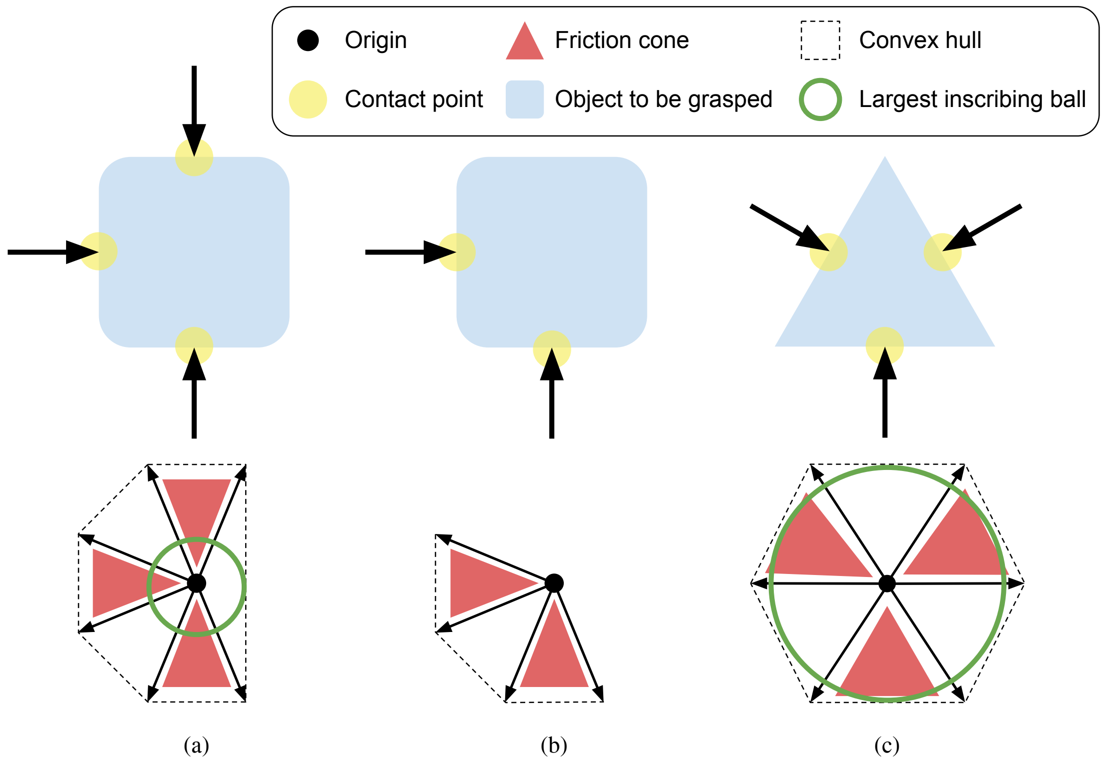


Figure 2.2: Grasp wrench space in a 2D situation. Adapted image from [7], ©2003, IEEE. (a) and (c) are force-closure grasps, while (b) is not force-closure. By comparing the largest inscribing ball in (a) and (c), the grasp with a larger inscribing ball (c) has a larger grasp quality than (a).

2.2 Two-fingered Grasping

Given an object (or a cluster of objects) and essential environmental constraints (location of the collision objects, camera pose relative to the robot base), grasp configuration detection aims to find a gripper configuration that maximizes the grasp quality metrics. The grasp configuration detection problem usually can be divided into two sub-problems: one is **grasp candidate generation** [114] and the other **grasp quality evaluation** [113]. Much work is proposed to improve either one or both problems for better grasp performances. Existing methods for grasp configurations usually fall into one of two categories: model-based or model-free, based on whether the 3D model of the object is known or not. Another difference is that the model-based methods usually need a grasp quality metrics or a human-labeled grasp score to rank the grasps. However, model-free methods usually use a neural network to rank the grasps based on a large grasping dataset labeled by the model-based method.

As a result, both model-based and model-free methods need proper grasp quality metrics to label the grasp quality. Model-based methods use this as a grasping knowledge base for grasp selection or online grasp evaluation. The grasp quality metrics can also be used for data collection and labeling, serving as the training dataset for the neural network.

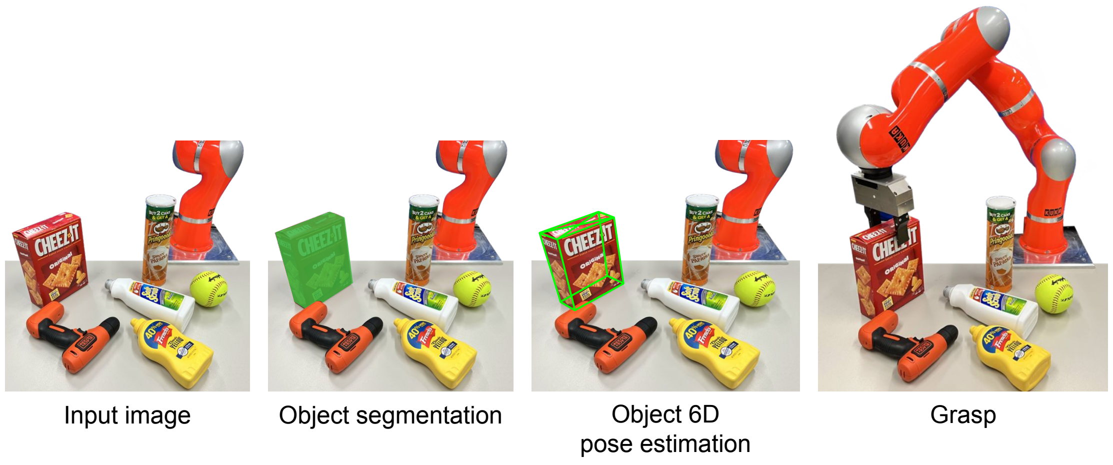


Figure 2.3: Pipeline for model-based grasping. Given an input image, the first step is object segmentation, which recognizes the target object, segments it, and preprocesses it into a clean point cloud. This process can be optimized by using a multi-camera system to get a complete object point cloud. The second step is to get the 6D pose of the object by matching the segmented point cloud with the object model in the dataset. The last step is to match the pre-calculated grasp pose to real robot scenario. The robot in this figure is a Kuka LWR arm with a SchunckWsg50 gripper.

2.2.1 Model-based Grasp Detection

Model-based approaches [131] typically rely on a pre-built grasp database of common 3D object models labeled with sets of feasible grasps and quality metrics provided by assisted tools like GraspIt! [81] or Dex-Net [76]. The pipeline for model-based grasping is shown in Figure 2.3. Miller and Allen proposed GraspIt! for robotic grasping simulation. A Graphical User Interface (GUI) is provided to visualize the grasp. In the backend, GraspIt! samples grasp candidates and calculate the grasp quality. The simulator provides a lot of predefined robotic end effectors, like the Baxter hand and the Shadow hand, and even the human hand, which enables researchers to calculate robotic grasps intuitively. Mahler *et al*. presented Dex-Net, a dataset of 3D object models and the code to calculate grasp quality accordingly. The code can evaluate grasp quality on the run if the object model is known. Both tools use the grasp quality metrics proposed in Section 2.1. Zeng *et al*. [131] proposed a grasp pipeline that first segments and labels multiple views of a scene with a Convolutional Neural Network (CNN) and matches the segmented point cloud with the 3D model in the dataset to calculate the 6D pose of the object. Then they used pre-calculated grasps for the object. So this system is limited regarding the known-object model, a well-performed 6D pose estimation method, and accurate CAD models of the grasped objects. As a result, this system will not be able to generalize to novel objects.

The above work requires 6D object-pose estimation, limiting the method to work on known objects. To grasp the unknown object with a single viewpoint input, Varley *et al*. [118] proposed to use a 3D CNN, which conducts convolution on a voxelized 3D grid from a point cloud to obtain the geometry representation of grasping objects. This representation will then be fed into a grasp generation model. Inferring a grasp with a 3D CNN could improve the analysis of grasp geometry. However, one of the main drawbacks of this method is that the runtime and memory complexity grows cubically with the resolution of the input 3D voxels [91]. As a result, the input will be limited to a pretty low resolution. Furthermore, the sparsity of the point cloud may even distract the neural network from learning meaningful features of grasp geometry since most of the voxels will not be occupied by any points.

After the object is detected and recognized, the grasp needs to be executed. The model-based methods need to associate the sensor input with an object entry in the database for grasp planning during execution. Such matching is mainly based on visual and geometrical similarity [6, 31, 10, 46]. However, due to imprecise sensing and the limited size of the database, model-based methods could arguably have poor generalized performances on novel objects and in environments where objects are presented in dense clusters.

2.2.2 Model-free Grasp Detection

Like model-based methods, model-free methods are usually composed of two parts: grasp candidate generation and grasp quality metrics. However, the difference is that the grasp quality calculation here is usually replaced by a neural network, so there is no need to match the pre-calculated grasp to the robot scene. In the first part, the geometry information captured by sensors will be leveraged as a heuristic or constraint [114] to build an adaptive grasp configuration sampler over the given object. A brief introduction to grasp candidate generation is described in Chapter 3. The quality metrics will then evaluate these grasp candidates. In some recent model-free methods, large grasp datasets are also needed for training better quality metrics based on deep neural networks [75, 41].

One of the challenges to robotic grasping is the uncertainty of perception. Since the robotic gripper needs to interact with the object in a 3D space, a precise and fine 3D visual analysis will be critical for a successful grasp. Motivated by the success of deep neural networks in various 3D computer vision tasks [15, 106, 133, 67], several trials on combining 3D computer vision techniques and grasp planning have been carried out [113, 75, 41, 118, 130].

Mahler *et al.* [75] proposed a grasp quality CNN to predict the grasp quality of the grasp candidates. Grasp candidates are represented by rotating and moving the input image such that the center of the image is the grasp center point and the grasp is aligned with the image width, as shown in Figure 2.4(a). Their work is designed for a top-down grasp where the gripper is perpendicular to the support surface. Work from ten Pas *et al.* [113] designed several projection features on normalized point clouds to construct a CNN-based grasp quality evaluation model called Grasp Pose Detection (GPD) and reach state-of-the-art performance in grasping objects among dense clutter. The sample input of GPD is shown in Figure 2.4(b). However, due to the network architecture and hand-crafted depth features, in our experiments, we found that GPD suffers from severe overfitting and performance reduction when the input point cloud is overall sparse. On the other hand, it could be hard to obtain a relatively comprehensive point cloud in most

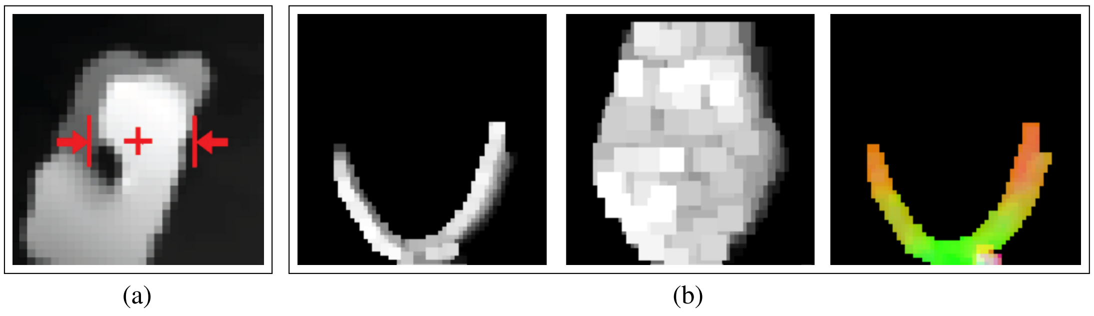


Figure 2.4: Grasp representation for Dex-Net and GPD. (a) Network input for Dex-Net [75]. Reprinted image: ©2017, IEEE. (b) Network input for GPD [113], the detailed method for this representation is illustrated in Section 3.3.1. Reprinted image: ©2017, SAGE Publications.

real-world grasping situations, especially when the clutter is highly occluded.

As shown in Figure 2.4, the approaches mentioned above all represent the input sensor information by 2D images and only take images as the input of the CNN, which may be insufficient for geometry analysis as these kinds of grasp representations may lose some geometry information. By introducing PointNet [89] for 3D representation learning and meticulous grasp quality labels for supervision, point-cloud-based methods [72] can outperform these results regarding both grasping performance and efficiency. Chapter 4 will describe this method in detail.

More recently, with the growing popularity of PointNet and generative models, a large part of research work generates grasps in an end-to-end manner with a single point cloud input. Mousavian *et al.* [84] formulated the problem of grasp generation as sampling a set of grasps using a variational autoencoder [57] and assessing and refining the sampled grasps using a grasp evaluator model. Wu *et al.* [126] further proposed a novel, end-to-end Grasp Proposal Network for 6 Degree of Freedom (DoF) grasp detection. This network has a grasp proposal module that defines anchors of grasp centers at discrete but regular 3D grid corners, which can generate precise and diverse grasps.

2.3 Multifingered Grasping

Multifingered grasping uses a high-DoF hand to plan and grasp the objects. This task is difficult in two ways:

- **The control.** The Shadow hand, an example of a multifingered robot hand, has 24 joints and is controlled by 20 motors. So the action space for controlling the Shadow hand is 20. That is to say, we need to give the hand a command that is composed of a 20-dimensional vector at each control timestep. Even without considering object dynamics, a perfect grasping action based on grasp synthesis for a high-DoF dexterous hand is still a challenging task [92].

- **The design.** Unlike a two-fingered gripper with only one control parameter, multifingered hands are hard to model and represent. Multifingered hands differ considerably in their designs, making it hard to come up with a general representation.

The DoF difference can vary from 4 to 20, which often limits the multifingered hand study to certain kinds of end-effectors.

2.3.1 Multifingered Grasp Configuration Detection

Multifingered grasping has been widely studied for decades. Like two-fingered grasping, it can also be divided into model-based and model-free methods. Two-fingered grasping usually considers two contact points, while multifingered grasping usually considers more than two contacts. Based on the contact points and the precise object model, we rank the grasp quality analytically.

However, the model-based methods in a real environment usually face many uncertainties like the object's physical property, sensor noise, and camera pose noise. Li *et al.* [68] proposed a probabilistic model to address robust dexterous grasping under these uncertainties. Due to the computational expense of multifingered grasp planning, Fan *et al.* [33] proposed a finger splitting strategy to plan precision grasps for multifingered hands from parallel grasps. This work starts from an optimal two-fingered grasp configuration using a finger splitting strategy to optimize the contact points and palm pose. Brahmbhatt *et al.* [8] presented a novel framework for grasp synthesis based on the surface shape and contact map of the target object. They used a human-demonstrated contact map as a constraint to optimize and refine the grasp candidates from GraspIt! [18]. However, ideal contact maps are hard to generate for unknown objects in real robot applications. Kumar *et al.* [61] used human hand motion demonstration to initialize and reduce the search space of their multifingered robot hand. However, this requires pose estimation for the objects to get an object bounding box. Although this work did not use object mesh information on multifingered grasps, the 6D pose estimation part needs the object mesh to match the input with the model in the dataset. In conclusion, the model-based method's common weakness is that it requires prior knowledge of the object model.

For the model-free method, Shao *et al.* [104] proposed a deep-learning-based method to generate grasp points for multifingered robots. It takes the object point cloud, and gripper point cloud generated from Unified Robot Description Format (URDF) as input and outputs the corresponding grasp contact points depending on how many fingers the input gripper contains. However, it is an open-loop grasping method.

As a result, RL is often used in multifingered grasping [3, 4, 37, 13, 125, 79]. Ficuciello *et al.* [37] proposed a synergy-based RL strategy and achieved stable grasps. However, the grasping performance critically depends on the quality of the hand preshaping. Furthermore, they proposed a hand-arm grasp system based on this work [36]. Due to the fact that this method needs to train the model in the real world, the whole grasp pipeline is time-consuming. Chebotar *et al.* [13] trained a prediction model based on tactile information to predict whether a grasp will be successful or not. Then they used this model as guidance for a RL algorithm to perform a regrasp action if a grasp failure was predicted. However, the test objects were rather simple. For training in the simulation, Wu *et al.* [125] proposed to use RL to train a grasp agent that can recover from a failed grasp due to a vision sensor error. The method uses tactile information and proprioceptive observation to output the grasp, lift, and regrasp. The highlight is that the tactile observation is preprocessed to a binary form, making the agent easy to transfer from simulation to real. However, the robot hand they used is a Barrett hand. The three-fingered Barrett hand contains four motors, and each finger is controlled by one motor. The hand also has a spread motion. The hand model is relatively simple, which makes the training of such an agent less challenging.

Merzić *et al.* [79] leveraged the contact force into multifingered grasp learning. The problem is phrased as a finite-horizon discounted Markov Decision Process (MDP). The observations are the object pose and the contact force. They concluded that contact forces could improve the grasping robustness specifically under pose uncertainty. However, transferring the contact force from simulation to the real world is quite challenging as the mapping from the sensor reading to a meaningful physical value is often very time-consuming. In [24, 25], the grasp type of dexterous grasping is studied to reduce the complexity of grasp planning. However, they only considered six grasp types in their work, which is not enough to represent all possible human-like grasp gestures.

2.3.2 Dimensional Reduction for Multifingered Hands

End-effector system like human hand has multiple numbers of actuators. Such a system, in theory, requires a vast amount of computational time for its control. However, the human nervous system can control hands extremely fast. The result from Santello *et al.* [99] indicates that human hand control takes place in a subspace of much lower dimension than the original hand's DoF. Figure 2.5 shows that the human hand grasping motion uses one motion parameter. This result gives us the theoretical basis and confidence to reduce the dimension needed to control the bio-inspired high-DoF robotic hand system. Furthermore, the dimensional reduction is an efficient method to learn a high-level representation of the robot kinematics and dynamics [35].

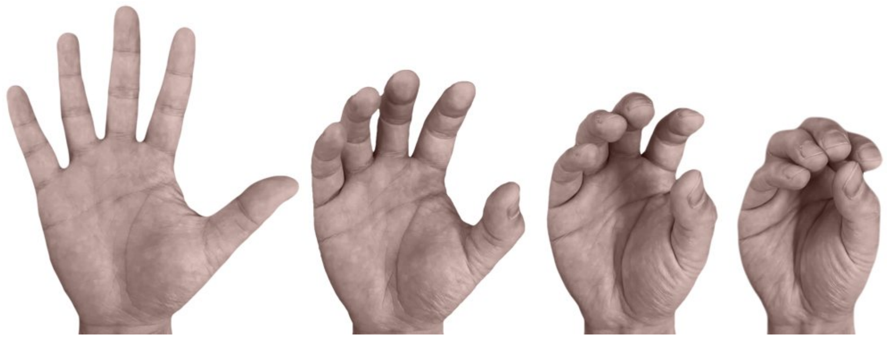


Figure 2.5: Four human hand motions which are only affected by one principal motion component.

Feix *et al.* [34] studied the human grasp and concluded a taxonomy with 33 different grasp types. This taxonomy is commonly used to define the natural human grasp types. Ciocarlie *et al.* [19] proposed to use dimensional reduction to control artificial hands for a given task. They brought up the concept of *eigengrasps* to define the grasps using a subspace. They then used *eigengrasps* to control four different hand types like the Baxter hand and the Shadow hand. The computational advantage is shown that a reduced dimensionality framework can be used in an interactive grasping system. Wimböck *et al.* [124] further applied the data reduction techniques to the DLR Hand II. A synergy impedance controller was derived and implemented using only two principal components. Bernardino *et al.* [5] used a data glove to teleoperate a real Shadow hand to grasp different objects using eight different grasp types. The collected data was used to compute *eigengrasps* using a linear mapping method, Principal Component Analysis (PCA). Humans automatically compensate for the mapping error between the Shadow hand and the human hand. The same method was further applied to the iCub hand. More recently, Starke *et al.* [108] studied human grasping data and used an autoencoder to learn a three-dimensional latent space for grasp representation. Compared to previous work that uses linear mapping like PCA for dimension reduction, their work uses a deep autoencoder that can better extract the human grasping synergy information. However, the above approaches did not provide an autonomous method to control this low-dimensional subspace. Ficuciello *et al.* [37] also use grasp synergies were used to reduce the dimensions needed for grasp planning. However, the Schunk Hand they used has 20 joints which are actuated by 9 motors. Thus, the action space to control this hand has been mechanically constrained. Instead, we concentrate on the Shadow Hand, which has 18 motors to control 22 finger joints and 2 motors for 2 wrist joints. It is difficult to control this hand by using traditional methods. A detailed discussion of the multifingered grasping we did using a Shadow hand can be found in Chapter 5.

2.4 Pouring Motion Generation

Pouring motion generation belongs to a subset of trajectory planning but with more constraints than the basic collision-free motion planning. It is also constrained by the liquid dynamics and properties of deformable objects (liquid in this case). With different liquid properties, the pouring strategy would be different. Figure 2.6 shows an ideal pouring motion of the source container. If the workspace allows, this motion can be simplified to a motion in a 2D plane that is perpendicular to the support surface.

Brandi *et al.* [9] suggested learning pouring tasks using kinaesthetic teaching. They then generalized pouring actions by warping the parameters from known containers to unknown containers. The proposed method can be used to generate motor control sequences for robotic pouring.

Langsfeld *et al.* [62] achieved dynamic pouring tasks from human demonstration. This work used an Imitation Learning (IL) approach to the pouring motion generation task. Instead of using Learning from Demonstration (LfD) that uses kinesthetic demonstrations, IL can learn how humans recover from failed attempts to perform compliant manipulation tasks. A robot experiment used a Baxter to pour water into a target container on a rotating table.

Yamaguchi *et al.* [129] solved the general pouring task using a skill library. This library stores different behaviors of human demonstration for flow control, such as tipping, shaking and tapping. Then a learning framework was proposed to select from these pouring behaviors. In the end, they used a PR2 robot for the pouring demonstration.

Since transferring compliant manipulation skills from humans to robots is always

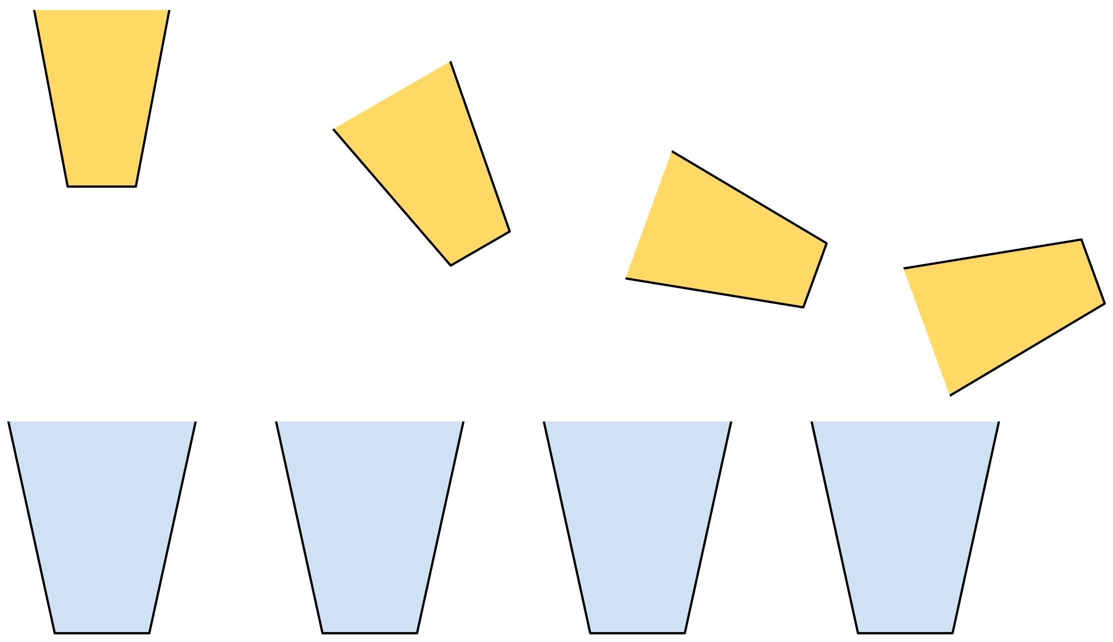


Figure 2.6: An ideal pouring motion of the source container.

tricky, Pan *et al.* [86] solved the online trajectories of the source container in simulation using a simplified dynamic model for fluid constraint and a receding-horizon optimization method to handle the fluid dynamics. Due to the simplified model, the simulator can only work in a pouring simulation with mild speed. However, the comparative experiment with a full-featured Computational Fluid Dynamics (CFD) simulation shows that in mild speed pouring, the simplified dynamic model can be used to replace the time-consuming CFD simulator.

More recently, Sermanet *et al.* [103] proposed to use an unsupervised perceptual procedure for automatic reward function generation. The visual inputs are processed into an intermediate abstraction then are used to get a smooth and dense reward function in their work. Do *et al.* [28] solved this problem by learning a pouring policy using Deep Deterministic Policy Gradient (DDPG) in simulation and transferring the learned policy from simulation to a real robot.

2.5 Liquid Perception for Robotic Pouring

Other researchers focus on using different modalities as input to detect viscosity [32], height [29], the volume of the liquid or granular material [20], thus relying on the perception results to perform pouring by a simple controller on the real robot. We focus on height perception in robot pouring, as height perception is essential for a robot to pour precisely into a target container. The rest of this section will introduce different modalities used for liquid height perception.

2.5.1 Visual Sensing

Vision is one of the most commonly used modalities regarding the pouring in human daily life and robotics. Obviously, vision-based perception strongly depends on the lighting conditions, the color of the liquids, and the shape of the target containers. Pithadiya *et al.* [88] compared several edge detection techniques for the filling height inspection of convex target containers. Do *et al.* advocated a probabilistic approach to estimate the liquid height based on an RGB-D camera [29]. They further switched the analytical estimation approaches depending on the types of liquid and utilized a Kalman filter to deal with the uncertainties of the vision data [27]. However, the mean height errors for ten pourings of 3 transparent liquids were larger than 4 mm. Instead of directly predicting the absolute height of the liquid, another popular method estimates the input volume of the liquid by analyzing the visual information of the water flow [101]. Schenck *et al.* [101] used a thermal camera to generate pixel-level ground truth data of heated water using thermal imagery. The estimation result was used to determine the water volume using both a model-based and a neural network method. However, this method suffers from poor estimation error due to the varied liquid types and the complex shapes of water flow. Figure 2.7 shows the network architecture of this work.

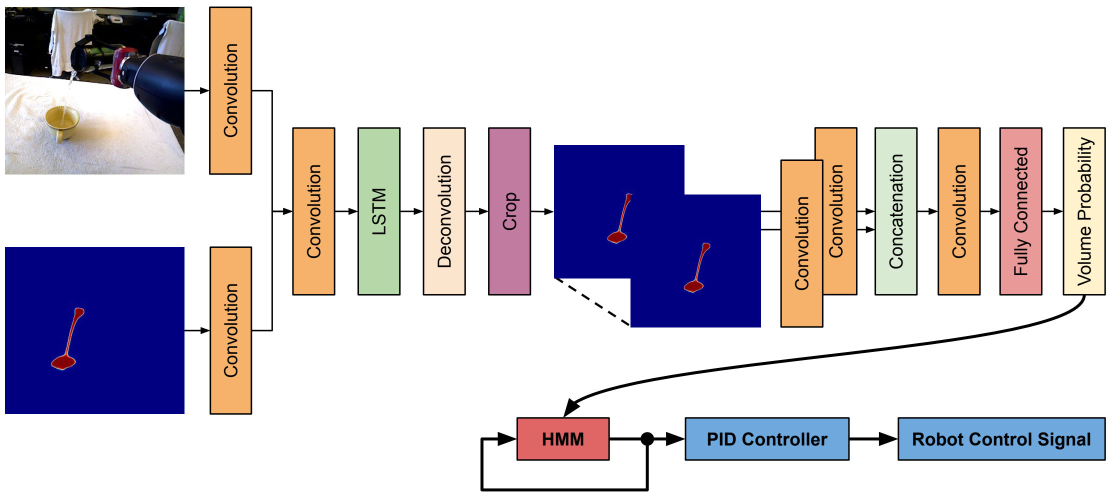


Figure 2.7: Robot control system of a vision-based pouring architecture adapted from [101]. Firstly, the pouring systems feeds an RGB image into a CNN and outputs a volume probability at each timestamp. Then, it uses Hidden Markov Model (HMM) to estimate the volume of the container. In the end, a proportional-integral-derivative (PID) controller is used to compute the control signal for pouring. ©2017, IEEE.

To predict the absolute height of the liquid, Dong *et al.* [30] used a point cloud to model the target container and a proportional-derivative controller to perform the pouring action. Do *et al.* [27] utilized a Kalman filter dealing with opaque and transparent liquids based on an RGB-D camera. However, the mean height errors for ten pourings of three transparent liquids were larger than 4 mm.

2.5.2 Audio Sensing

The auditory information embodies sustainable clues when liquid or granular materials interact with other objects, such as the frequencies and vibrations of the liquids and the air. Griffith *et al.* [40] indicated that auditory and proprioceptive data enhance the classification tasks for the interaction between objects and water. Sakiko *et al.* [52] verified that the vibration when pouring liquid out as an audio-haptic rendition significantly affects the amount of liquid poured. Clarke *et al.* [20] used audio-frequency vibration generated by shaking the granular material to evaluate the weight that poured out. However, the weight of the poured-out granular materials does not allow estimating the target container's filling height. Nevertheless, granular materials and liquids also have entirely different properties, which is hard to transfer. None of the above work solves the height regression problem in robotic pouring by exploiting audio vibration of the air in the target container. As a result, in this thesis, we would like to study the use of audio sensing in liquid height perception.

2.5.3 Haptic Sensing

Haptic sensing, especially force and torque sensing, is also popular in robotic pouring perception. Specifically, force data is exerted to generate pouring trajectories by predicting the angular velocity of the pouring container in simulation [50]. Rozo *et al.* [94] used a parametric hidden Markov model to retrieve joint-level commands given the force-torque inputs from the human demonstration. Saal *et al.* [97] examined the viscosity estimation of the various liquids from tactile sensory data. Although force from the pouring container could explicitly represent the volume of the poured liquid, force cannot measure the liquid height in an unseen target container. The goal of learning robust robot pouring from human demonstrations, including hand trajectories and force/torque data, is explicitly discussed by Huang and Sun [51], as part of their multitask dataset on daily human interactive manipulation. Matl *et al.* [78] utilized a force sensor mounted at the end of a robotic arm with a container grasped by a gripper to get the change of wrenches, and they used a physics-based model to estimate the mass and volume of the liquid. In the above three papers, perception happens before the pouring action begins. Although the force signal from the pouring container can explicitly represent the volume of the poured-out liquid, it cannot measure the liquid height in an unseen target container.

2.5.4 Multimodal Fusion

Recently, many multimodal approaches have been applied in various fields. For instance, in the field of automatic speech recognition, Afouras *et al.* [1] presented a deep audio-visual speech enhancement network to separate a speaker's voice. Furthermore, by utilizing vibration and force feedback, Zhang *et al.* [132] proposed a robust slicing approach to robotic cutting. Moreover, Lee *et al.* [65] implemented a neural network leveraging multimodal feedback from vision and touch for contact-rich manipulation tasks.

Nonetheless, the multimodal approaches work well in a wide range of tasks. However, in pouring tasks, multimodal neural networks are rarely used.

In robotic pouring, recent studies have shown that multimodal sources represent the environmental features better than a single modality in pouring tasks [98]. Wu *et al.* [127] presented a hierarchical Long Short-Term Memory (LSTM) [48] model, which could detect if a pouring sequence was successful. This model is based on a pouring dataset, including visual sequences and IMU data. However, this work only classifies whether a pouring action was successful but cannot predict the precise height of the poured liquid. Wilson *et al.* [123] implemented a multimodal CNN to fuse audio and visual data to predict the weight of the poured liquid, detect overflow and classify the liquid and the target container. Park *et al.* [87] proposed another interesting robot application for anomaly detection during robot manipulation using haptic, visual, auditory, and kinematic sensing. As described in Section 2.5.2, the audio-only network failed to work in a noisy environment or when pouring liquids with high viscosity as the audio signal is weak in these conditions. Based on how humans rely on the correlations of haptics and audio while pouring, in [73], we take advantage of audio and force/torque data as the input to promote the robustness of robotic pouring (Chapter 7).

Chapter 3

Grasp Candidate Generation

For a robot to grasp novel objects autonomously, the standard approach is to generate grasp candidates according to vision sensor input (RGB image, depth image, or point cloud). An analytical or data-driven-based method is used to evaluate the grasp candidates, based on which the robot executes the best grasp candidate out of the evaluation procedure.

This chapter will first introduce one of the grasp candidate generation algorithms in Section 3.1. This section introduces the grasp generation algorithm (GPG) proposed by ten Pas *et al.* [114] to perform heuristic grasps sampling using geometry from the point cloud input. Then some modifications [70] to improve GPG algorithm are introduced in Section 3.2. Finally, in Section 3.3, several robot experiments are conducted to verify that the modified grasp pose generation method can acquire more grasp candidates and thus help to improve the performance of the later grasp evaluation network.

3.1 GPG Algorithm

Recall that in Section 2.1, we introduced the definition of friction cone, force-closure grasp and GWS. The grasp candidate generation provides a large set of grasp candidates, which we hope most of them are force-closure grasps. GPG is the algorithm to generate such grasps.

3.1.1 Grasp Simplification

In this chapter, we focus on the two-fingered grasp candidate generation. So the geometry of a gripper can be simplified to a shape that consists of three cuboid shapes as shown in Figure 3.1. The grasp $g \in G$ can be defined by four parameters, *i.e.*, finger width, finger height, gripper outer diameter, and finger depth where G is the set of grasp candidates. Note that the gripper outer diameter is the maximum distance a gripper can open.

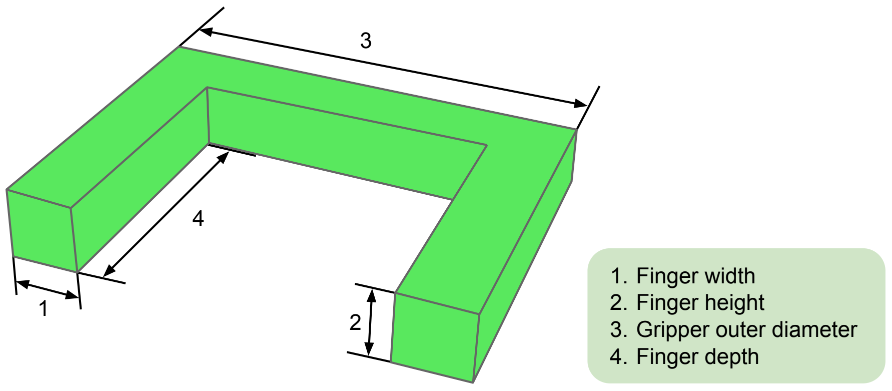


Figure 3.1: Geometry simplification of a two-fingered gripper.

3.1.2 Antipodal Grasp Simplification

As described in Section 2.1, the antipodal grasp is defined as if a grasp has two contact points with an object, then the line connecting these two points is inside the two *friction cones* generated by the two contact points.

However, the GPG algorithm uses a modified definition, *i.e.*, the grasp is not closed and does not have contact with the target object when generating the grasp candidate. Instead, it opens the gripper to its *gripper outer diameter* and expects the gripper to be antipodal when it grasps. The benefit of this is that we can check the collision when the gripper is not closed. That can make sure the whole grasping motion is collision-free until the gripper contact some point on the object.

3.1.3 Darboux Frame and Principal Curvature

The GPG does not sample grasps directly from $G \subseteq SE(3)$[1], which would be very inefficient as a sample in $SE(3)$ will result in many useless grasps far away from the object. Instead, GPG samples grasps depending on the object geometry using point cloud as input to ensure every grasp is relevant to the object geometry. The geometry knowledge used for GPG is the Darboux frame and principal curvature.

In differential geometry, each point on a 3D surface has two principal curvature directions: main principal curvature direction and minor principal curvature direction. These two directions are calculated by the PCA that represents how differently the surface is bent in different directions at the given point. The point normal direction and two principal curvature directions can form a frame. This frame is also called a Darboux frame, as shown in Figure 3.2. Although any orthogonal frame of the surface is a Darboux frame, GPG only takes the one consisting of the principal curvatures, which is the orientation of a grasp candidate. Note that the calculation of the Darboux frame requires the knowledge of the normal direction of the selected point on the object surface. Usually, estimating the normal direction of a surface is a trivial task as it is easy to calculate a perpendicular vector to a flat surface; however, estimating it using a point cloud directly is significantly more complex as a point cloud is acquired from the depth

[1]$SE(3)$ is the Euclidean group of rigid body displacements in three-dimensions.

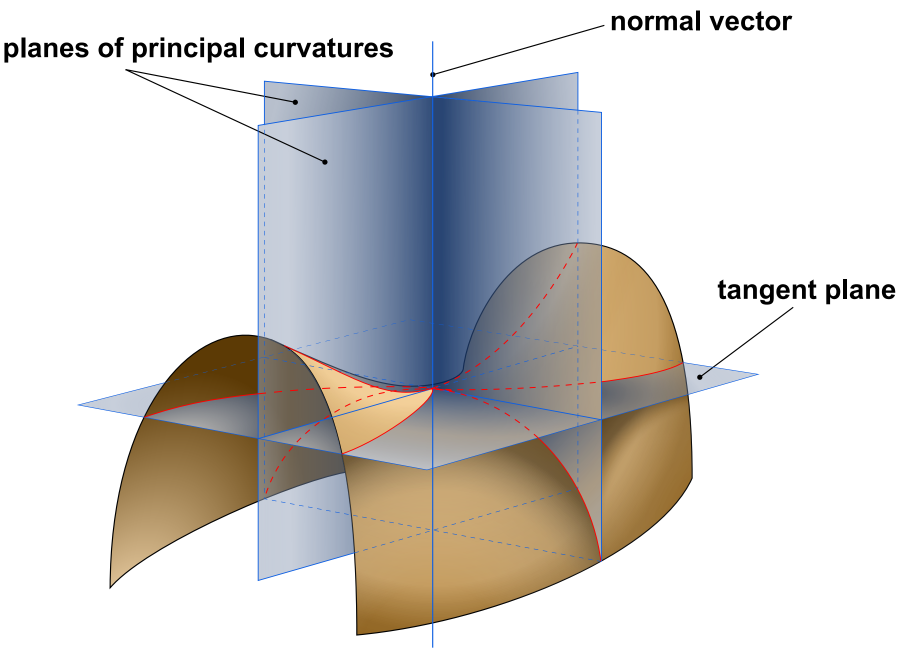


Figure 3.2: Surface curvature planes (Darboux frame). Reprinted image from [39], ©2016 CC-BY-SA Wikipedia.

sensors are only samples of a real surface. One way to overcome this is to convert the point cloud to a surface and then calculate the point normal. However, converting a point cloud to a surface usually requires some manual modifications and fine-tuning. So for code automation and efficiency, GPG chooses to use point cloud to estimate normal direction directly. The method used to estimate point cloud normal is from [112] to be robust to local surface discontinuities.

The generated grasp frame g is shown in Figure 3.3. The x axis is the normal direction of the sampled point in the input point cloud. The y axis is the main principal curvature direction, also called the binormal direction, and the z axis is the minor principal curvature direction.

3.1.4 Grasp Generation Using Geometry

The above section describes how to form a grasp candidate, and this section will illustrate how to sample as many grasp candidates as possible.

1. Process the input point cloud $\mathcal{C}$ using voxelization and workspace limits. Voxelize the point cloud would help distribute the input point cloud evenly in the 3D space to avoid sampling in only certain areas.

2. Sample n points $p \in \mathcal{C}$ evenly and randomly.

3. Estimate the normal direction of the input n points.

4. Calculate the Darboux frame of the n input points.

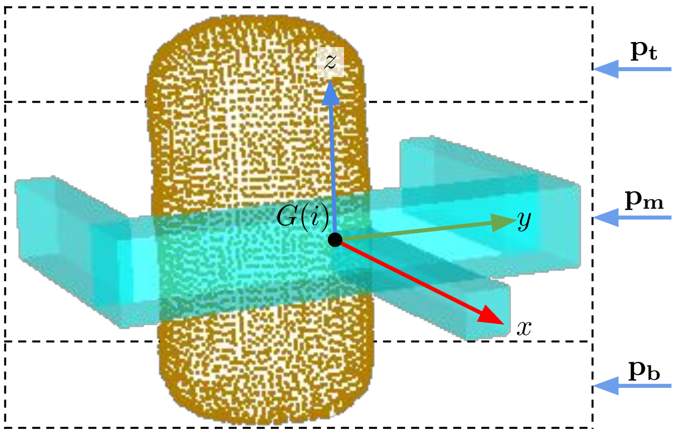


Figure 3.3: Grasp coordinate frame $G(i)$ in sample point i (block circle in the figure) with grasp superimposed at the origin. $\mathbf{p_t}$, $\mathbf{p_m}$ and $\mathbf{p_b}$ are the top, middle and bottom areas of the point cloud.

5. Grid search based on the n Darboux frame.
    (a) Based on the sampled Darboux frame, make a 2D grid that contains (Φ, X), where Φ denotes a discrete set of orientation, and X denotes a discrete set of grasp positions (The values are set to 8 and 20 respectively in GPG).
    (b) Loop around all the combinations of Φ and X, and push the gripper until collision, then count the final result as a grasp candidate.
6. Collision check for unsuitable grasp candidates removal.
    (a) Remove grasp candidates that have a collision with the input point cloud $\mathcal{C}$.
    (b) Remove grasp candidates that the grasp closing area does not contain any point. The definition of the grasp closing area is shown in Figure 4.5(b).
7. If the desired grasp candidate number is not achieved, go to item 2 and continue the grasp sampling procedure.

3.2 GPG Algorithm Improvements

This section describes four modifications for the generation process that facilitates the generation of more high-quality grasp candidates.

3.2.1 Attentional Sampling Process

As noted in GPG, top grasps show a higher success rate than side grasps in cluttered scenes. Also, grasps are not allowed to collide with the support surface. Whereas previous work exploited this information only during candidate selection, we modify the sampling process already. We divide the workspace $\mathbf{p}$ into three parts, $\mathbf{p_t}$, $\mathbf{p_m}$ and $\mathbf{p_b}$, where $\mathbf{p_b}$ denotes the point cloud from the bottom up to the gripper height of $\mathbf{p}$, $\mathbf{p_t}$

denotes a similar margin of the point cloud from the top of $\mathbf{p}$, and $\mathbf{p_m}$ denotes the remaining part of $\mathbf{p}$ excluding $\mathbf{p_b}$ and $\mathbf{p_t}$. If the height of $\mathbf{p}$ exceeds three times the gripper height, we uniformly sample 60% of all candidate points randomly from $\mathbf{p_t}$ and the rest from $\mathbf{p_m}$. Otherwise, we uniformly sample all points from $\mathbf{p_t} \cup \mathbf{p_m}$. Figure 3.3 illustrates the different parts of a point cloud, and an example grasp pose G with the grasp superimposed at the origin. As demonstrated in Figure 3.4, this process samples more candidates in the top region of the point cloud.

3.2.2 Three-dimensional Grid Search

For each point i that is sampled from the input cloud, a local reference frame $G(i)$ is computed. To improve the variance of generated grasps for this point, GPG performs a two-dimensional grid search along the y axis and around z. We extend this grid search to three dimensions and include rotations around the y axis to cover even more local variations. Figure 3.4 illustrates positive grasp candidates generated for five sample points by the two methods. As can be seen, the extended grid search in Figure 3.4(a) yields a higher number of grasp candidates, which are better distributed over the graspable object surface. Moreover, Figure 3.4(b) shows the original GPG output grasps. Even though the three-dimensional grid search also generates several infeasible grasps that have to be removed later, some of these will be corrected in the next step, and some remain to support the neighbor-based selection described below.

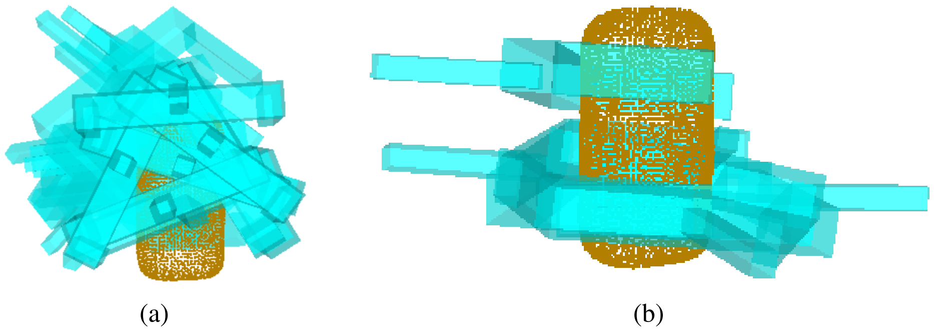


Figure 3.4: Candidate grasps generated by different methods. (a) Grasps sampled by attentional sampling and 3D grid search method (ours). (b) Grasps sampled by uniform sampling and the 2D grid search method (original GPG).

3.2.3 Projection of Invalid Candidates

One remaining issue with the generated candidates is that while they respect possible collisions of the fingers and the environment, they do not always allow for feasible gripper approaches. In particular, the system generates grasps that approach objects from below, enclosing an angle of less than $90°$ between the approach direction and the normal of the support surface. In these cases, we project the infeasible candidates to align their approach directions to the support surface. This is illustrated in Figure 3.5.

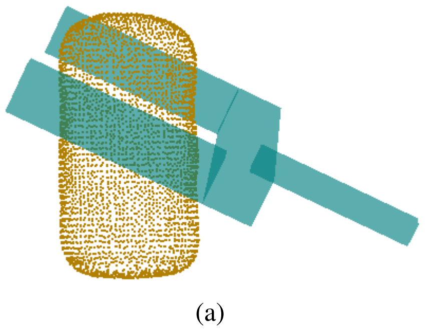


Figure 3.5: Example of invalid candidate projection. (a) A grasp candidate generated by original GPG. (b) The candidate projected to the support surface.

3.2.4 Conservative Approach Depth

To fully define the sampled grasp, the last missing parameter should specify how far the respective grasp point should penetrate the volume inside the gripper. In the original GPG, the gripper is pushed forward until it collides with the perceived point cloud to generate a maximally stable grasp that encloses as much as possible of the object with the gripper. While this strategy can generate very safe grasps, it mainly applies to two-finger parallel grippers with limited finger length. When the length of the fingers exceeds the typical size of individual objects in a scenario, the fingers will often collide with unobserved parts of the scene and more often pick multiple objects in unstable grasps. Additionally, most adaptive grippers focus on a stable grasp volume between the fingertips but not necessarily within the whole volume inside the gripper. This is the case for the Robotiq 3-finger adaptive robot gripper visible in Figure 3.7(a) when used for precision grasps.

In order to avoid these problems in a general way, we compute the approach depth by pushing the gripper further forward either until there is no collision with the perceived point cloud or until no further points are added to the volume inside the gripper. Figure 3.6 contrasts both criteria. The modified strategy in Figure 3.6(a) is less aggressive and keeps the fingers of the gripper further away from possibly unseen obstacles.

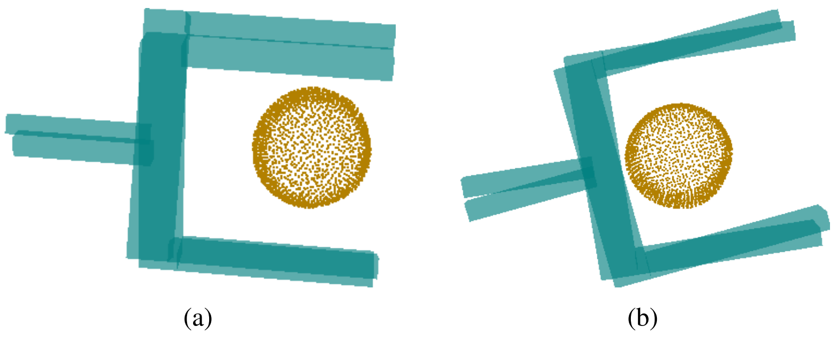
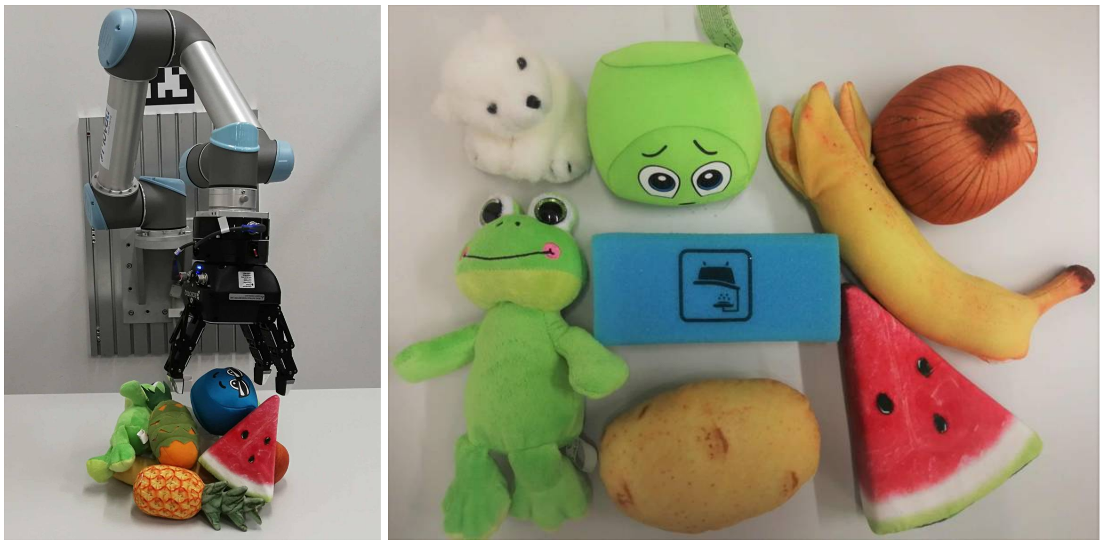


Figure 3.6: Variations of approach depths for grasp candidates. (a) Conservative strategy and (b) GPG's greedy strategy.

3.3 Robot Experiments

To validate the efficiency and flexibility of the improved grasp generation algorithm, experiments were conducted using two conditions: objects were presented to the robot in isolation and a cluttered scenario. The experiments were carried out on a UR5[2] robotic arm with an attached Robotiq 3-finger adaptive robot gripper[3]. We manipulated eight different, previously unseen soft toys with sizes between 3 cm and 10 cm. The robotic setup and the objects can be seen in Figure 3.7. A Kinect2[4] RGB-D camera captures input point cloud data from one fixed view. The whole system is implemented using the Robot Operating System (ROS) framework [90] and was tested on an Intel i9-7900X 3.30 GHz system with 128 GB RAM. Inverse kinematics uses an analytical solver based on IKFast [26], and motion planning was implemented via the MoveIt library [21].



(a) (b)

Figure 3.7: Experimental setup for grasping unknown soft objects. (a) Grasp in dense clutter. (b) Soft objects used in our experiment.

3.3.1 Brief Introduction of GPD

As this chapter only compares the grasp generation performances, we use Grasp Pose Detection (GPD) algorithm [113] as our grasp evaluation backend. Section 3.1 illustrated the steps needed to generate the grasps using the GPG algorithm. This section will introduce how the grasp evaluation algorithm GPD works.

1. Generate: Generate a set of grasp candidates using GPG as illustrated in Section 3.1.

[2]`https://www.universal-robots.com`
[3]`https://robotiq.com/products/3-finger-adaptive-robot-gripper`
[4]`https://developer.microsoft.com/en-us/windows/kinect`

2. Encode: Represent and encode the grasp candidates into several 2D images.
3. Score: Use a CNN to score the represented grasp candidates.
4. Select: Choose the best grasp from the scoring result.

The encoding process converts input point cloud and a 6D grasp pose into 2D image representations. As shown in Figure 2.4(b) on page 15, the 6D grasp pose was represented as three 2D images, which represent one projection direction where the first two images have one channel each, and the third image have three channels. After repeating this process in all three projection directions, 15 channels of information have been acquired.

In detail, the point cloud of grasp closing region $C(h)$ is scaled into a grid of $M \times M \times M$ dimension. Then the point cloud in $C(h)$ should be mapped into one of the M^3 grid cells. $V(x, y, z) \in \{0, 1\}$ denotes whether the grid cell is occupied, $U(x, y, z) \in \{0, 1\}$ denotes whether the grid cell has been observed. Moreover, $\hat{n}(x, y, z) \in S^2$ denotes the surface normal that points outwards from the grid cells, where S^2 is the surface of the input point cloud. With the symbols defined above, we can define the grasp representation as follows. In the grasp frame, as shown in Figure 3.3, in each coordinate axis, the grid cells can be mapped into an $M \times M \times \alpha$ tensor, where α is the channel number. For each of the three projection directions, GPD projects three images. For the projection of the (x, y) plane as an example, we can calculate the three images I_o, I_u, I_n as shown in Equation 3.1.

$$\begin{aligned}
I_o(x,y) &= \frac{\sum_{z\in[1,M]} zV(x,y,z)}{\sum_{\in[1,M]} V(x,y,z)} \\
I_u(x,y) &= \frac{\sum_{z\in[1,M]} zU(x,y,z)}{\sum_{\in[1,M]} U(x,y,z)} \\
I_n(x,y) &= \frac{\sum_{z\in[1,M]} \hat{n}(x,y,z)V(x,y,z)}{\sum_{\in[1,M]} V(x,y,z)}
\end{aligned} \tag{3.1}$$

In the Equation 3.1, I_o and I_u are $M \times M \times 1$ tensor, and the I_n is $M \times M \times 3$ tensor. In which, I_o represents the averaged heightmap of the occupied points, I_u represents the averaged heightmap of the unobserved region, and I_n represents the averaged surface normal. Please refer to the original paper [113] for more detailed explanations.

For the scoring, a four-layer CNN is used to get a single-value output to represent the grasp quality. The CNN structure used in GPD is same as LeNet [63]. The first two layers are two combinations of a convolution layer followed by a pooling layer. The third layer is one inner product layer with a rectified linear unit as output, and the fourth layer is one inner product layer with a softmax as the output. The final output is a value that represents the grasp quality and with a selected threshold for binary classification.

In the end, a selection will have to be made to provide the robot with a grasp to execute. All the grasp candidates above the good grasp threshold are sorted from high score to low score, and then we use MoveIt to check whether the grasp is collision-free and executes the best grasp if the grasp trajectory is possible.

3.3.2 Objects Presented in Isolation

We ran ten trials for each object presented in isolation on the table. The results shown in Table 3.1 demonstrate that our improvements result in more grasp attempts and a higher success rate. Note that the number of attempts of some low objects (in this dataset, the potato and the banana) is zero. In these cases, we let the algorithm run for five minutes, but no actionable grasp was generated. This is mainly because all generated grasp candidates collided with the table below the object. In contrast, the conservative approach depth avoids these collisions and manages to grasp the objects.

Table 3.1: Grasp experiments on isolated object

Objects to pick		Ball	Potato	Frog	Sponge	Onion	Banana	Melon	Bear	Total
GPD	Success / Attempts	8/10	0/0	7/10	7/10	10/10	0/0	5/10	9/10	46/60
	Success rate	80%	Null	70%	70%	100%	Null	50%	90%	76.7%
Ours	Success / Attempts	10/10	10/10	8/10	10/10	10/10	7/10	9/10	8/10	72/80
	Success rate	100%	100%	80%	100%	100%	70%	90%	80%	90%

In Table 3.1 considering the eight failures in our method, four were due to inaccuracies of collision objects in the MoveIt planning scene, and three were due to the collision of the fingers with the object prior to forming a grasp, one was due to point cloud registration errors or inaccuracies in the kinematic calibration of the robot.

3.3.3 Objects Presented in Dense Clutter

We randomly put all objects in a heap in a dense clutter condition and let the system pick them successively. We ran ten trials in this task, and the results can be found in Table 3.2. Our method achieved a success rate of 91.1% for attempted grasps and picked 78.8% of all objects. For both metrics, this outperforms the original GPD implementation by at least 20%.

Note that the number of successes differs from the number of grasped objects in both cases. In some cases, the gripper grasps more than one object at once, but as there is no concept of an object in the entire grasp generation process, this is a common phenomenon. That is because we do not segment the point clouds when we preprocess the point clouds. In Table 3.2 considering the five failures in our method, three were due to inaccuracies of collision objects in the MoveIt planning scene, and two were due to the collision of the fingers with the object prior to forming a grasp.

Table 3.2: Grasp experiments on clutter objects

	Success / Attempts	Grasped Objects / Total Objects	Success Rate	Completion Rate
GPD	32 / 45	44 / 80	71.1%	55%
Ours	51 / 56	63 / 80	91.1%	78.8%

3.4 Discussion and Summary

In this chapter, the grasp candidate generation algorithm was introduced, and its modifications were proposed. Robotic experiment shows that The experimental results demonstrate that our improvements can increase the number of attempted grasps, as well as achieve higher success rates for attempts and completion rates for pick tasks. Overall the GPG improvements generate more robust and reliable grasp candidates.

Even so, it leaves many alternative pathways untouched. While sampling of candidates allows for more intuitive internal dynamics, it also requires a lot of computation. Training function approximators to generate grasps of similar quality seems a promising approach to reduce overhead, yet it remains a difficult area in practical applications. Creating 3D occupancy grid maps will help reduce the grasp failure caused by the arm and gripper's accidental collision with other objects. Moreover, the objects used in our experiments are all soft objects that are usually easy to grasp. Furthermore, the grasp evaluation network uses GPD, which is not using the point cloud directly and may lose information while processing the point cloud into 2D projection.

In the next chapter, we will design a novel grasp evaluation network that takes the 3D point cloud as input and tries more challenging objects and dense clutters in real robot grasping experiments.

Chapter 4

Two-fingered Grasping Using Point Clouds

To enable the service robot to complete autonomous grasp tasks, two-fingered grasping is the fundamental technique that a robot needs to learn. Chapter 2 introduced the grasp theory and Chapter 3 introduced the grasp generation method. However, the grasp evaluation network used in the robotic experiment is GPD, a network that uses the 2D project of the 3D point cloud from different hand-crafted viewpoints as input. However, the projection may lose some important 3D information from the scene. This chapter will propose an end-to-end grasp evaluation network (PointNetGPD) to address the challenging problem of localizing robot grasp configurations directly from the point cloud. Compared to recent grasp evaluation metrics that are based on hand-crafted depth features and a CNN, PointNetGPD is lightweight and can directly process the 3D point cloud that locates within the gripper for grasp evaluation. Taking the raw point cloud as input, PointNetGPD can capture the complex geometric structure of the contact area between the gripper and the object even if the point cloud is very sparse. To further improve PointNetGPD, we generate a large-scale grasp dataset with 350k real point clouds and grasps with the YCB object dataset [11] for training. The performance of the proposed model is quantitatively measured both in simulation and on robotic hardware.

The structure of this chapter is as follows, Section 4.1 motivates why we need a new grasp evaluation network and illustrates our main contributions. Section 4.2 introduces the main problem of grasp evaluation and the main challenges. Section 4.3 explains the generation procedure of a new grasping dataset. Section 4.4 mentions the network architecture for grasp quality evaluation. Section 4.5 presents the network evaluation compared with the baseline method. Section 4.6 reports the robot experiments results that show our network can outperform other CNN based networks. In the end, the conclusion and future work of this chapter presents in Section 4.7.

4.1 Introduction

Planning a grasp under uncertainty is a difficult task in robotics. For a robot that operates in the real world, uncertainty may come from varied aspects. This chapter mainly

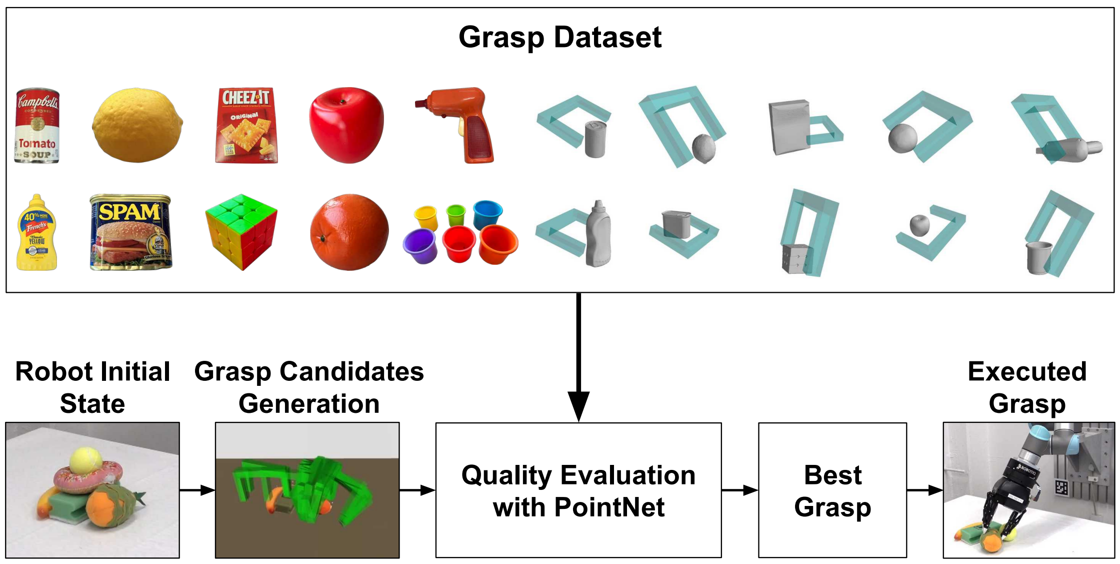


Figure 4.1: An illustration of the proposed PointNetGPD for detecting reliable grasp configuration from point clouds. Taking raw sensor inputs from a common RGB-D camera, the depth maps will be first converted into a point cloud. Then several grasp candidates will be sampled with essential geometry information as heuristic or constraints. For each candidate, the point cloud within the gripper will be cropped and transformed into local coordinates and finally fed into the grasp quality evaluation network PointNetGPD. The grasp with the highest score will be executed. The model is trained with a large-scale grasp dataset based on the YCB object dataset [11].

concentrates on the uncertainty caused by the imprecision and deficiency in sensing. This kind of uncertainty is usually associated with the sensor we use for robotic perception [119]. To address this problem, a grasping model that can work with raw sensor input is needed. Some recent advances suggest to use deep neural networks that have been trained on large-scale grasp datasets labeled by humans [76, 75] or grasping outcomes done by robotic hardware [41, 42] to plan grasps directly with sensor input like RGB images [66], depth images [75] or point cloud [114]. Such research work yields promising results across a wide variety of objects, sensors, and robots, and their models generalize well to novel objects that are not present in the training set. However, most of the current methods still rely on 2D (image) or 2.5D (depth map) input; some grasping models even require complex hand-crafted features [113] before they can process the data, while very few of them will take the 3D geometry information into consideration [130]. Intuitively, whether a grasp is successful or not is always related to how the robot (gripper) interacts with the object surface in 3D space; thus the lack of geometry analysis could entail side effects to grasp planning, especially when accurate and complete sensing is not available.

To tackle these unsolved issues, inspired by the recent work of PointNet [89] that directly operates on point clouds for 3D object classification and segmentation, a point cloud based grasp evaluation method for detecting reliable grasp configurations from the point cloud is proposed in this thesis. As illustrated in Figure 4.1, PointNetGPD provides an effective pipeline to generate and evaluate grasp configurations. Compared with previous grasp detection methods that depend on multi-view CNN [113] or 3D CNN [118], our approach does not require point cloud projection on multiple 2D images or rasterization into dense 3D volumes. As a result, it can sustain the geometric information of the original point cloud and infer grasp quality more efficiently.

Recent success in deep neural network based grasp detection methods [75, 66] emphasizes the importance of training on large-scale datasets. To further improve the performance of the proposed grasp detection method, we built a grasp dataset with a 350k real point clouds captured by depth cameras, parallel-jaw grasps and analytic grasp metrics over a subset of the YCB [11] object dataset. Different from other grasp datasets like Dex-Net [75], we provide fine-grained scores for each grasp instead of binary labels. Specifically, given a 6D grasp pose and a CAD model of an object, we perform force-closure [85] and a friction-less GWS [58] analysis on the grasp respectively to obtain such scores. Quantitative scores make the more flexible label assignment possible during training, which could also improves the performance of our grasp quality evaluation network.

To summarize, our key contributions of this chapter are:

- We propose to evaluate the grasp quality by performing geometry analysis directly from a 3D point cloud based on the network architecture of PointNet [89]. Compared with other CNN-based methods [75, 113, 130], our method can exploit the 3D geometry information in the depth image better without any hand-crafted features and sustain a relatively small amount of parameters for learning and inference efficiency. Also, we found our proposed method still works well even when the point cloud is very sparse, which implies its potential for planning grasps under imprecise and deficient sensing.

- We build a large-scale grasp dataset that contains 350k parallel-jaw grasps with the point cloud captured in real world. Meticulous grasp quality scores that combine the force-closure and GWS analysis are provided. Our experiments demonstrate that our grasping model can obtain significant performance gains from these meticulous scores and labels.

- To overcome the drawback of single-viewed point cloud might generate invalid grasp candidates, we tested to add a shape completion framework to enhance the quality of grasp candidates generation.

4.2 Problem Formulation

4.2.1 Definitions

Given a specified object o, things that are related to grasping will be the coefficient of friction between the object and gripper $\mu \in \mathbb{R}$, the object's geometry and mass properties M_o, and the 6D pose $W_o \in \mathbb{R}^6$. Let $\mathrm{s}_o = (W_o, M_o, \mu)$ represent the state of the object. We denote a grasp configuration in 3D space as $\mathbf{g} = (\mathbf{p}, \mathbf{r}) \in \mathbb{R}^6$, where $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ and $\mathbf{r} = (r_x, r_y, r_z) \in \mathbb{R}^3$ specify the position and orientation of the gripper respectively. We only consider parallel-jaw grippers in this chapter. Also, we assume a camera to capture the depth map, and the converted point cloud that contains N points is denoted as $\mathbf{P} \in \mathbb{R}^{3\times N}$. For simplicity, all spatial quantities are in camera coordinates. To evaluate the quality of a grasp, we denote a quality metric as $Q(\mathbf{s}, \mathbf{g}) \to \mathbb{R}$. Notice that Q works with an accurate object state instead of a point cloud, and our grasp quality is a continuous quantity instead of a binary label.

4.2.2 Objective

Given a gripper configuration $\mathbf{g}$ and sensor observation $\mathbf{P}$, our goal is to learn a quality metric $Q_\theta(\mathbf{P}, \mathbf{g}) \in \{c_0, c_1, \cdots\}$ to predict the grasp quality from a point cloud. θ defines the parameters of our proposed grasp quality evaluation network described in Section 4.4. $c_0, c_1, \cdots$ are labels that represent the quality of a grasp $\mathbf{g}$, and can be assigned to any ground truth quality metrics $Q(\mathbf{s}, \mathbf{g})$.

4.2.3 Challenges

There are two main challenges to solving the problem mentioned above. First, learning such a grasp quality metric may require a massive number of samples over a wide range of objects to achieve good performances and generalization. Second, the input point cloud $\mathbf{P}$ could be imprecise and deficient, which leads to additional difficulties in geometry analysis. Consequently, we propose to evaluate the grasp quality by direct point cloud analysis with PointNet [89], and train our grasp quality metric on a generated large-scale dataset of 350k real point cloud and grasps over objects from the YCB [11] object dataset to obtain robust grasp classification results.

4.3 Grasp Dataset Generation with Meticulous Scores

The generation of our grasp dataset involves two steps: sampling and scoring. Grasp candidates are firstly sampled over provided object meshes; then these candidates will be labeled by robust grasp quality metrics including force-closure and GWS, details are listed as follows:

4.3.1 Sampling

Although the YCB [11] object dataset provides registered point clouds for most of the objects, we still sample over the precise meshes scanned using Google Scanner instead to prevent the sampler from generating unfeasible grasps (such as grasps that collide with the object). For each grasp, we randomly sample two surface points $\mathbf{p}_1$, $\mathbf{p}_2$ as contact points and an approach angle between $[0, \pi/2)$ as the approach angle, then a grasp $\mathbf{g}((\mathbf{p}_1 + \mathbf{p}_2)/2, \mathbf{r})$ will be constructed. To further eliminate unfeasible grasps, we conduct a sanity check by simulating the approach and close-finger action with a gripper model to see whether it will collide with the object. Finally, the remaining grasps are then transformed from mesh into point cloud coordinates. The transform matrices are obtained by doing Iterative Closest Point (ICP) using PCL [96] between the mesh and corresponding registered point clouds.

4.3.2 Scoring

Given a sampled grasp g and object state s, we adopt two different robust grasp quality metrics to label the grasp. One of them is a force-closure metric Q_{fc}; it requires the coefficient of friction μ and only provides a binary outcome that indicates whether the grasp is force-closure or not. Here we modify it to enable quantitative scoring: Starting from 0.4, we gradually increase μ until the grasp is antipodal, then the value $1/\mu$ will be recorded as a score for the current grasp. Such modification is intuitive since an antipodal grasp that requires lower friction could be arguably better. As is shown in Figure 4.2, the grasp with lower μ could be more robust and feasible. We also observe that such difference will be more notable when the object has a more complex physical shape.

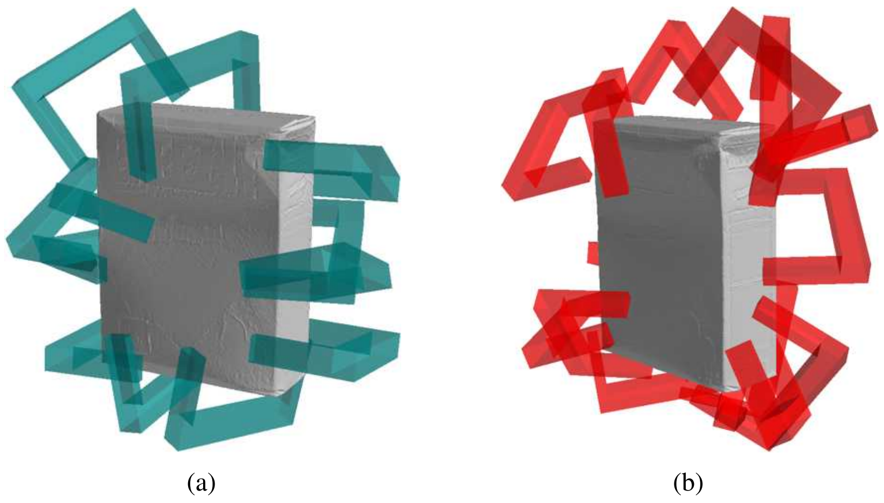


Figure 4.2: Example grasps in our dataset label with Q_{fc}. (a) green grasps are labeled with $\mu = 0.4$. (b) red grasps are labeled with $\mu = 2.0$. We find that on this relatively simple box-like object, there is a significant difference in robustness between the green and red grasps.

The other grasp metric Q_{gws} is based on GWS analysis [58]. Compared to Q_{fc}, GWS analysis proposes to use the radius of GWS as a quantitative score of grasp quality. GWS itself can either be calculated in $\mathbb{R}^3$ or $\mathbb{R}^6$ space. In practice, here we only apply a simplified Q_{gws} with $\mathbb{R}^3$ friction-less grasp wrench space.

We adopt a weighted sum to combine these two kinds of metrics, and produce a final quality score:

$$Q(\mathbf{s}, \mathbf{g}) = \alpha Q_{fc}(\mathbf{s}, \mathbf{g}) + \beta Q_{gws}(\mathbf{s}, \mathbf{g}). \tag{4.1}$$

We observe that Q_{gws} could be much larger than Q_{fc} for most of the grasps and objects, thus we choose $(\alpha, \beta) = (1.0, 0.01)$ in our experiments.

4.3.3 Training Dataset

We use the dataset generated in section 4.3 to train our grasp quality evaluation model. Since there are quantitative quality (Equation 4.1) values instead of binary labels in our dataset, it will be flexible to assign classifying labels and even enable the multi-class grasp quality classification. The threshold for each label will be discussed in Section 4.5. There are 350k point cloud and grasps over 47 YCB objects. To keep a balance of grasps with different qualities, we sample an equal number of grasps with Q_{fc} value from $\{1/0.4, 1/0.45, 1/0.5, 1/0.8, 1/1.2, 1/1.6, 1/2.0\}$. For the point cloud, as suggested in [113], we use the real point cloud provided by YCB object dataset instead of the simulated point cloud obtained with a CAD model for a better generalization to real-world grasping tasks.

4.4 Learning a Grasp Quality Metric from Point Cloud

4.4.1 Brief Introduction of PointNet

To deal with the point cloud as input, a neural network that can process point cloud information efficiently is needed. Most researchers would like to transform input point cloud into voxel grids and use network like VoxelNet [133] for inference. Or people can project point cloud into several images like [113] that treat the inputs as images and processing them using CNN. However, using a network that can directly take point cloud as input and can respect the permutation invariance of the point cloud would be a better choice. PointNet [89] is the network architecture that meets our needs.

The architecture of PointNet is shown in Figure 4.3. The input $n \times 3$ points were first going into a mini-network (T-Net) for calculate an affine transformation matrix, and use matrix multiply to transform the input point cloud to a unified coordinate. Then with two Multi-Layer Perception (MLP) layers and a similar mini-network for the transform of the features that output from the MLP. This feature transformation matrix is used to align features from different input point clouds. This local feature output F_{local} is then sent to three MLP layers and a max pooling layer to get a global feature F_{global}. For classification task, F_{global} can be directly used as input of three MLP layers. The output is k classes depends on the label number. For segmentation task, F_{local} and F_{global} are concatenated together. In detail, F_{global} is inserted into each element of F_{local}. Then throw several MLP layers, the output is $n \times m$, where m is the number of parts of the input scene.

4.4.2 Network Architecture and Grasp Representation

The architecture of our grasp quality evaluation network is illustrated in Figure 4.4. Our PointNet based network will take as input the grasp represented by the point cloud within the closing area of the gripper. For learning and inference efficiency, we do not take the whole point cloud as input like [75, 66]. The point cloud will firstly be transformed into the unified local gripper coordinate introduced in Figure 4.5, this is mainly

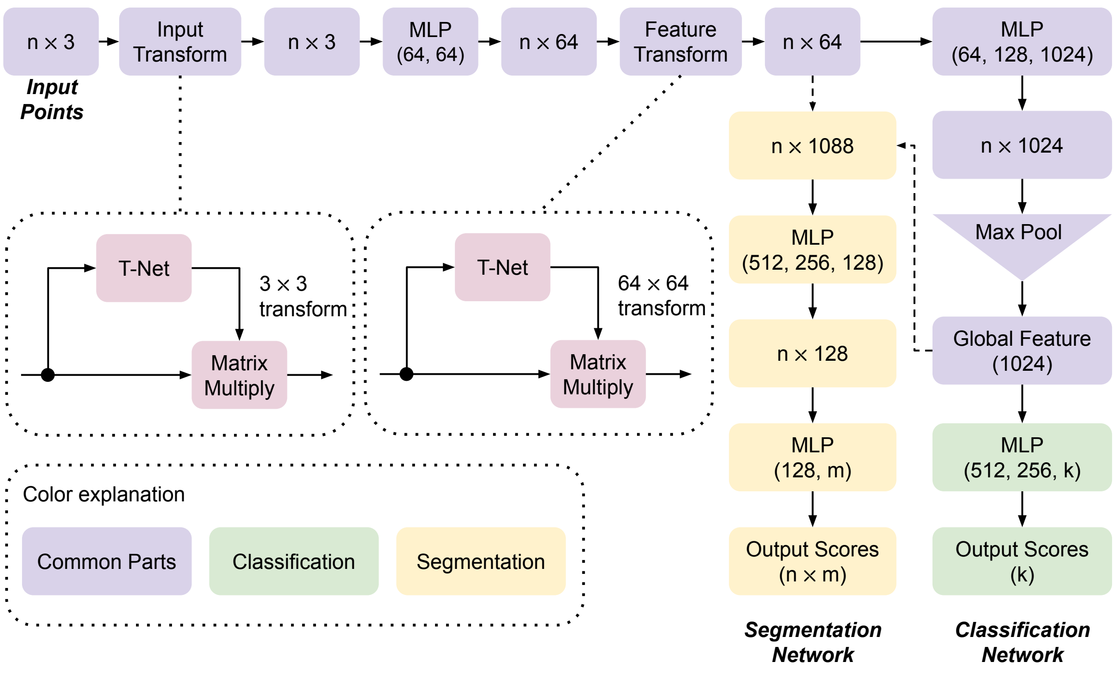


Figure 4.3: Architecture of PointNet. The input of the network is n points. With several components in common (violet blocks), PointNet can handle different perception tasks like classification (green blocks) and segmentation (yellow blocks). The segmentation network is the extension of the classification network, which get features both from local side ($n \times 64$) and the global side (1×1024). Adapted from [89], ©2017, IEEE.

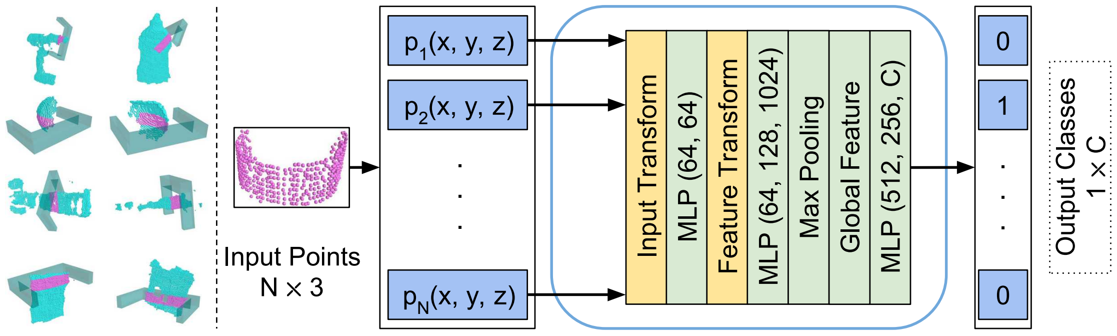


Figure 4.4: Architecture of our grasp quality evaluation network based on PointNet. Given a grasp and point cloud, the grasp is represented by the points within the closing area of the gripper. As is shown in Figure 4.5, all the points will be transformed into local gripper coordinates before being fed into the network. After several spatial transformations and feature extractions, the final global feature will be applied to classify the quality level of the input grasp.

to eliminate the ambiguity caused by the different experiment (especially camera) settings. Specifically, we treat the approaching, parallel and orthogonal directions of the gripper as the XYZ axes respectively, while the origin will be located at the bottom center of the gripper. Then these N points will be passed through the network to estimate the level of quality. Compared to other CNN-based grasp quality evaluation networks, our model is lightweight and only has approximately 1.6 million parameters.

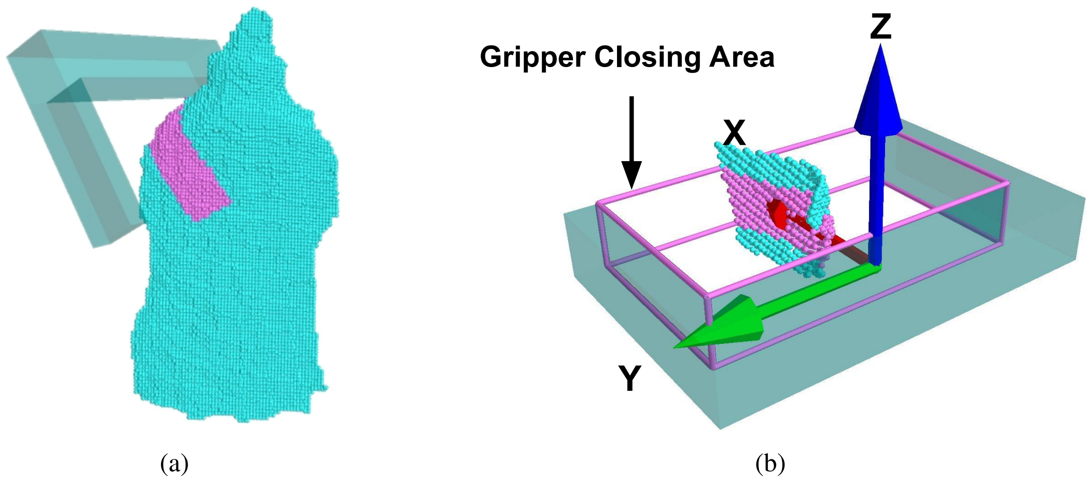


Figure 4.5: Grasp representation in the local gripper coordinate. A grasp is represented by the point cloud within the gripper closing area. (a) a typical grasp configuration. (b) axes of local coordinates and the transformed point cloud within the gripper closing area (magenta) that serves as the grasp representation.

4.4.3 Training and Inference Details

We use a C-class cross-entropy loss as the objective of our classifier. C equal to 2 and 3 in our case. The whole network is optimized with Adam [56] optimizer, and all the parameters are initialized with values sampled from a zero-mean Gaussian distribution. We augment our data by adding a random offset to the point cloud, but still keep all the points within the gripper closing area.

4.5 Network Evaluation

4.5.1 Network Evaluation Details

In network evaluation, we mainly want to compare the performances on grasp quality classification between our proposed PointNetGPD and current state-of-the-art methods. We choose GPD [113] as the baseline. In the dataset, since we cannot acquire the camera location for computing the unobserved area used in the 15 channel version of GPD, we only compare the 3 and 12 channel versions, and we compare with 15 channels version of GPD in robotic experiments. Also, to examine the stability on sparse point cloud, we provide either point cloud from 1-viewed or full point cloud input for each grasp. The point cloud of 1-viewed is taken from the camera in front of the object. For the full point cloud, we register the point cloud from all the available viewpoints. After the point cloud is ready, we discard the sample that has less than 50 points in the gripper closing area, then for the rest, we upsample/downsample their point cloud into 1000 points.

Finally, we run a 3-classes classification experiment mainly for verifying the validity of the scores we provided in our grasp dataset. For 2-classes classification, we regard a grasp with score above $1/0.6$ as positive, while for 3-classes classification, the score thresholds for 3-classes classification will be $1/0.5$ and $1/1.2$. See Equation 4.1 for the detailed calculation.

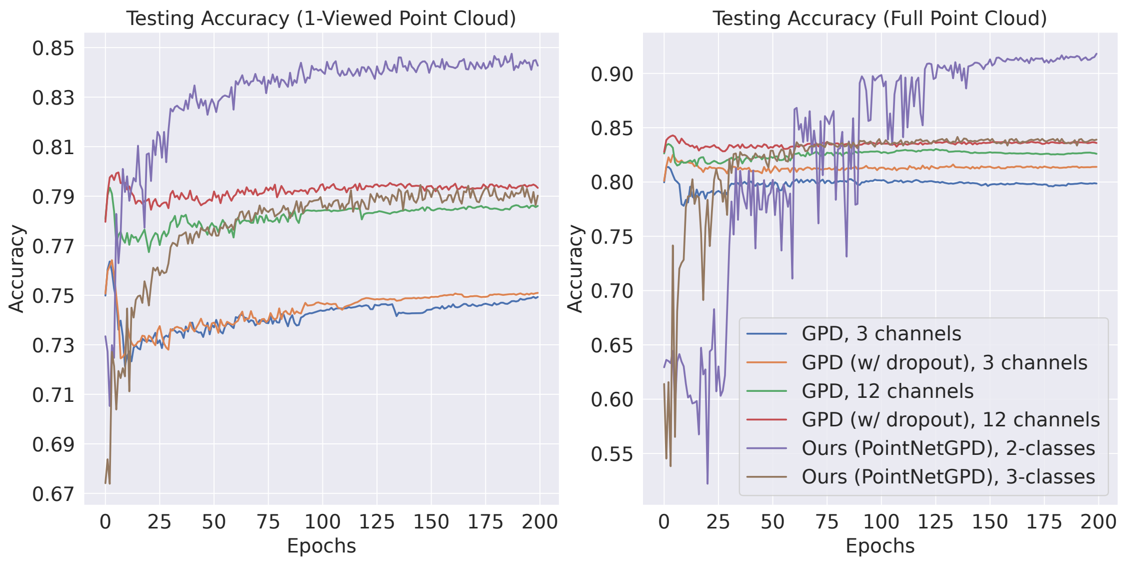


Figure 4.6: Classification accuracy with different models and configurations on single views and on full point cloud. We can find that all the models obtain better performance with full point cloud input than with a single view, while the proposed grasp evaluation model outperforms the baselines on both input types. More quantitative results can be found in Table 4.1.

Table 4.1: Accuracy of different models and configurations

	GPD (3 channels)		GPD (12 channels)	
	without dropout	with dropout	without dropout	with dropout
#Parameters	3.63M		3.64M	
1-Viewed Point Cloud	76.36%	76.42%	79.34%	79.96%
Full Point Cloud	81.38%	82.50%	83.50%	84.29%
	Ours (2-classes)		**Ours (3-classes)**	
	All classes	Best classes	All classes	Best Classes
#Parameters	**1.60M**			
1-Viewed Point Cloud	**84.75%**	82.26%	79.45%	**90.37%**
Full Point Cloud	**91.81%**	**92.18%**	84.15%	89.76%

4.5.2 Results Analysis

The testing accuracy of all the considered models during training is demonstrated in Figure 4.6. We list the best result among the 200 epochs in Table 4.1. Here we highlight some important facts we found in these results. First, our proposed PointNetGPD performs significantly better on grasp quality classification than all the GPD baselines. Even on the most difficult 1-viewed point cloud, PointNetGPD still has an averaged

4.79% improvement over the best GPD baseline. Furthermore, from Figure 4.6 we can see that GPD can easily get overfitting on the training set. However, although we make it easier by utilizing Dropout [107] on the GPD network, there is still a performance gap between GPD and our method. Such results are partly due to the number of parameters. Compared to GPD, the network in our proposed method has fewer parameters and performs better, which means that our network is more effective regarding geometry analysis, especially from the sparse point cloud.

For the 3-classes experiment, we found that the accuracy of the class of the best quality is even better than the best class in the 2-classes experiment. This may imply that a grasp with a higher score will be easier to identify (Equation 4.1). In our robotic experiments, we will make further validations by comparing the results of 2-classes and 3-classes grasping models.

4.6 Robot Experiments

For experiments on the robotic hardware, we conduct several robotic grasping tasks to see whether our model can generalize well to real-world settings. We validate the reliability and efficiency of our proposed PointNetGPD in two robotic experimental conditions: objects were presented to the robot in isolation as well as in a clutter. These experiments were carried out on a UR5 robotic arm with an attached Robotiq 3-finger adaptive robot gripper. As shown in Figure 4.7(a), the gripper works under pinch mode, in which only two contact surfaces are allowed to move toward and away from each other along a 1-D manifold. Especially, since we only use one Kinect2 depth sensor, all the point clouds provided in robotic experiments are 1-viewed, which makes it even more challenging.

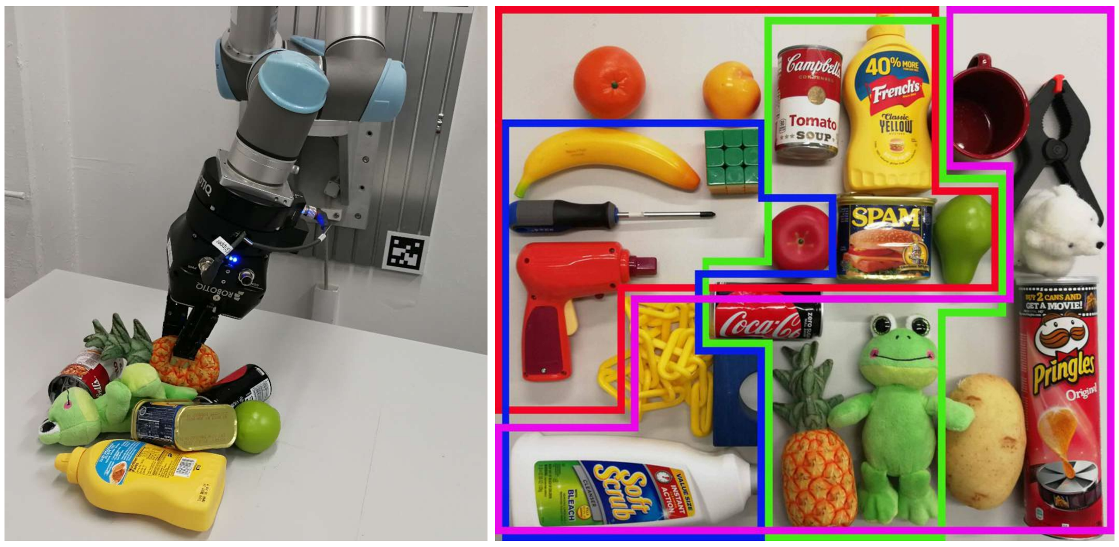
We selected 22 objects from the YCB object dataset. In these objects, 11 of them have already been presented in our grasp dataset, while the rest are novel. We also selected 16 from 22 objects to construct two object sets that are used for clutter removal. Details can be found in Figure 4.7(b).

The whole system is implemented using the ROS framework, particularly, a fast hybrid evolutionary inverse kinematics solver bio-ik [95] is used for solving inverse kinematics within the MoveIt [21] framework.

For both conditions, we compare a 2-classes and a 3-classes PointNetGPD with a 15-channel GPD baseline. In addition, to validate the significance of the quality scores provided in our dataset, we also compare the grasp performance between the best and the second class in 3-classes PointNetGPD.

4.6.1 Data Preprocessing

Single acquired point cloud view from 3D cameras generally includes a non-negligible level of noise and data outside our defined region of interest. Here we denoise the input clouds by voxelizing, remove outliers and clip the cloud around a defined region of interest. Notably, this last step removes the support surface of graspable objects from the input. We don't segment objects from these point clouds because we generate grasp



(a) (b)

Figure 4.7: Settings of our robotic grasping experiments. (a) Grasping experiment setup with UR5 robotic arm and Robotiq 3-finger adaptive robot gripper. (b) Objects used in our experiments. Red polygon shows the objects presented in the training dataset, magenta polygon contains the objects that are not in the dataset. The green (clutter 1) and blue (clutter 2) polygons present the two object sets used in clutter experiments, respectively.

candidates using geometry information rather than the pose and the shape of an object. Then GPG with our improvement as mentioned in Chapter 3 is used to generate grasp candidates and PointNetGPD is used to evaluate the proposed grasp candidates.

4.6.2 Objects Presented in Isolation

In this experimental condition, all the objects presented in Figure 4.7(b) are tested. We test each object for ten rounds with random initial orientations. If the gripper failed to grasp an object or no collision-free grasp pose was generated within a long time (in our practice, we use 5 minutes), we mark this attempt as failed. We only consider the success rate for performance evaluation in this experiment.

Table 4.2 demonstrates the grasping results for a single object using three different models. Note that Table 4.2 does not contain the objects whose success rates are 100% for all the three models, such as chips can, Rubik's Cube, plastic apple and so on, or 0% such as the medium clamp. The 0% success rate of this object is probably caused by the poor quality and the low height of the acquired point cloud, and the irregular shape. As Table 4.2 illustrated, the two types of PointNetGPD methods manifest a higher average success rate, which suggests that the proposed model can better understand the spatial geometry of the point cloud in the graspable region.

Table 4.2: Results of single object grasping experiments

Method	Avg.	Bleach cleanser	Mug	Meat can	Tomato soup can	Banana	Toy power drill	Chain	Mustard bottle	Wood block	Screw driver
GPD	49%	**100%**	30%	60%	90%	20%	**80%**	0%	90%	**90%**	20%
Ours 2-classes	81%	**100%**	50%	**80%**	**100%**	**90%**	70%	**60%**	**100%**	**90%**	70%
Ours 3-classes	**82%**	90%	**70%**	70%	**100%**	**90%**	**80%**	**60%**	90%	**90%**	**80%**

Table 4.3: Results of clutter removal experiments

	GPD		Ours 2-classes	
	Success rate	Completion rate	Success rate	Completion rate
Set 1	84.83%	95%	86.54%	94.08%
Set 2	61.13%	81.50%	61.07%	84.38%
	Ours 3-classes (best class)		**Ours 3-classes (second class)**	
	Success rate	Completion rate	Success rate	Completion rate
Set 1	**89.33%**	**100%**	52.10%	**100%**
Set 2	**66.20%**	**95%**	43.75%	37.5%

4.6.3 Objects Presented in Dense Clutter

In the dense clutter condition, we select 16 objects from those who have grasping success rate above 0 for all the compared model to construct two object sets (Set 1 and Set 2). The green and blue polygons in Figure 4.7(b) represent these two sets, respectively. Furthermore, Set 1 has six objects with 100% success rate in isolated condition for all the three models, while Set 2 only have two objects with 100% success rate. We run experiments with each object set for five rounds.

Besides the models we compared in the isolation condition, here we also test the grasp that is predicted to be the second class through our 3-classes PointNetGPD. This is mainly to verify the validity of the multi-class classification. We use *success rate* and *completion rate* as the criterion for performance evaluation. The *success rate* is the percentage of successful grasps, while the *completion rate* is the percentage of objects that are removed from the clutter.

From the results presented in Table 4.3, we found that all the models overall perform better in Set 1 than Set 2 because the objects in Set 1 could better fit the geometry shape of the gripper, and some of them have a higher roughness. Meanwhile, grasps from the best class of 3-classes PointNetGPD show the best grasping outcomes, especially on *completion rate*. This shows a significant averaged improvement of 13.5% over GPD. Moreover, the fact that grasps from the best class of 3-classes PointNetGPD are hugely superior to the second class confirms the effectiveness of 3-classes classification, which implies the capability and implication of the meticulous scores in our dataset.

Occlusion and incomplete point could cause failures in this experiment since we only have one fixed view of the point cloud. Additionally, multiple adjacent objects could be treated as one single object because of missing the object segmentation procedure. Therefore, the final grasps are employed across multiple objects, resulting in failures.

4.6.4 Object Shape Completion

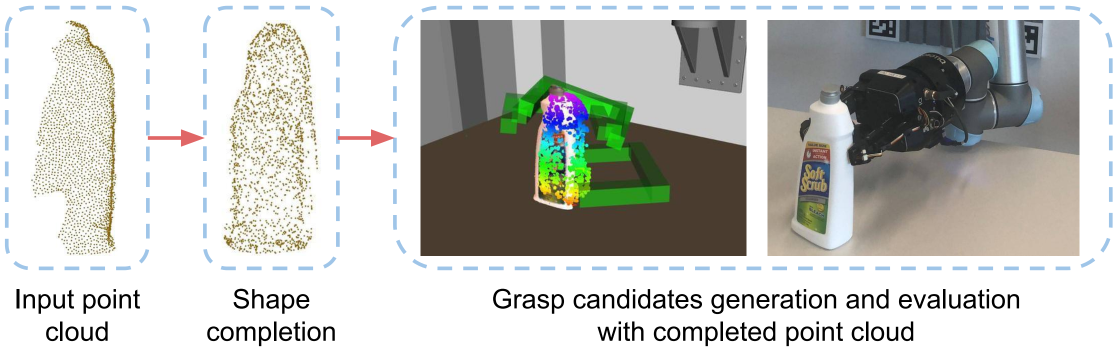

Input point cloud
Shape completion
Grasp candidates generation and evaluation with completed point cloud

Figure 4.8: Overview of the shape completion based grasp pipeline. The left two blocks are shape completion module. In this module, a partial point cloud in first block is sent to a shape completion network and the output is in second block. The output point clouds serve as the input of GPG, then PointNetGPD is used to evaluate the grasp candidates.

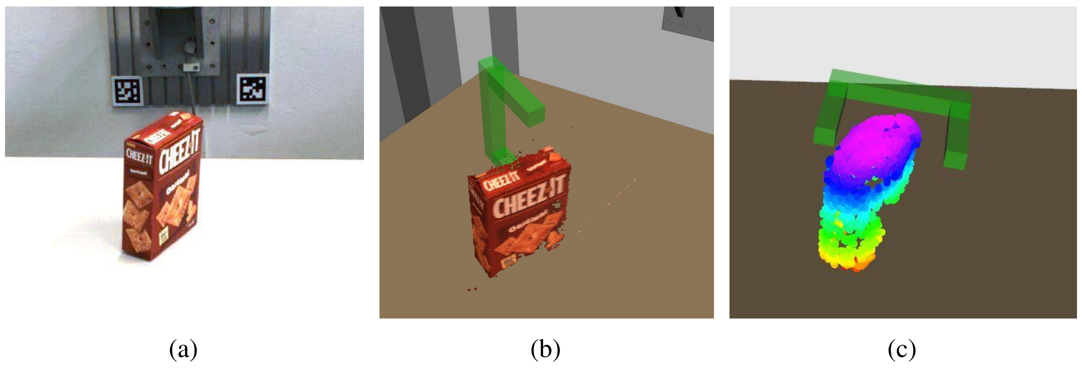


Figure 4.9: Comparison of grasp candidates generated using GPG [114] with different input point cloud. (a) RGB image to show the example environment, (b) grasp generated with partial point cloud, (c) grasp generated with complete point cloud.

PointNetGPD is trained on a grasp dataset generated using reconstructed YCB object mesh and evaluates the input grasp quality. The grasp candidates in the grasp dataset are all collision-free with respect to the target object. As a result, the grasp evaluation network assumes all the input grasp candidates are not colliding with the object. If the object has occlusion due to the camera viewpoint, current geometric-based grasp proposal algorithm will generate grasp candidates that collide with the object. Thus, using a complete point cloud could ensure that the grasp candidate generation algorithm generates grasp sets that do not collide with the graspable objects [14]. Figure 4.9 shows the comparison of grasp generation result using GPG [114] with and without point cloud completion, Figure 4.9(b) shows a candidate generated using partial point cloud and Figure 4.9(c) shows a grasp candidate generated using complete point cloud. We can see that the grasp in Figure 4.9(b) has a collision with the real object while Figure 4.9(c) avoids generating such grasp.

To evaluate the performance improvement using complete point cloud for robotic grasping, we choose six YCB objects to test the grasping success rate. The robot arm and gripper used in this experiment is same as above. However, the vision sensor is changed to Mech-Eye, an industrial 3D camera from Mech-Mind[1] to acquire a high-quality partial point cloud. The image of the camera and its point cloud sample are shown in Figure 4.10. The package that used to integrate this camera into ROS is developed by us based on the official CPP API and is open sourced[2]. The selected six objects are listed in Table 4.4. We select these objects because they are typical objects that may fail to generate good grasp candidates without shape completion. For other objects such as banana or marker, they are quite simple and small, which causes that improvement of shape completion on the grasping result is minor.

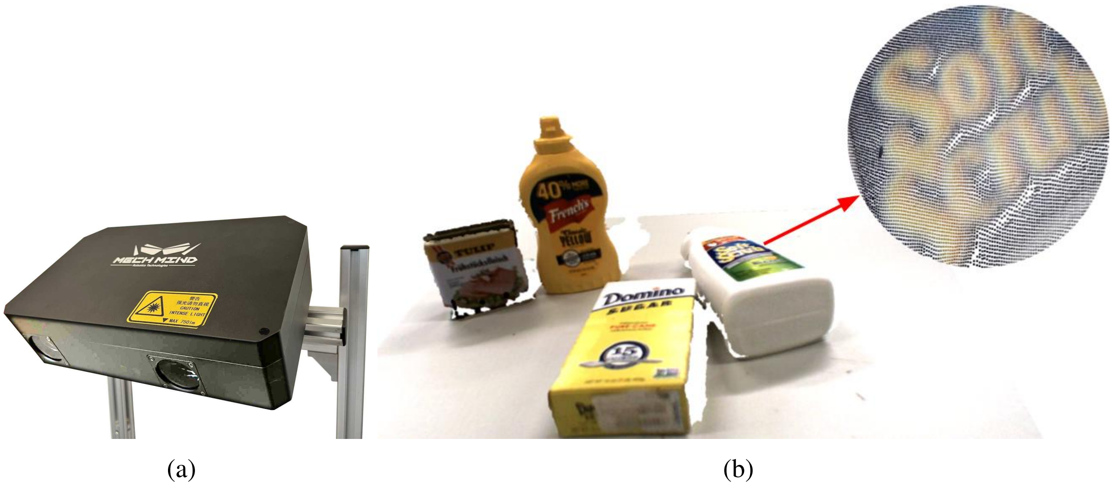


Figure 4.10: (a) Image of Mech-Eye camera from Mech-Mind mounted on a camera holder. (b) Example point cloud take from Mech-Eye camera.

Table 4.4: Real robot experiment result

Method	Cracker box	Mug	Meat can	Pitcher base	Bleach cleanser	Power drill	Avg.
Without shape completion	70%	70%	80%	80%	90%	40%	71.67%
With shape completion	**80%**	**100%**	**100%**	80%	90%	**50%**	**83.33%**

For the selected six objects, we perform grasp evaluation on two different methods: PointNetGPD grasp with/without shape completion. We run the robot experiment by

[1]`https://en.mech-mind.net`
[2]`https://github.com/MechMindRobotics/mecheye_ros_interface`

randomly putting the object on the table and grasping for ten times, then calculating the success rate. The experiment result is shown in Table 4.4. We can see that all six objects outperform or equal with the one do not use shape completion as input point cloud. The low success rate of power drill for both methods is because when the robot tries to grasp the head of the power drill, the contact area is too slippery. A typical failure we observed is from the limited camera viewpoint. In some viewpoint, GPG generates grasp candidates that sink into the object. An example of this situation is shown in Figure 4.9. This is a strong evidence that shape completion model can improve the grasp success rate in some particular situations.

4.7 Discussion and Summary

In this chapter, we presented PointNetGPD, a novel approach for detecting grasp configurations from point clouds. As the core module in our grasp pipeline, we proposed to address the challenging grasp quality evaluation over imprecise and deficient point cloud with PointNet. To further improve the performances, we generate a large-scale grasp dataset with 350k real point cloud and grasps with the YCB object dataset for training. Our experiments show that our model outperforms the state-of-the-art grasp detection method. The grasping dataset and software for dataset generation and network are public available[3].

As a final remark, our goal is to integrate the grasp candidate generation step into the network for performing grasp planning in an end-to-end fashion. Additionally, our grasp generation method did not consider objects that are placed tightly together, thus our network may give a false grasp output that grasp more than one objects. We plan to do clutter segmentation simultaneously [128, 120], which can prevent the model from planning unfeasible grasps that cross more than one object.

[3]`https://lianghongzhuo.github.io/PointNetGPD`

Chapter 5

Multifingered Grasping Based on Multimodal Reinforcement Learning

5.1 Introduction

Even though the two-fingered grasping problem has been widely studied and reaches a satisfying success rate as demonstrated in the last chapter, multifingered grasping is still far from solved. The fact that the robotic community is beginning to expect robots to approach the manipulation capabilities of humans makes it important to solve this problem. Even with carefully planned trajectories or a dedicated mechanism design [43, 44, 74, 54], two-fingered grippers can only be used to execute some simple object interactions and manipulate some specific object categories. Therefore, to endow robots with the same dexterity as human hands, which can effectively grasp different objects and utilize various tools like a spraying bottle or in-hand rotating a cube [2], anthropomorphic hands have become a promising solution and have gained much attention over the past years of research.

The human hand is a very complex, articulated biomechanical system, and the problem of replicating its structure and capability is very challenging in terms of mechanical design and motion planning and control. When grasping an unknown object in daily life, humans usually first use visual perception to estimate the object's physics properties, *i.e.*, size, weight, center of mass, surface smoothness, and stiffness, based on prior experience. A rough grasp pose can be estimated from visual information. During approaching and contacting with the object, humans begin to use their comprehensive tactile and force feedback to explore the object further and finally choose a stable posture to grasp it. During this process, humans will estimate whether the grasp is stable by the tactile and visual feedback before lifting the object to avoid slipping out of the hand.

Inspired by how humans grasp objects, we develop a robust robotic grasping strategy by merging multiple sensing modalities with anthropomorphic dexterous hands. The fusion of different data from tactile fingertips, torque sensors, and robot proprioception (joint positions) promises a robust and intelligent method to teach the robot multifingered grasping.

Therefore, we propose a hand-arm system for multifingered grasping. We start with

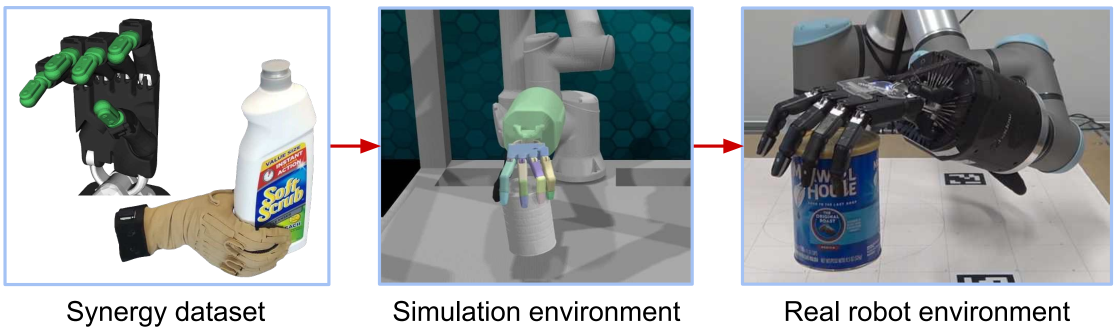


Figure 5.1: An illustration of the multifingered hand-arm grasping system. To simplify the hand's control and make the hand move like a human, in the left block, we collect a Shadow hand pose dataset using Cyberglove to teleoperate a virtual hand. We use this dataset to perform a principal component analysis to reduce the number of inputs needed to control the robot hand. The middle block presents our simulated training system. In the simulation, we use multimodal information as observation to train a multifingered grasping agent. The right block illustrates that the grasping agent purely trained in simulation has proved to work well in the real world by several grasping experiments.

a point cloud as input and run the grasp evaluation network PointNetGPD proposed in Chapter 4 to generate a candidate two-fingered grasp, then map this as the pre-grasp pose for the dexterous hand. To control the dexterous hand to execute the grasping action, instead of using a fixed grasping trajectory, we first use hand synergies [99] to make a dimensional reduction. Then we use the reduced dimension information as target action space to train a multimodal RL based agent. With this agent, the dexterous robotic hand can close its fingers and grasp the object successfully.

Our contribution of this chapter can be summed up as follows:

- We collect a dataset of common human hand motions and map these motions to the Shadow robot hand successfully. Because the corresponding motion data of the Shadow hand joints are too complicated and take up a lot of computing memory, we calculate postural hand synergies using PCA to reduce the dimension of controlling the hand;
- We build a dexterous manipulation simulation environment based on CoppeliaSim [93] and PyRep [53] which has a UR10e[1] robot arm mounted with a Shadow robot hand[2];
- We introduce a multifingered grasping agent that fuses multimodal sensor data (fingertip tactile sensing, joint torques, and hand proprioception) based on the RL algorithm. The agent is easy to transfer from simulation to the real world platform with the help of binary tactile information and level-based torque information. Our robot experiments prove that our agent trained in simulation works well on the real robot system and outperforms the baseline methods;

[1]`https://www.universal-robots.com/products/ur10-robot`
[2]`https://www.shadowrobot.com/dexterous-hand-series`

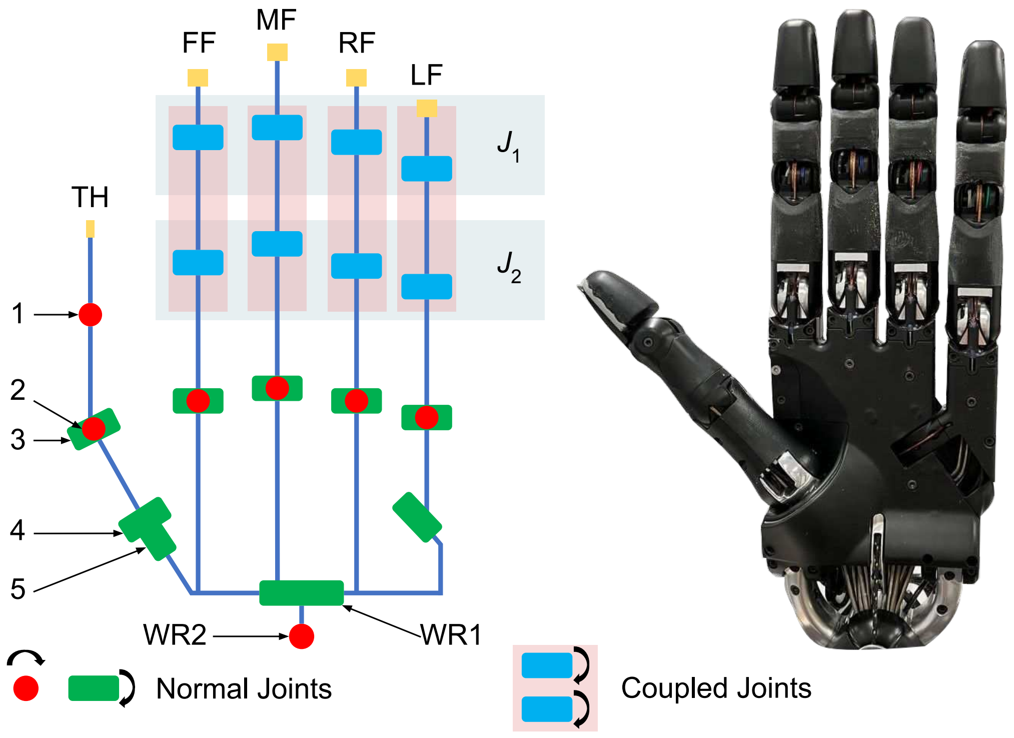


Figure 5.2: Joint mechanics of the Shadow hand. $\theta_w \in \{\mathrm{WR1}, \mathrm{WR2}\}$ refers to wrist joints 1 and 2. And $\theta_f \in \{\mathrm{LF}, \mathrm{RF}, \mathrm{MF}, \mathrm{FF}, \mathrm{TH}\}$ refers to little finger, ring finger, middle finger, first finger, and thumb (22 joints). Joints 1 and 2, marked blue in each finger, are coupled (these two joints are controlled by one motor).

- Through comparative experiments in simulation, we find that the fusion of multimodal sensing increases the performance of the agent, and the policy using the Recurrent Neural Network (RNN) structure (Gated Recurrent Unit (GRU) in our work) is better than a simple MLP structure. We also find that the most suitable dimension reduction value for PCA is 5. Real robot experiments are performed with different models. By comparing the model with different modalities and different baselines, we verified the effectiveness of our proposed algorithm.

The rest of this chapter is organized as follows. The grasp synergies dataset is introduced and collected in Section 5.2. Then the multimodal grasping policy is proposed to solve the multifingered grasping problem in Section 5.4. In Section 5.6, both the simulation and real robot experiment are described to compare the success rate between RL agent and two baselines. Finally, the conclusion and future work are discussed in Section 5.7.

5.2 Grasp Synergies Dataset

Similar to the human hand, the Shadow hand has five fingers with 24 joints, including 2 active joints on the wrist, 18 active joints on the fingers and 4 coupled joints, as shown in Figure 5.2. However, it is pretty challenging to train a RL agent in such a high-

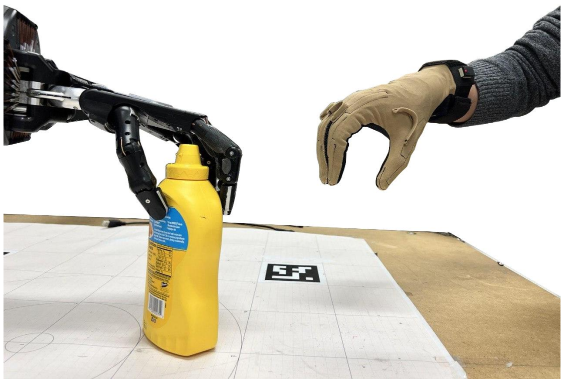


Figure 5.3: An illustration of human teleoperating Shadow hand using a Cyberglove for robotic grasp pose dataset collection.

dimensional action space. High-dimensional action space equals huge exploration space, which will cause a low learning efficiency and without constraints is likely to generate weird hand poses. To ensure that the generated hand poses are more human-like and the exploration space can also be significantly reduced, we simplify this problem by creating a motion subspace.

In the research field of human hand grasping, it is popular to apply the definition of grasp synergies [99] to describe and simplify human grasps by PCA. Therefore, a Shadow hand pose dataset is necessary to calculate the eigenvectors and eigenvalues representing the most correlated directions in the joint space.

To this end, we collect a Shadow hand pose dataset, using the 33 grasp types of humans described in [34], such as large-diameter grasp and tripod grasp. In this chapter, we made a big dataset of anthropomorphic robotic hand grasping datasets that consists of three parts of data based on that taxonomy. The first portion of the hand pose dataset is from our previous work [5]. Three human subjects teleoperate an air-muscle version of the Shadow hand with a Cyberglove[3]. Eight precision grasp types are used to grasp twelve primitive objects and 442 joint samples are recorded. Figure 5.3 is an illustration of using Cyberglove to teleoperate a Shadow hand for robotic grasp pose dataset collection. To improve the collection efficiency and the object diversity of the dataset, we further collect 3000 samples by controlling the virtual Shadow hand using the Cyberglove as shown in Figure 5.1(left). Furthermore, we extend the hand pose dataset based on the YCB-Affordance dataset [23]. The YCB-Affordance dataset contains manually annotated human grasps covering all 33 grasp taxonomies on the 58 objects of the YCB benchmark set. We use the bio-ik [95] solver to calculate a corresponding Shadow hand mapping via the labeled human hand keypoints. After removing some unreachable grasps for the Shadow hand, we finally got more than 200 robot hand postures.

We consequently perform a principal component analysis to reduce the joint space dimensions used to control the multifingered hand based on the collected dataset. The n eigenvectors corresponding with the largest n eigenvalues are selected to form a transformation matrix. The optimal number of synergies n is determined by our comparative

[3]`http://www.cyberglovesystems.com`

model evaluation in Section 5.5. Figure 5.4 illustrates the hand motion corresponding to the first three components, which are called the first, second, and third grasp synergies (C1, C2, C3).

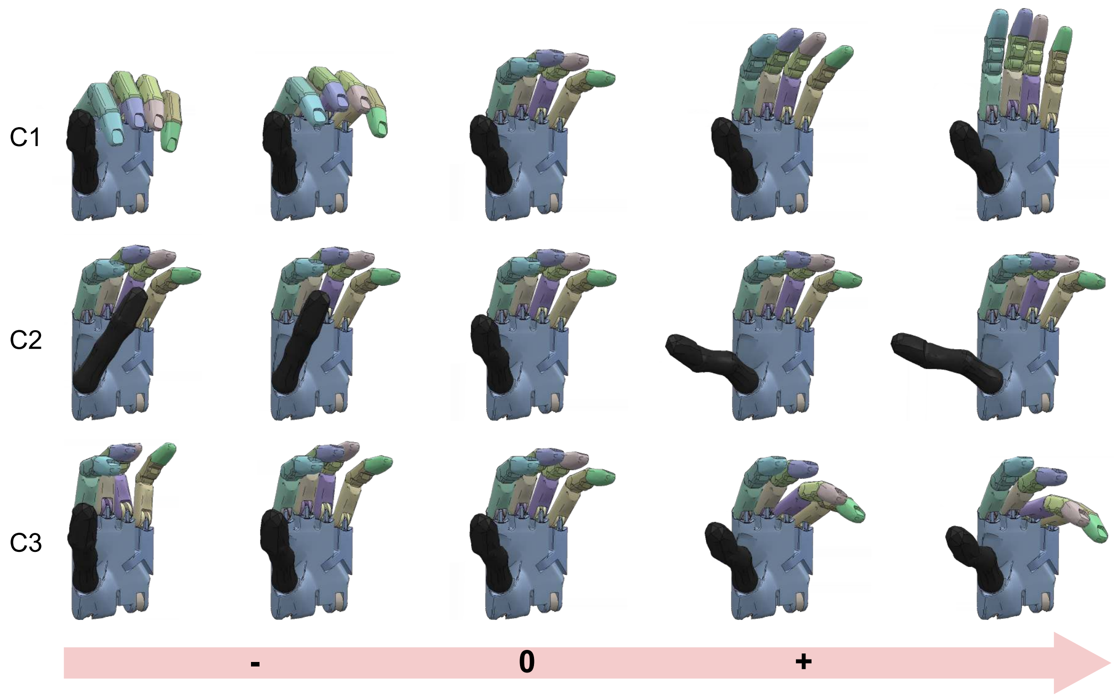


Figure 5.4: Shadow hand synergies. C1, C2, and C3 rows show example grasp postures of the Shadow hand controlled by the first three principal components of our dataset. For each component, C1 is responsible for open/close all fingers, C2 rotates the thumb for precision grasp, C3 is for the in-hand rotation.

5.3 Simulator Selection

Using a physics simulator for robotic grasping evaluation and grasp planning is a very useful approach. Collins *et al.* [22] gave a detailed review on the pros and cons in different robotic application areas. Each simulator has its own strengths, like some simulators are for medical robotics, they focus more on the teleoperation simulation performance. Some simulators focus more on soft body simulation or liquid simulation. In this thesis, we focus more on different performances on robotic manipulation, especially on robotic grasping. All the experiments are purely calculated in a computer and do not need to worry about damaging the real robots [17]. A simulator can simulate the rigid body contacts of different objects, links, bodies while controlling a moveable part inside the simulator. In the simulator, all the physics definitions apply Newton's three laws. So we can use simulation to evaluate the grasp quality. Note that RViz[4] is not a simulator. RViz is only a visualizer that can visualize all the robot models in the given pose.

[4]`https://github.com/ros-visualization/rviz`

There are a lot of simulators available in the market. But they can all be divided into two parts, the back end and the front end. The back end, also called the simulation engine, is usually responsible for calculating the forces caused by the contacts and the position, speed and acceleration of all the objects in the simulation environment. Often, the user can also add some sensors into the environment, like vision sensor, force/torque sensor, tactile sensor that simulate the corresponding sensor in the real world. The front end is a GUI that shows all the objects in the simulation environment, the light, and the rendering of the vision sensors.

Currently, popular physics simulators/engines are Gazebo [59][5], Bullet[6], DART[7], Darake[8], Webots[9], MuJoCo [116][10], CoppeliaSim [93][11], Chrono [111][12] and other simulation software.

Recently, as deep learning comes more and more popular, there is a trend that make the training in the simulator differentiable like Nimble[13], Brax[14] and Isaac Gym [77][15]. A comparison of different simulation software is shown in Table 5.1.

Table 5.1: Free physics simulators comparison

	Open Source	Engine	GPU support	Differentiable
Gazebo	✓	✗	✗	✗
Bullet	✓	✓	✗	✗
DART	✓	✓	✗	✗
Darake	✓	✓	✗	✗
Webots	✓	✗	✗	✗
MuJoCo	✓	✓	✗	✗
CoppeliaSim	✓*	✗	✗	✗
Chrono	✓	✓	✗	✗
Nimble	✓	✓	✓	✓
Brax	✓	✓	✓	✓
Isaac Gym	✗	✓	✓	✓

*The CoppeliaSim lib core is open-sourced under GNU GPL.

For doing robotic grasping simulation, we choose to use CoppeliaSim. This is because this simulator has quite a large user community and is free to educational use.

[5]`https://gazebosim.org`
[6]`https://pybullet.org`
[7]`https://dartsim.github.io`
[8]`https://drake.mit.edu`
[9]`https://cyberbotics.com`
[10]`https://www.mujoco.org`
[11]`https://www.coppeliarobotics.com`
[12]`https://projectchrono.org`
[13]`https://nimblephysics.org`
[14]`https://github.com/google/brax`
[15]`https://developer.nvidia.com/isaac-gym`

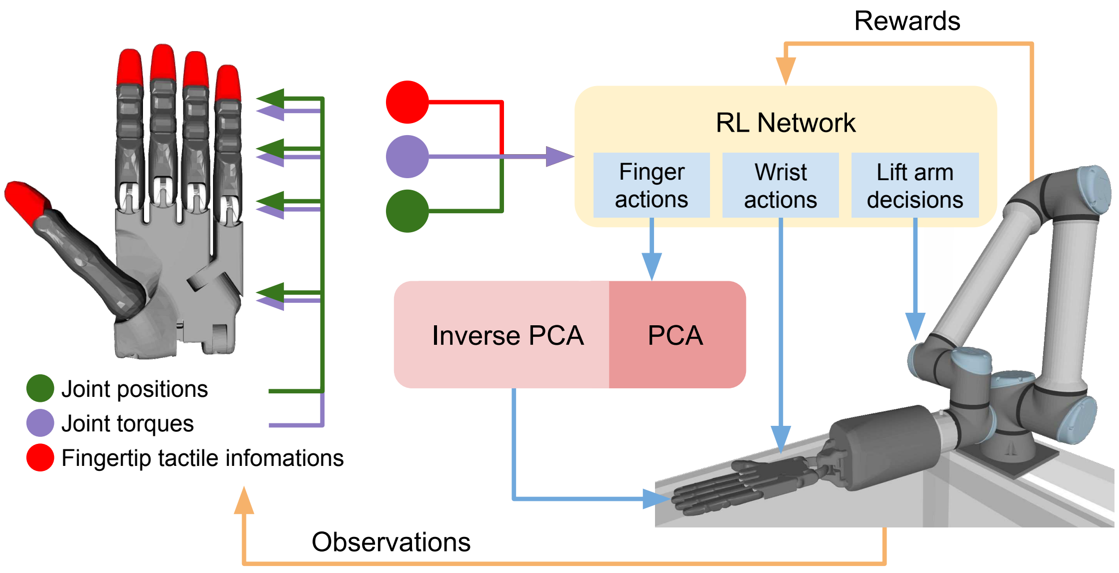


Figure 5.5: An overview of the multimodal reinforcement learning structure. At each timestep, three different types of input information (joint positions, joint torques, and fingertip tactile information) are captured from the environment and concatenated as one vector, representing the agent's current state. This state vector is then stacked with several history states as the input and goes into the RL policy network. Three types of actions (blue lines) result from the policy net, controlling the lifting decision, wrist rotation and finger joints separately.

James *et al.* [53] has developed a great python interface for using CoppeliaSim. Some great simulators also have a lot of users, but it was either not free when we did the work (MuJuCo), or is not public available at the time author began the study (Isaac Gym).

5.4 Multimodal Grasping Policy

In this section, a complete grasp motion can be defined as the following steps:

1. Find a set of pre-grasping poses that describe where to move the Shadow hand.
2. Find a collision-free trajectory to move the arm and hand to a pose from the pre-grasp pose set.
3. Close the five fingers of the Shadow hand using zero value of the Shadow hand as the target, stop the hand motion when any of the fingertips has contact with the object.
4. Perform the RL grasping policy to close all the fingers and lift the object.

The first two steps are the initial grasp generation, in the training, as the object pose is known by the simulator, there is no need to apply any advanced vision based grasp generation, we simply get the object's center position and add an offset that makes the Shadow hand 5 cm above the target object. The object pose will have a small disturbance during the training to make the agent generalize. The initial grasp generation for the real robot is introduced in Section 5.6.2.

The third step is to close fingers to a pre-grasp pose. Starting from all joints of the hand at zero position, the hand moves under the position control commands that go to the PCA zero position where the PCA value is all zero, and use inverse PCA to get the corresponding hand joint positions. When the fingertips of the hand (red part as shown in Figure 5.5) detect the contact, the hand pre-grasp motion ends. Otherwise, the hand will stop until it reaches the target joint positions.

The fourth step is to perform RL grasp policy as shown in Figure 5.5. The RL algorithm we used here is Proximal Policy Optimization (PPO). The agent perform actions according to the environment observations, which contains fingertip tactile information, joint positions and torques. At each timestep, all three information are captured from the agent and concatenated to an array. RL agent using a stochastic policy $\pi(a_t|O_t)$ take the observations and output three actions: finger actions, wrist actions and lift arm decisions. The simulator will perform the action and return the new observations and rewards. In Section 5.4.4 is the details of reward function. The goal of the RL agent is to find a policy π that maximizes the expected sum of discounted rewards over a finite trajectory T. The action value function is defined as:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\tau(\pi)} \left[\sum_{t=0}^{T} \gamma^t r_t \right] \tag{5.1}$$

where $\gamma \in (0,1)$ is the discount factor, τ is the trajectory distribution under the policy π.

5.4.1 Simulation Environment

As shown in Figure 5.1, the simulation and real robot setup are kept the same. The simulator we use is CoppeliaSim [93]. As shown in Figure 5.6, this setup consists of a UR10e robot arm and a Shadow dexterous hand (left-hand version). This simulator provided a plugin to import URDF files to the simulator scene.

There are several modifications of the environment to make the simulator works as expected:

1. In the original URDF file, we defined some dummy objects with no visual but a big collision shape. Like some virtual walls, cameras for our lab's tracking system, and five virtual spheres in the Shadow hand's fingertip. These objects are removed as CoppeliaSim will treat them as visual objects and make the motion planning hard as there are too many collision objects.

2. As the control of the simulated system is based on PyRep [53], some modifications are done to meet the requirements for the robots[16].

3. Several commits are contributed to PyRep for bug fixes and feature requests[17].

[16] `github.com/stepjam/PyRep/blob/master/tutorials/adding_robots.md`
[17] `github.com/stepjam/PyRep/commits?author=lianghongzhuo`

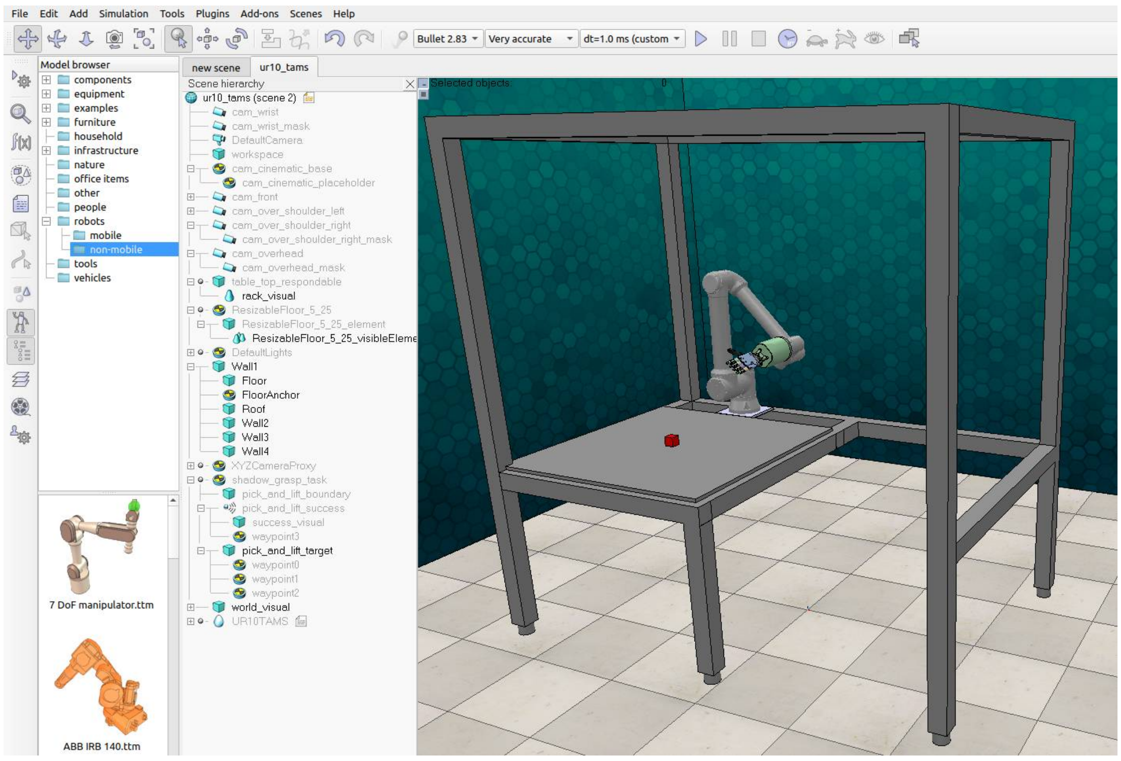


Figure 5.6: CoppeliaSim with the imported UR10e robot arm and a Shadow left hand.

4. In this chapter, the focus is on the interesting motion that fingers interact with the target grasping object. If all links in the hand-arm system are using the actual weight value in the real world, the fingers in the hand are shaking while doing the experiment as the weight of the fingertip is very light compared to the robot arm. A tiny motion displacement will lead to a huge oscillation in the fingertip. To avoid this hand trembling, all the links in hand increase the weight by 100 times relative to its real weight defined in the URDF file by the Shadow company. As the hand is now exceptionally heavier than the real world hand, the simulated arm cannot hold this arm, so the connector link between hand and arm is set to static.

In the simulation, we use 61 objects for training and testing, including 11 objects from the YCB object dataset [11], 14 objects from ShapeNet [12], 5 primitive shapes, and 31 objects from EGAD [82] as listed in Figure 5.7. Since the object pose is known in simulation, we set the initial grasp pose by adding an offset along the z-axis (perpendicular to the table) and a slight disturbance of the x- and y-axis as shown in Figure 5.8.

5.4.2 Observations

Inspired by human hands using multiple modalities to grasp objects, the observations used in this chapter include tactile sensing, torque, and joint angle. Except for joint angles, which are consistent in both simulation environment and real world, other modalities are known hard to transfer from simulation to real world. To transfer the training

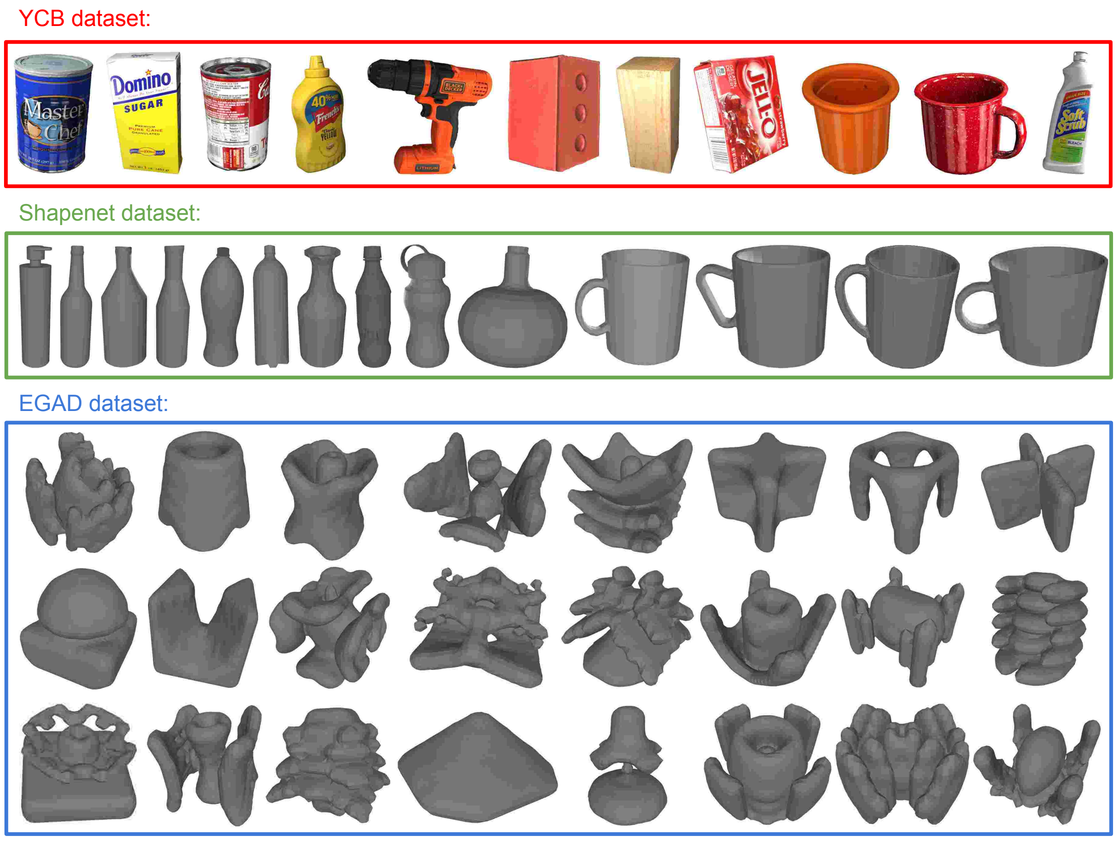


Figure 5.7: Objects used to train the RL agent. This includes 11 objects from the YCB object dataset [11] (red rectangle), 14 objects from the ShapeNet object dataset [12] (green rectangle), 5 primitive shapes (4 boxes and 1 cylinder), and 31 (only 24 shown) objects from the EGAD dataset [82] (blue rectangle).

model directly from the simulation to a real platform without any further training, observations of the agent should be as similar to a real robot as much as possible. Accurate contact force values and joint torque values are notoriously hard to get in simulation environments, and these continuous values are difficult to map to a real robot, as shown in Figure 5.9, the fingertip force is unexpectedly high, and the values have a large oscillate. The high value results because as explained in Section 5.4.1, the simulated Shadow hand was set an unrealistic weight to make the simulation stable. The raw value oscillates because the simulator takes all contacts as rigid bodies, so the contact of two bodies in the simulator is not continuous. In the simulator, when two bodies have a contact, it generates a force against each other and moves the objects apart to make contact disappear. When we control the body to continuously make contact, these two bodies actually keep making contact and going away. Thus the contact force reading is not continuous. If the raw values generated in the simulator are used to train the RL agent, it is clear the RL agent trained in the simulation will be hard to transfer to the real world.

To make the sim-to-real problem easier, the input observations are preprocessed to make the observation space as similar as possible in simulation and reality. In detail, the fingertip tactile information was processed to binary contact information denoted as

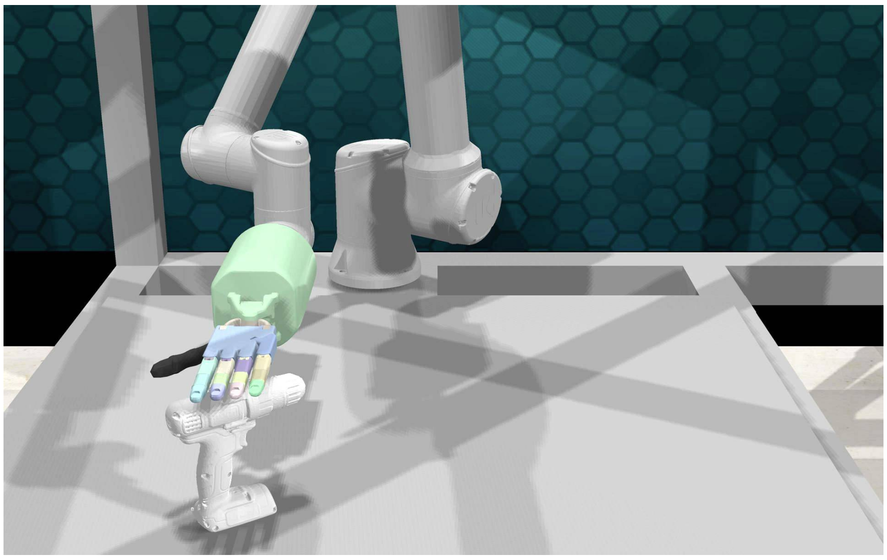


Figure 5.8: Initial grasp pose in the simulator.

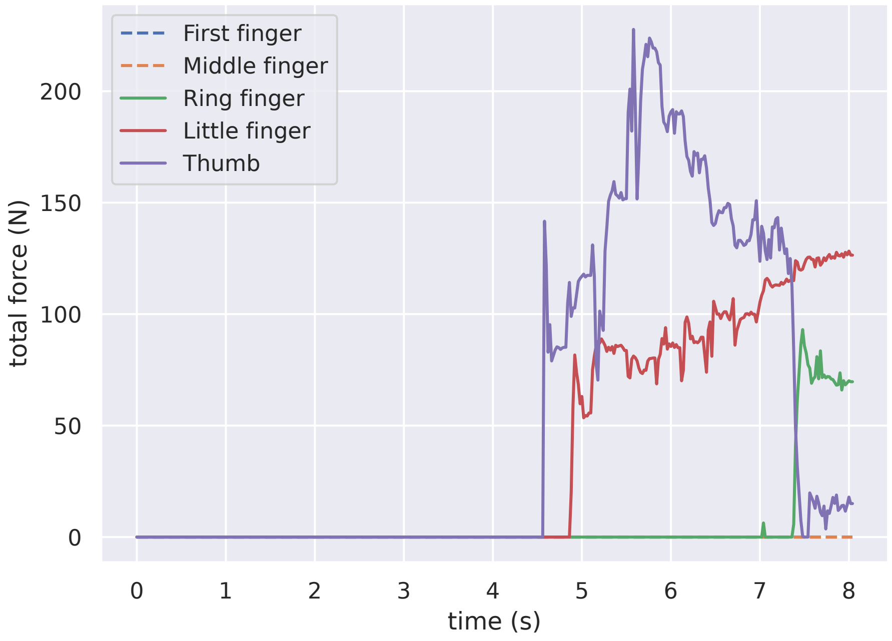


Figure 5.9: Example contact forces of five fingertips read from the simulator during grasping. Note that, in this grasp, only three fingertips are activated (ring finger, little finger and thumb).

$\phi \in \{0, 1\}$ and joint torque value is similar to the fingertip force value, was preprocessed to level-based joint torque denoted as $\tau \in \{0, ..., 5\}$ in the model to minimize the gap between simulation and real scene. The detailed mapping from raw values to the abstracted values is described in Section 5.6.3. The whole observation at $t = t_k$ is then defined as $o_k = \langle \phi, \tau, \theta \rangle$, where ϕ are fingertip tactile information, τ are the joint torques, θ are joint angles of the hand. The previous grasping observations are also useful, as the fingertip tactile information can reason the object's shape during the grasping. To utilize the benefit of historical observations, a RNN based model is proved useful.

So we use h observation timesteps $O_t = \langle o_{t-(h-1)}, ..., o_{t-1}, o_t \rangle$ as our final observation ($h = 3$ in our work).

5.4.3 Actions

The actions generated from the policy have three parts:

1. Principal component increment values $\Delta\epsilon$. Add this value to the current principal component values gives the target principal component value. Then the resulting target joint value θ_h are calculated by the inverse PCA.

2. Wrist joint angle increment value $\Delta\theta_w$. The Shadow hand has two wrist joints WR1 and WR2 as shown in Figure 5.2. Just like a human grasping an object with no vision help, rotation of the wrist can help to adjust the grasp pose. In the robot the wrist motion is also useful when the agent interacts with the object. Adjusting the wrist can expand the exploration space, especially when the initial hand pose is not optimal. The target wrist joint angle θ_w is then calculated by adding $\Delta\theta_w$ with the current wrist joint angle.

3. Lifting decision ρ. This is a discrete action that let the agent decide when to lift the arm to complete the grasp motion. Before the agent decides to lift the object $\rho = 0$, the arm keeps static, and when $\rho = 1$ the agent controls the arm to move w cm higher relative to the current wrist pose. During the lifting, the hand keeps the joint states. The episode ends after the agent finishes the lift arm motion.

The combined action output from the policy is represented as $a_t = \{\Delta\epsilon, \Delta\theta_w, \rho\}$ as shown in Figure 5.5. Then the log action probability can be denoted as:

$$\log \pi(a_t|O_t) = \log \pi(\rho|O_t) + (1-\rho)\big[(\log \pi(\Delta\epsilon|O_t) + \log \pi(\Delta\theta_w|O_t)\big] \tag{5.2}$$

For the finger motions, although the inverse PCA can output 22 joint angles for the five fingers, four pairs of coupled joints need special consideration. The four pair coupled joints are shown in Figure 5.2 with four pink blocks marked as J_1 and J_2. According to the Shadow hand mechanical design, the coupling method for J_1 and J_2 is that J_1 will not move unless J_2 reaches its joint limit ($\pi/2$). So on the controller side, the robot will first take the sum of J_1 and J_2 input value to control J_2 until J_2 reaches $\pi/2$, then the real value for J_1^* and J_2^* will be:

$$J_1^* = \begin{cases} 0, & J_1 + J_2 < \frac{\pi}{2} \\ J_1 + J_2 - \frac{\pi}{2} & J_1 + J_2 \geq \frac{\pi}{2} \end{cases} \tag{5.3}$$

$$J_2^* = \begin{cases} J_1 + J_2, & J_1 + J_2 < \frac{\pi}{2} \\ \frac{\pi}{2} & J_1 + J_2 \geq \frac{\pi}{2} \end{cases} \tag{5.4}$$

5.4.4 Reward

In this multifingered grasping task, a training episode is terminated after the lifting attempt, and then a binary reward $R_s \in \{0, 1\}$ representing whether the object has been picked up successfully is returned. The concrete reward function is defined as:

$$R = \begin{cases} 1 + (e_{max} - e) \times \zeta & \text{if } \rho = 1 \text{ and } R_s = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

This sparse reward function only gives a positive reward when the grasp is successful. Where $\zeta = 0.1$ is episode length extra reward to encourage the agent to lift as soon as possible; e_{max} is the maximum episode length when the agent reaches this episode, the arm will lift and finish the grasping, $e_{max} = 10$ in our case; $e \leq e_{max}$ is the timestep when the arm lifts.

5.4.5 Curriculum Learning

Curriculum learning is a commonly used performance improvement method in the training of machine learning models [64]. Curriculum learning requires the training of a model in a meaningful order, from an easy task to a sophisticated task. The key to the successful use of curriculum learning is how to rank the task difficulties and how to choose the proper pacing function for introducing more complicated tasks.

In our task, we want to train the agent in simulation to grasp objects even though object pose estimation is inaccurate. The initial horizontal position and yaw orientation of the object are randomized in a range related to the learning process $p \in [0, 1]$. Here, the initial pose change should be selected carefully that makes the agent either not too easy to accomplish the task or too difficult (*e.g.*, move the object far away that the hand can reach). The reason that we use object initial pose change as the task difficulty metrics is that during the simulation experiment, we found the initial position and pose of the object change can influence the grasp success rate a lot. In the real robot experiment, the sensor noise and camera calibration error are inevitable. At the beginning of the training, a fixed pose of the object is used for the agent to grasp where $p = 0$ in this case. As the grasp success rate s_r increases, the p increases linearly.

The initial position pos_i, and orientation ori_i of the object can be calculated by:

$$\begin{cases} pos_i = pos_i + \delta_{pos} & \delta_{pos} \in [-\delta_p, +\delta_p] \\ rot_i = rot_i + \delta_{rot} & \delta_{rot} \in [-\delta_o, +\delta_o] \end{cases} \tag{5.6}$$

in which $\delta_p = 0.01 \times p \pm 0.003$ and $\delta_o = 0.1 \times p$ are the variation range of orientation and position, respectively. δ_{pos} and δ_{rot} are both sampled from uniform distribution: $U(-\delta, \delta)$.

5.5 Model Evaluation

In the simulation, we train our agent with different parameters to choose the best performing model regarding grasp success rate and time to converge during training.

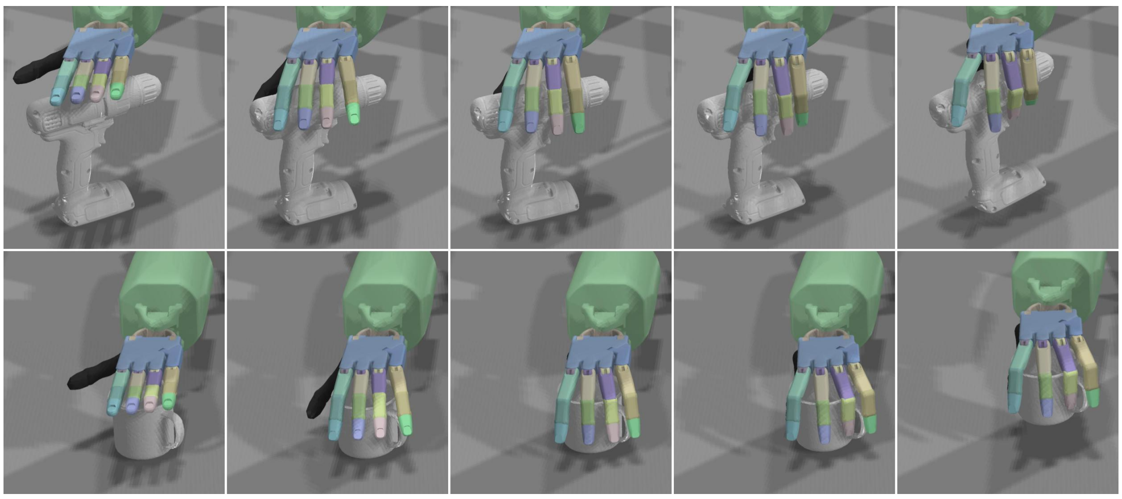


Figure 5.10: Grasp examples using agent `GRU-M3PCA5` in the simulator.

We test our algorithm with different parameters for comparison: 1) Comparing the network performance with different input modalities: using joint angle information only, using joint angle as well as tactile information, and use all the information: joint angle, tactile and joint torques; 2) Comparing networks that use GRU and MLP; 3) Comparing different PCA dimension reduction numbers. We use the pattern `NET-MXPCAY` for naming the models, `NET` represents different network architectures. `MX` where $\texttt{X} \in \{1, 2, 3\}$ means different number of input modalities. `PCAY` where $\texttt{Y} \in \{3, 5, 8, 10\}$ means the dimension reduced from original Shadow hand joint space. All models are trained with three timesteps of history observations as input ($h = 3$). Figure 5.10 shows example grasp experiments using agent `GRU-M3PCA5` in the simulator. From Figure 5.11 to Figure 5.13 show the evaluation result of the above three parameters respectively. Each curve is plotted with five individual runs trained using the same hyperparameters. All models are trained for 1600 episodes.

5.5.1 Comparing Different Input Modalities

This experiment (Figure 5.11) illustrates the grasping performance of different input modalities. The learning curves are similar in the first 500 episodes, after which the learning curve with only joint angles as input (`M1`) begins to drop, then increases back to 80% and drops again. This is because the task difficulty is changing all the time through curriculum learning (section 5.4.5). The agent with two modalities as input (`M2`) exhibits similar instability from episode 1000 for the same reason. As a comparison, the multimodal agent (`M3`) shows the best robustness to the increasing task difficulty.

5.5.2 Comparing Different Network Architectures

This experiment (Figure 5.12) demonstrates that GRU outperforms the MLP architecture. The GRU architecture can better understand the history information, thus got a higher grasp success rate.

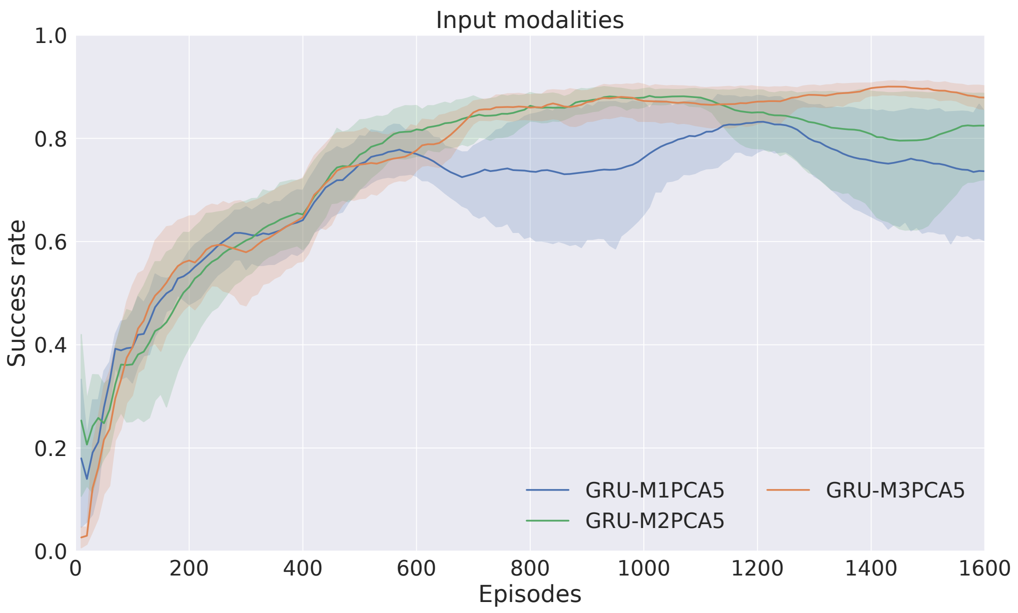


Figure 5.11: Network evaluation result on different input modalities. `M1` means the input has only joint angles. `M2` means the input modalities are joint angles and fingertip tactile sensing, `M3` means the inputs have one more modality: joint torques.

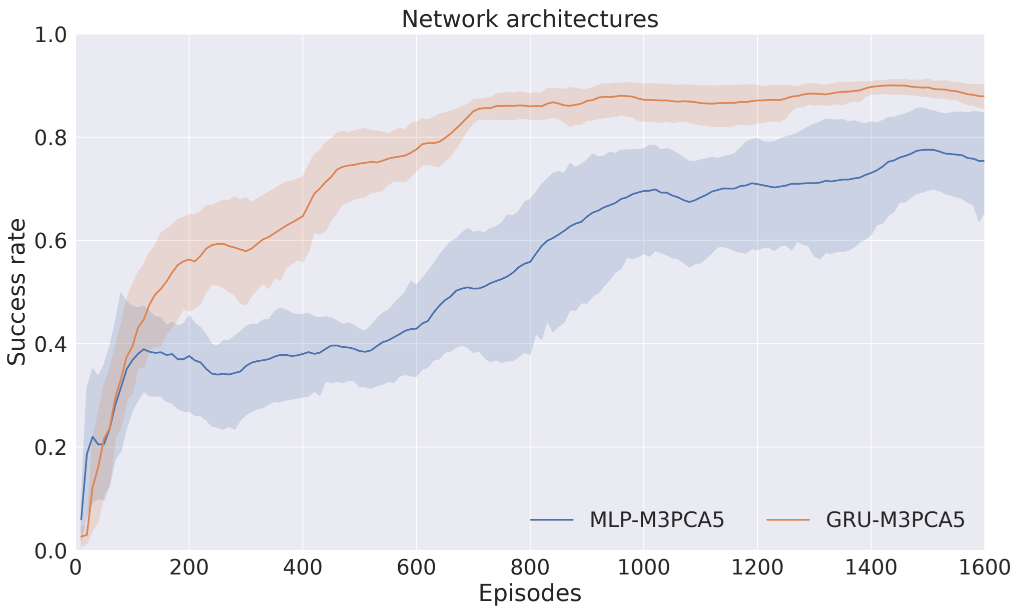


Figure 5.12: Network evaluation result on GRU and MLP network architectures.

5.5.3 Comparing Different Dimension Reduction Dimensions

For this experiment (Figure 5.13), we test the performance using different PCA dimension reduction values. A latent space with a higher dimension means a bigger action space to explore and more dexterous hand motions to learn. The best performance is from the agent with latent dimension space `Y` $= 5$. The model `GRU-M3PCA3` learns faster than other models initially. However, the curve stops growing after episode 500

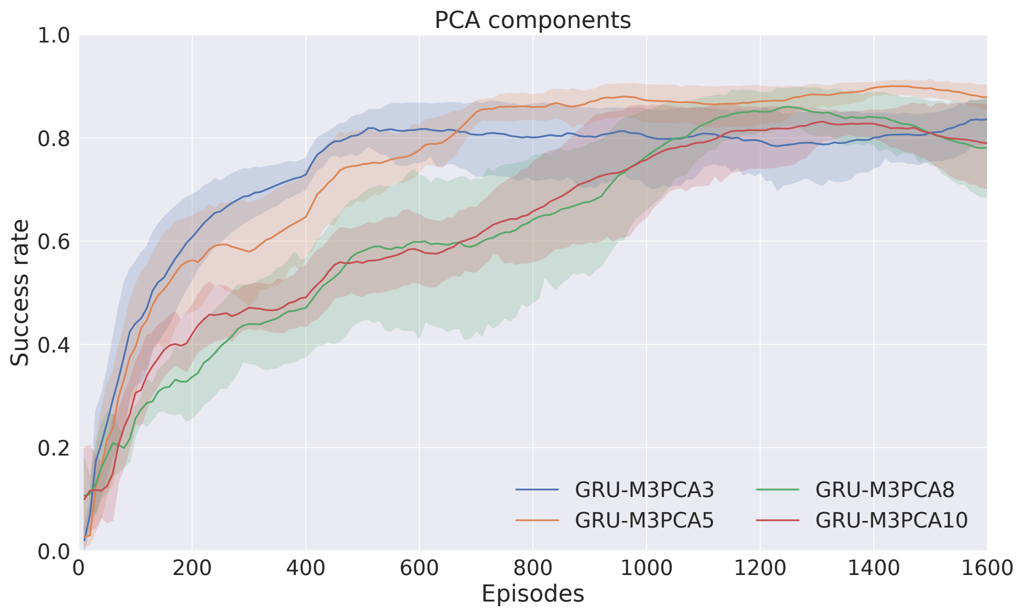


Figure 5.13: Network evaluation result on different dimension reduction values for the Shadow hand. The reduced dimension values tested here are 3, 5, 8 and 10.

and converges at a lower success rate of 80%. This signifies that the latent space is too small to learn dexterous hand motions to grasp all the objects correctly. When we increase the dimension value to 8 and 10, the agents learn more slowly and have a lower success rate than other models, which indicates that the action space is too large and makes learning how to grasp a challenge for the agent.

An interesting find is that the robot tends to use the little finger and the thumb to form a grasp. We guess the reason is that using these two fingers can make the hand cover more object surfaces due to the mechanical design of the hand which approximates the human hand but is not perfect.

5.6 Robot Experiments

For the robotic experiments, the best-performing agent in the simulation is used to execute the multifingered grasping which is agent `GRU-M3PCA5`.

5.6.1 Robot Experiments in Simulation

Although in the previous section, we have reported the overall grasp success rate while training using different hyperparameters, it is still interesting to get the success rate of each object. For each object in the simulator, we performed 100 times of grasp trials and calculate the grasp success rate.

The grasp success rate of each object in the simulator is shown in Figure 5.14. From the figure, we can see that most objects can reach near 100% success rate. However, object with ID 21 has 0% success rate. This is because this object has a small head and all the initial grasp given by the simulator is only above the object. During the grasping,

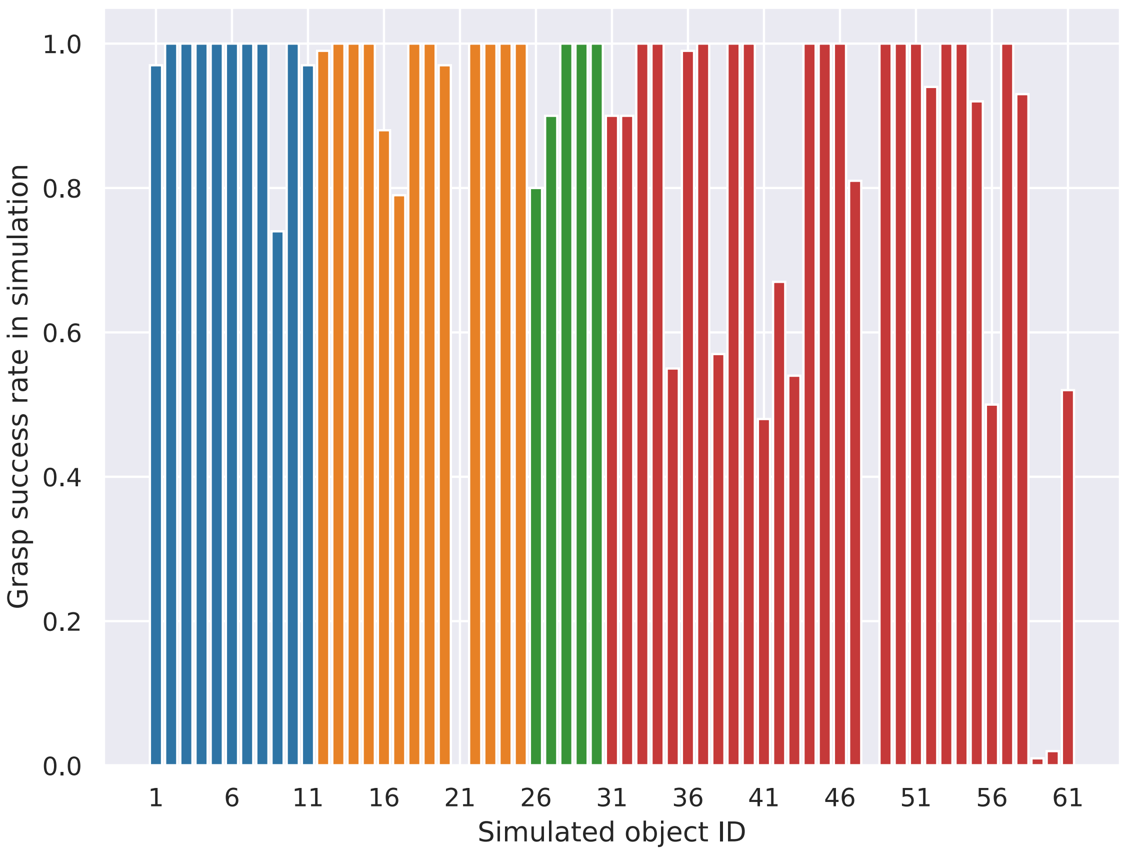


Figure 5.14: Grasp success rate tested in the simulation using agent `GRU-M3PCA5`. From 1-11 are objects from YCB object dataset (blue), 12-25 are objects from ShapeNet object dataset (orange), 26-30 are primitive shapes (green), 31-61 are objects from EGAD dataset (red).

the Shadow hand can not explore and get much useful observation. Objects with ID 48, 59, 60 are from EGAD dataset, the success rate of these objects is low as they are designed to be different to grasp and have very challenging geometries.

5.6.2 Initial Grasp Generation for Real Robot Experiments

Since we do not know the exact object poses in the real-world experiments, we use our previous work [72] to serve as an initial grasp pose generator for real robot experiments. Recall in Chapter 4 that PointNetGPD is a two-fingered grasp evaluation network that takes the partial point cloud near the grasp candidate as input and outputs the grasp quality of the grasp candidate. After motion planning and collision checking using MoveIt [21], the kinematically feasible grasps are chosen as our initial grasp proposals. As these grasps are initially intended for a two-fingered gripper, a proper mapping from a two-fingered gripper to a five-fingered hand is needed, as illustrated in Figure 5.15.

The grasp mapping proceeds as follows:

1. We move all fingers of the Shadow hand to a predefined pose where all the fingers make a **“C”** shape as shown in Figure 5.15 (middle);

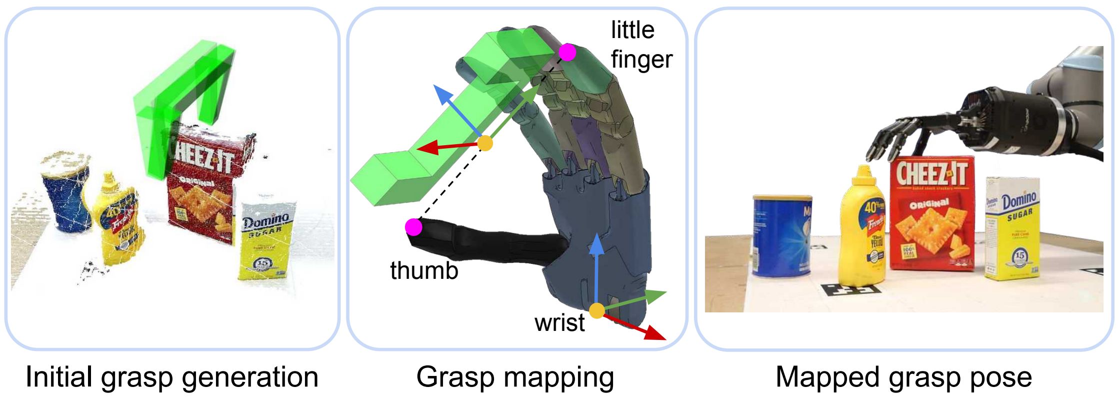


Figure 5.15: Illustration of grasp mapping from a two-fingered gripper to the Shadow hand. (left) The initial grasp generation using PointNetGPD. (middle) Mapping from two-fingered grasp to Shadow hand. (right) Initial grasp pose in the real robot system.

2. The grasp location is defined as the middle point between the fingertips of thumb and little finger;
3. The approach direction of the grasp (x-axis) is the palm norm inverse direction;
4. The y-axis of the grasp is chosen by the connection of the fingertips of thumb and little finger;
5. Then the z-axis can be defined as the cross product of the x- and y-axis.

5.6.3 Sensor Mapping

We need to process the raw data from the tactile and torque sensors to the abstracted sensor data that are used as RL inputs. For the tactile sensor, in the simulator, we detect whether there is a contact between the fingertip and object to determine the mapping values. On the real robot, we press each fingertip sensor manually and record the raw reading to get the upper sensor range of each finger. The lower range of the tactile sensors is calculated by keeping the hand still, reading the sensor raw value ten times, and getting an average. Then we mark a tactile observation as 1 if the sensor reading is higher than a threshold value of 0.3% of the total range, otherwise as 0. For torques, in the simulator, we directly map the measured joint torques to the discrete levels. However, in the real robot, as the Shadow hand does not provide torque reading out of the box, we use the measured tendon force of each motor instead. When doing experiments with the Shadow hand, we also found that the tendon reading τ_j will drift after some time. Therefore, we take the tendon reading when the hand is empty at the beginning of each grasp attempt and set it as the initial tendon value $\tau_{j,0}$ for each joint. For mapping the torque data, we use empirical thresholds $\in \{-200, -100, 0, 100, 200\}$ for the real robot and $\{-20, -10, 0, 10, 20\}$ for the simulator to map the reading $\tau_j - \tau_{j,0}$ to the level-based reading $\in \{0, 1, 2, 3, 4, 5\}$ expected by the RL agent.

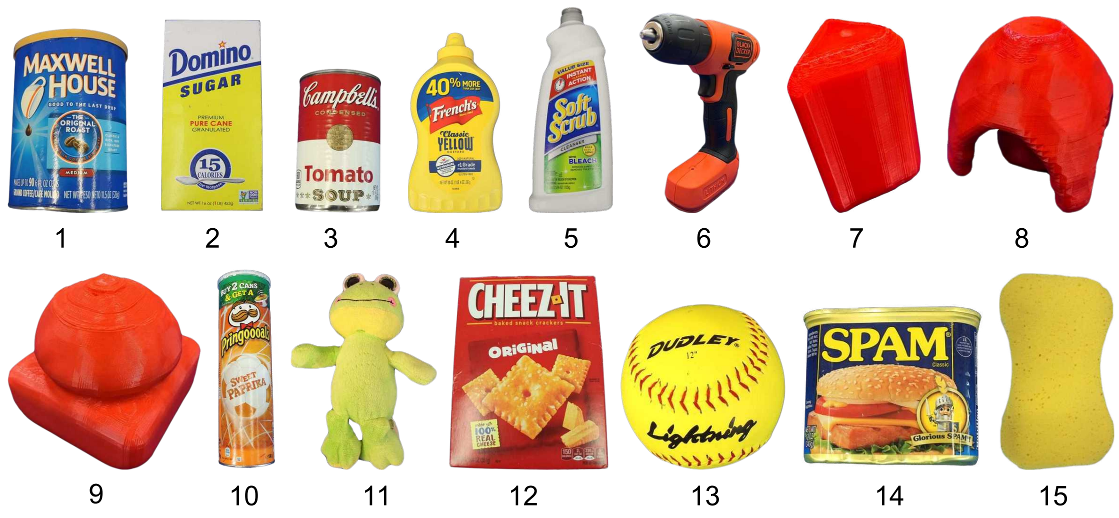


Figure 5.16: Objects used in the real robot experiments. Objects with IDs from 1 to 9 are in the training dataset, and objects with IDs from 10 to 15 are unseen by the multifingered grasping agent. Objects 7, 8, and 9 are 3D printed models from EGAD.

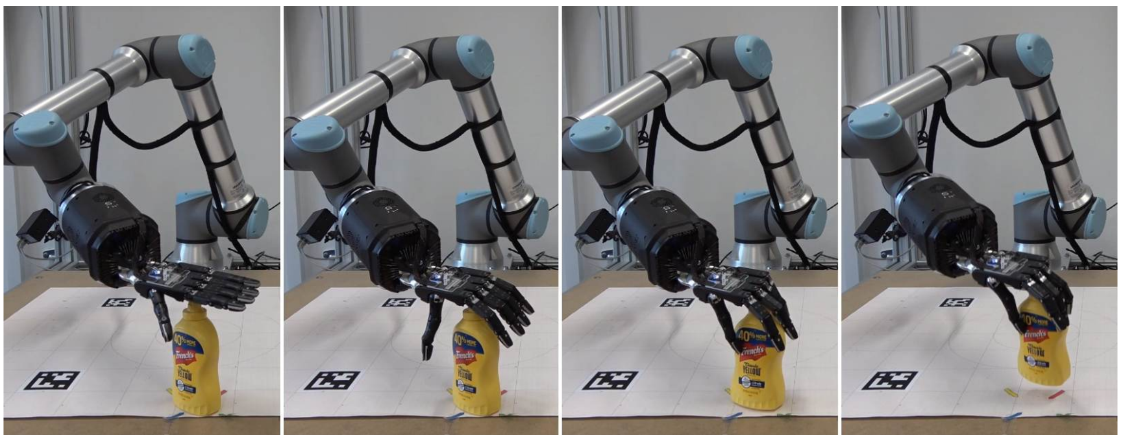


Figure 5.17: Grasp example in real robot environment. The agent used in these grasps is `GRU-M3PCA5`.

5.6.4 Real Robot Verification

For the real robot experiments, we selected 15 objects, including some objects used during training (object ID 1 to 9) and novel objects (object ID 10 to 15), as can be seen in Figure 5.16. A Kinect2 depth camera is used to get the object point cloud required for initial grasp pose generation in the actual robot experiments. For each object, we conduct ten grasp trials for each of the below four different agents. The first two agents are our `Baseline1` and `Baseline2`, respectively, which use a hard-coded sequence to close the fingers for all the grasp trials. In `Baseline1`, we set a torque limit for each joint and control the hand in position mode. All active joints are controlled to track the given trajectory until reaching the target positions or the tendon force limit. In `Baseline2`, besides joint position and joint torque-sensing, we add one more modality: tactile sens-

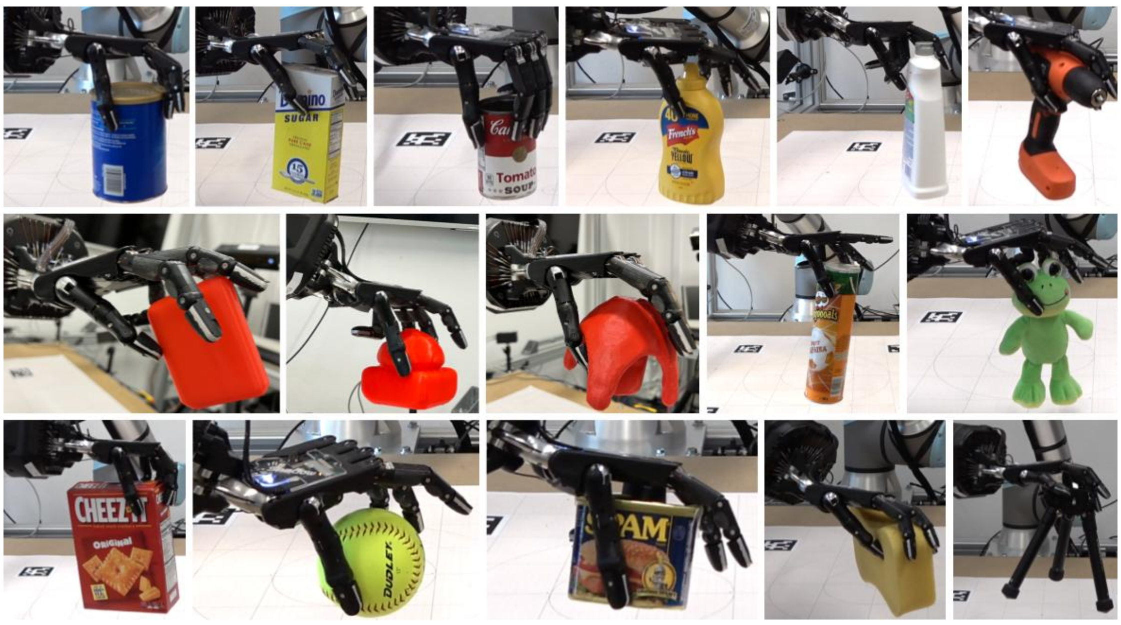


Figure 5.18: Grasp examples for each object. The agent used in these grasps is `GRU-M3PCA5`.

ing. The first finger, middle finger, and ring finger will stop closing if the tactile sensor on the fingertip is triggered, which helps to prevent from over pushing the object beyond a proper grasping position. The third agent uses the `GRU-M2PCA5` model. The fourth agent uses the `GRU-M3PCA5` model.

To perform a fair comparison of the experiments to demonstrate the difference between agents when the grasp pose is the same, we first mark the target object in a fixed position on the table. Then we use a point cloud and PointNetGPD to generate ten initial grasp poses that are limited to top-down grasps. After this, we perform the finger motion with the above four different agents. The grasp success rates in Table 5.2 indicate that the RL method can outperform the baseline method in most objects, which establishes that the agent trained using RL has learned a robust grasping strategy. The RL agent trained using three modalities performs better than the RL agent using two modalities. We also see that the success rates for objects 4 and 5 are quite low. This is because these objects are smaller on the top, which needs more precise grasping actions. From the experiments, we find out that the average episode length in simulation and real experiments is 5.8 and 6.2 for model `GRU-M3PCA5`. The action frequencies of the models are 1.8 Hz and 1.6 Hz in simulation and the real world. Besides the random grasp experiments, we also conduct a grasp experiment where we fix our object pose, and initial grasp pose for all three agent conditions to show the difference between agents when the grasp pose is the same. Figure 5.17 shows an example of grasp sequence in real robot experiment. Grasp examples on the real robot are shown in Figure 5.18.

Table 5.2: Robot experiment result for multifingered grasping. The results of object ID 12 and 15 are not shown as they got a 100% grasp success rate in all methods. The definition of object ID is in Figure 5.16.

Object ID	1	2	3	4	5	6	7	8	9	10	11	13	14
`Baseline1`	80%	70%	40%	40%	20%	20%	50%	50%	80%	60%	60%	50%	70%
`Baseline2`	70%	80%	40%	40%	30%	20%	40%	60%	80%	70%	60%	60%	**80%**
`GRU-M2PCA5`	60%	90%	**80%**	**60%**	**50%**	**80%**	60%	**70%**	90%	**100%**	90%	90%	**80%**
`GRU-M3PCA5`	**100%**	**100%**	**80%**	50%	40%	**80%**	**70%**	**70%**	**100%**	**100%**	**100%**	**100%**	70%

5.7 Discussion and Summary

This chapter proposes a novel hand-arm multifingered grasping system to solve the autonomous multifingered grasping problem. We first build a hand pose dataset to teach the dexterous Shadow hand how humans commonly move their hand during grasping by mapping human motions to the Shadow hand. A PCA-based hand synergy is then trained to reduce the dimension used to control the hand, which accelerates the training speed of a grasping agent. Then we build a simulation environment to train the RL agent. Detailed simulation trials with different parameters demonstrate that our agent works best with three modalities as input and a GRU network architecture. Real robot experiments show that the trained RL agent can be applied in the real world even if the model is trained purely in simulation. Our method clearly outperforms the baseline method.

Training the robot to do object in-hand manipulation tasks such as tool use will be our ongoing work. Domain randomization [115, 117] plays an important role in bridging the gap between simulation and the real world. The dynamic parameters such as friction, object mass, and even moment inertia will be randomized in a reasonable range. We also plan to accelerate the simulation time with a differentiable physics simulator. Besides, adding vision to the agent to let the agent find the initial grasp by itself is also an exciting research direction.

Chapter 6

Making Sense of Audio Vibration for Robotic Pouring

In this chapter, we focus on the challenging perception problem in robotic pouring. Most of the existing approaches either leverage visual or haptic information. However, these techniques may suffer from poor generalization performances on opaque containers or concerning measuring precision. To tackle these drawbacks, we propose to make use of audio vibration sensing and design a deep neural network AP-Net* to predict the liquid height from the audio fragment during the robotic pouring task. AP-Net* is trained on our collected real-world pouring dataset with multimodal sensing data, which contains more than 3000 recordings of audio, force, video and trajectory data of the human hand that performs the pouring task. Each record represents a complete pouring procedure. We conduct several evaluations on AP-Net* with our dataset and robotic hardware.

6.1 Introduction

Robotic pouring [110] is a crucial robotic task in both domestic and industrial environments. In a nutshell, a robot is required to pour a liquid from one container to another while preventing it from spilling. Therefore, the robust and accurate perception will play an essential role in this task, especially in estimating the liquid height in the target container. Recent approaches to solving this perception problem mostly rely on visual sensing [101, 102, 83, 100]. Using a camera situated in front of the target container, the current liquid height can be regressed from the visual features of the captured image. However, these approaches cannot generalize to opaque containers since the liquid height cannot be seen or could suffer from poor estimation errors. On the other hand, haptic sensing is another important modality for the perception of robotic pouring. For example, when the force and torque feedback from the manipulator is available, we can either estimate the volume of liquid being poured or directly learn a pouring policy in an end-to-end manner [94]. However, the correlation between haptic information and the pouring liquid can be rather complicated and are varied among different end effectors and containers. These drawbacks in existing perception methods suggest that the robust and accurate perception in robotic pouring still remains an open problem. To sum up,

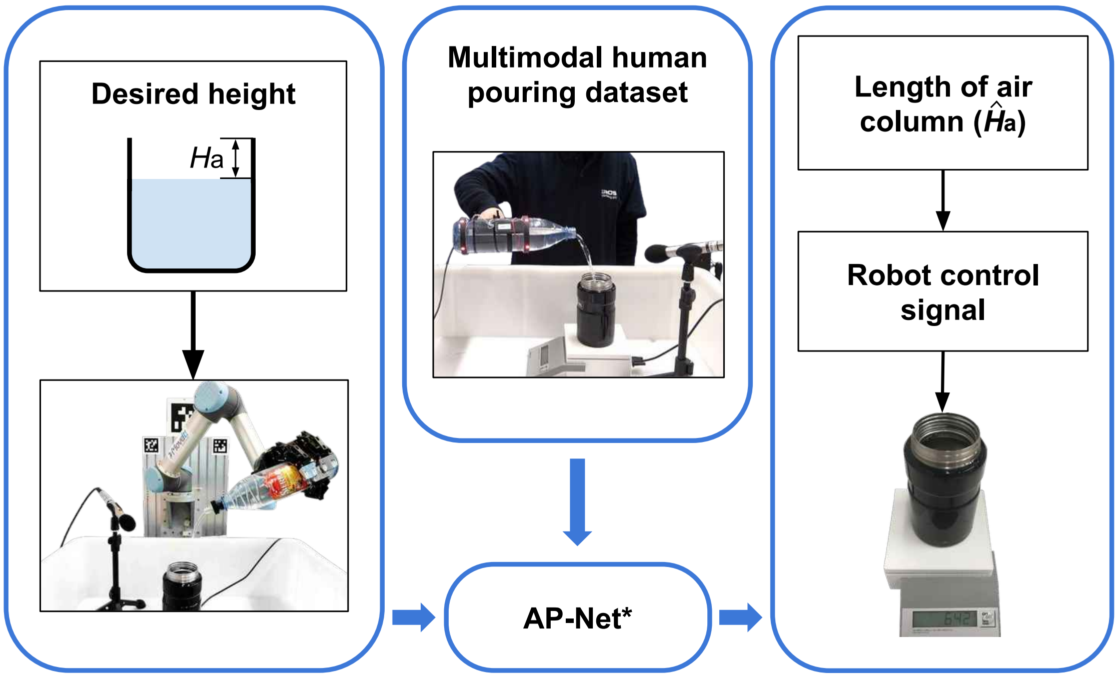

Desired height
Ha
Multimodal human pouring dataset
Length of air column ($\hat{H}a$)
Robot control signal
AP-Net*

Figure 6.1: The robotic pouring system. (Left) Given the robot a target liquid level, audio vibrations during the pouring manipulation by a robot are recorded by a microphone then fed into AP-Net*. (Center) AP-Net* is trained offline to predict the length of the air column of the target container from our multimodal pouring dataset. (Right) The length of the air column H_a predicted by AP-Net* is used to guide the robot's pouring control signals.

the major challenges in the perception for robotic pouring are twofold:

1. **Generalization**. The perception in robotic pouring should be able to generalize to different containers, liquid type, and liquid status.

2. **Precision**. The estimation result, *i.e.*, the prediction of liquid height, should be accurate enough to satisfy the requirement in pouring task.

We propose to tackle these issues by leveraging the modality of acoustics. Inspired by how humans judge the liquid height during pouring with their hearing, we try to design a model that can estimate the position of liquid height from audio vibration. This is based on the observation that the vibrational frequency of the air in the container will change as the level of liquid rises during the pouring procedure. Moreover, estimating liquid height using audio vibration is immediate. Thus, there is no need to explicitly perform an integration, which further reduces the prediction bias and achieves more accurate results.

In this chapter, we introduce a deep network called AP-Net* that utilizes the audio vibration to estimate liquid height for robotic pouring. Specifically, the main part of AP-Net* is a RNN that maps an audio fragment into the prediction of the current height of one liquid in the target container. Figure 6.1 illustrates the perception pipeline in our proposed method for robotic pouring. There are two considerations on choosing a recurrent structure for our AP-Net*. Firstly, using RNNs was already shown to be effective in audio analysis due to the inherent sequential properties of audio itself. Secondly, a recurrent structure can implicitly integrate the prior knowledge that the liquid level rises up monotonically during the pouring, which may help the model to reduce the noise and to predict more smooth results. These two considerations are verified in both our dataset and robotic experiments.

Recent successes in deep neural network based robotic perception methods [60, 109, 105, 45, 49] emphasize the importance of training on large-scale datasets. To further improve the performances of the proposed method, we built up a dataset which contains more than 3000 records of motion trajectories from human demonstrations including audio-frequency recordings, force and torque, and video flows.

In summary, our key contributions are twofold:

- We propose to tackle the challenging liquid height estimation task in the perception of robotic pouring by leveraging the audio vibration data and a neural network model with a recurrent structure. Extended experiments on dataset and robotic hardware demonstrate that our method can facilitate a robust and accurate audio-based perception for robotic pouring.
- We built a large-scale multimodal dataset with a focus on the perception task for robotic pouring, which contains more than 3000 human pouring sequences. To the best of our knowledge, this is the first multimodal dataset for the perception task in robotic pouring.

6.2 Data Preparation

6.2.1 Multimodal Pouring Dataset

Training a regression model which learns the correlations between the audio vibration and the liquid height in the receivers relies on a meaningful pouring dataset. We collected a multi-sensor dataset by humans in a quiet environment. Besides the audio data and liquid height labels (measured indirectly via a digital scale under the target container), we also recorded videos of the whole pouring procedure, the trajectories of the human hand that performed the pouring by a motion tracking system, and the haptic feedback of the end effector of the manipulator via a torque sensor on the source container. The cost of collecting dataset by humans is comparatively low compared to a robot. However, this approach brings the challenge of transferring a model learned from this dataset to robotic pouring due to the different pouring trajectories and loud background noise.

Our dataset setup is shown in Figure 6.2. It includes a source container, three different target containers with three different materials (referred to as glass, Thermos, and mug, shown in the first three items of Figure 6.9 on page 80), a Behringer B-5 microphone (44.1 kHz), an ATI Mini40 force/torque sensor (500 Hz), a Maul Logic digital scale (1 Hz), a Logitech web camera (30 Hz), and a PhaseSpace Impulse X2E motion tracking system (240 Hz). The height of the glass, Thermos, and mug are respectively 127 mm, 150 mm and 99 mm. We placed the source containers relative to the bottom

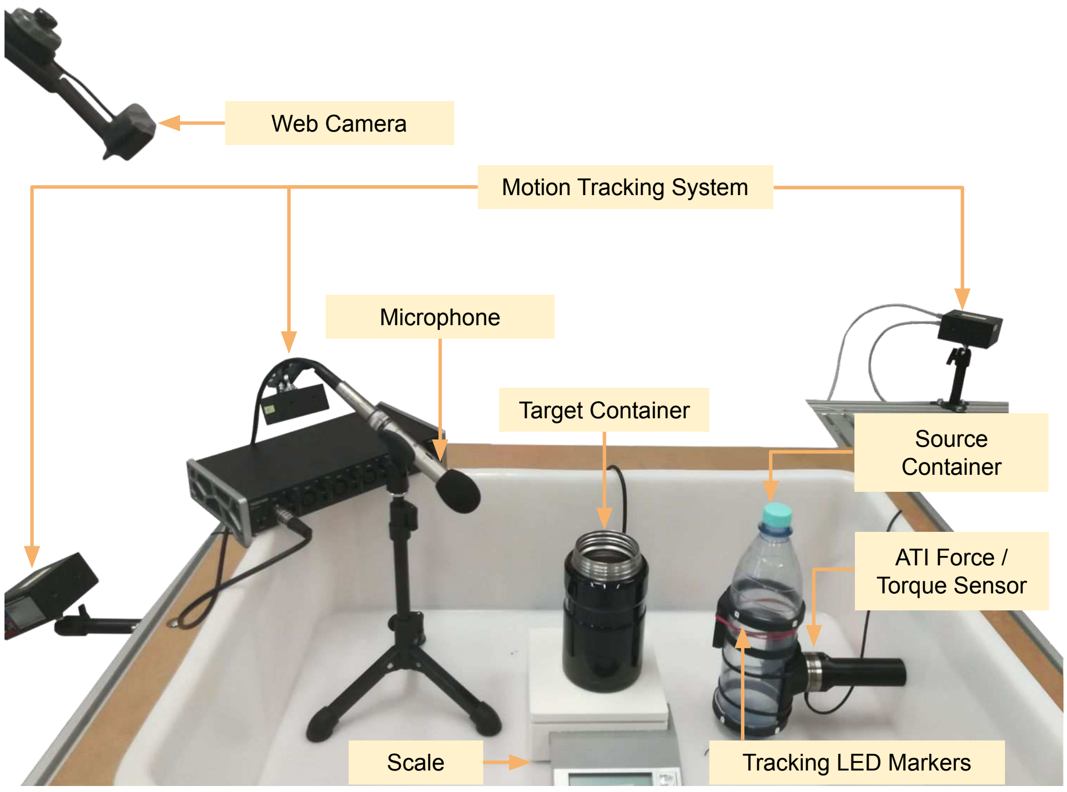


Figure 6.2: Setup used to collect multimodal human pouring dataset.

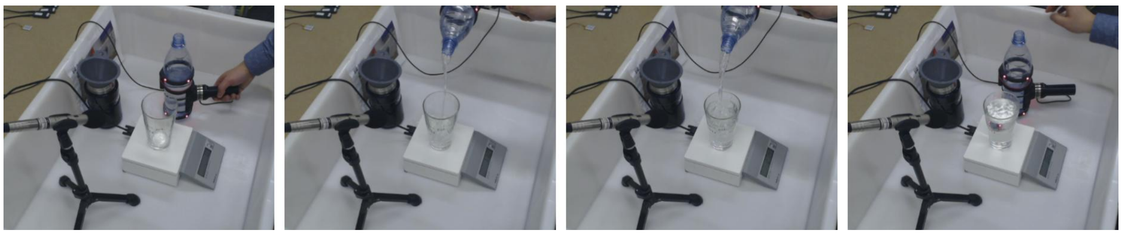


Figure 6.3: A pouring example recorded by a webcam during the human pouring data collection.

center of the microphone at a horizontal distance of 250 mm and a vertical distance of 750 mm. For each pouring trial, the subject held the handle of the source container and started pouring task at an angle varying between $8^\circ \pm 15^\circ$ and at a random position which is relative to the mouth of the target container ranging from 450 mm to 500 mm. Pouring during the training only involved water.

In this manner, we respectively collected 1000 trials for three target containers involving two subjects. For each trial, we took multimodal data right before the scale reading started to change, and right after the scale reading became stable. The lengths of one pouring recording varied from 4 seconds to 11 seconds. Although we only consider the modality of audio vibration in this chapter, future work on perception methods with multimodal fusion can be facilitated with our dataset.

6.2.2 Data Analysis

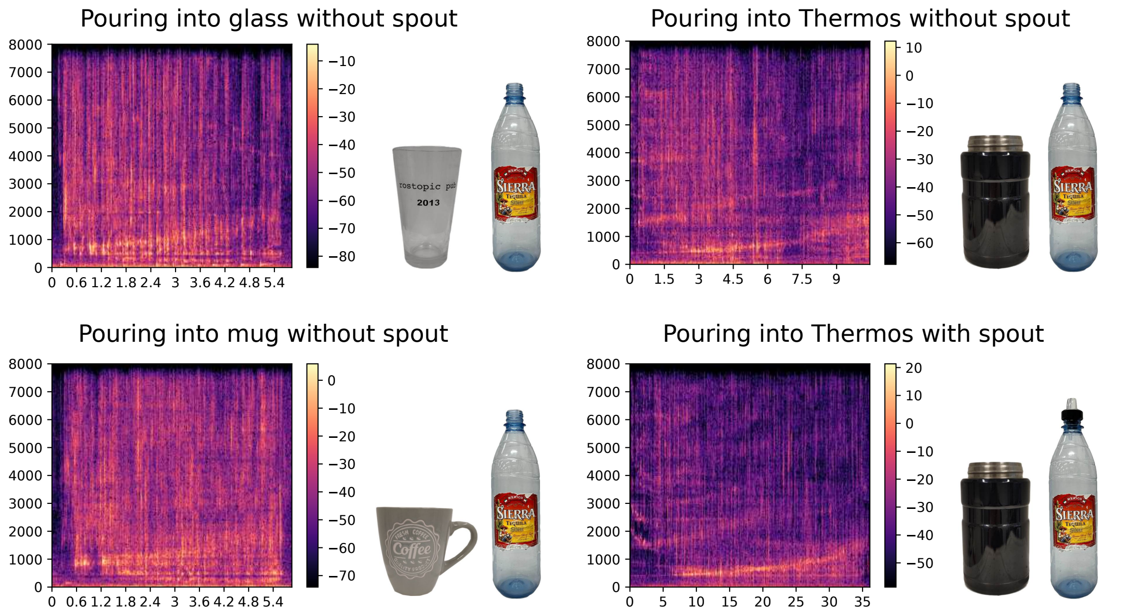


Figure 6.4: Examples of audio spectrograms in our dataset. The x axis of the spectrograms represents time (seconds), and the y axis represents the frequency (Hz). From left to right, top to bottom. The first three spectrograms are generated from the audio signals pouring from a bottle without a spout pouring into a glass, a Thermos, and a mug. The last figure is the spectrogram for a bottle with a spout pouring into a Thermos. The spout is in Figure 6.5.

Audio-frequency data

When pouring the liquid into a container, as the liquid height rises, two distinctive resonance frequencies will change. First, like in an organ pipe, when the length of the air column of the container gets shorter, the air vibrates faster and the resonance frequency of the air increases. Second, the resonance frequency of the container itself becomes slightly lower because of the resistance to water pressure [38].

We resample all audio data to 16 kHz and computed spectrograms with a window length of 0.032 seconds and a half-window overlap. We use a Short-Time Fourier Transform (STFT) frequency resolution of 512, and this generates audio slices with 257 descriptors. As shown in the four spectrogram examples in Figure 6.4, one high-energy and rising curve between 256 Hz-2048 Hz is clearly visible in all spectrograms. This curve represents the resonance frequency of the air. Meanwhile, the bottom two spectrograms in Figure 6.4 demonstrate that different pouring speed does not change the main theory of the resonance frequency of the air. We propose that the rising of the resonance frequency of the air actually depends on the liquid height and not on the much heavier and more heavily damped liquid height. Therefore, using audio vibration to estimate the real-time liquid height could make sense. But the resonance frequency of the target container which should principally have a slight downtrend cannot be clearly seen in our dataset.

Figure 6.5: The spout equipped on the source container in robotic experiments.

Scale data

The weight data measured by the scale is used to calculate the liquid height. To accomplish the real-time perception task, we deploy linear interpolation to the initial weight recordings because of the low sample rate of the scale. We respectively generate a quadratic polynomial with fifteen pairwise weight and height measurements for each target container. Afterward, we calculate the corresponding liquid height using the polynomial from the interpolated weights. It follows that the height data is a 257×1 times continuous series.

6.3 AP-Net* Architecture

Our goal is to acquire the desired filling height of the liquid through learning a robust model based on audio vibration. Although the correlations between the resonance frequency of the air and the liquid height are time-independent, the pouring task is a sequential and variable-length problem. Instead of choosing a simple feed forward network to reach our goal, we make use of the RNN to exhibit and learn the real-time height of the liquid. There are two popular recurrent units we can use: LSTM unit and GRU [16], both of which alleviate the vanishing gradient problem in a gating mechanism.

Furthermore, determining a well-suited ground truth is also crucial in supervised learning. The physical model suggests that the increase of resonance frequency in pouring motions results from the reduced length of the air column. And this conclusion exists on different types of target containers. Intuitively, we deduce that target containers with an air of the same height have a similar resonance frequency although they are of different shapes and different filling heights. That is to say, in our task, to make sense of audio vibration, using the length of the air column as ground truth could be more generative and indicative than using the height of the liquid level. For example, in Figure 6.6, two target containers are filled at different liquid heights but have an air column of the same length. Otherwise, if we take the height of liquid level as the ground truth, we would have one label corresponding to different audio-frequency features.

With the above considerations, we design a recurrent deep network (AP-Net*) P_θ to predict the length of the air column H_a. θ defines the parameters of our proposed AP-Net*. The network architecture is shown in Figure 6.7.

In order to augment our dataset, we randomly choose audio clips with a length of 4 seconds from one complete pouring audio sequence. The number of audio clips is pro-

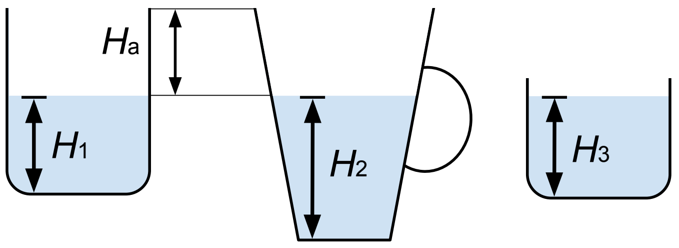


Figure 6.6: Illustration of length of the air column. In first and second containers, different target containers are filled to different height of the liquid, but have the same length of the air column H_a and a similar resonance frequency of the air when the liquid is poured into them. Compared with first and third container, the liquid height H_1 and H_3 are same. However, the third container container is almost full and first container is only half fill. Thus, they have different sound patterns during the pouring.

portional to the length of a pouring trial. We take the raw 4 seconds audio fragment and transformed them into 257×251 window slices. Then each time slice of the spectrogram is progressively fed into the encoder module (1-layer LSTM/GRU unit) to a layer of 256 recurrent features A_h. The height predictor (a 2-layer MLP) takes the recurrent vector as input and performs regression of the temporary length of the air column. The height predictor is supervised with a mean squared error (MSE) loss $\mathcal{L}_{height}$

$$\mathcal{L}_{height} = \|\hat{H}_a - H_a\|^2. \tag{6.1}$$

In addition, leveraging the principle that the liquid height in the target container increases monotonically, we introduce an auxiliary $\mathcal{L}_{mono}$ to enforce the estimated length of the air column being decreasing along the time t

$$\mathcal{L}_{mono} = \sum_t [\max(0, (\hat{H}_{a_{t+1}} - \hat{H}_{a_t}))]. \tag{6.2}$$

Overall loss. Combining both $\mathcal{L}_{height}$ and $\mathcal{L}_{mono}$, the complete training objective for AP-Net* is defined by $\mathcal{L}_{audio}$

$$\mathcal{L}_{audio}(\theta) = \mathcal{L}_{height} + \alpha \cdot \mathcal{L}_{mono}, \tag{6.3}$$

where α is a hyperparameter for balancing these two loss functions. In our implementation, we set it to 0.01 for the best performances via some preliminary experiments.

6.4 AP-Net* Evaluation

We examined which structure of AP-Net* could learn the most indicative representations during the pouring events. To explore the appropriate encoder module of the audio branch, we designed a basic feed forward network (two layers of MLP for regression) as one baseline. Moreover, to find out the recurrent units that could maintain the consequent frequency memory better, we evaluated two popular recurrent units: LSTM and GRU. We refer to these structure as AudioFC, AudioLSTM, and AudioGRU respectively. There were two evaluation metrics used: 1) the fraction of audio sequence whose

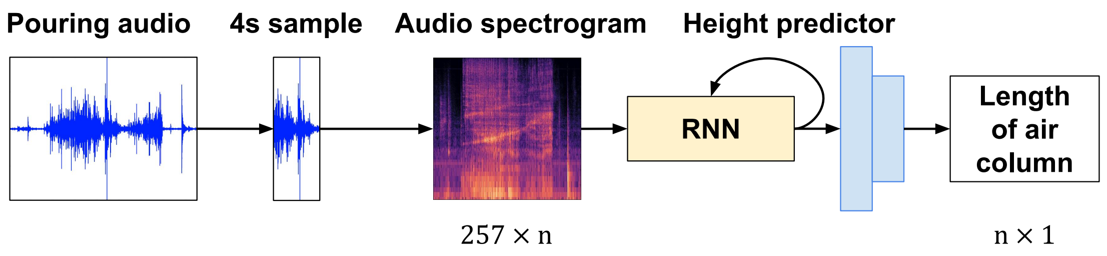


Figure 6.7: AP-Net* architecture. The raw audio data is transformed into a spectrogram with 257 descriptors. Then the encoder module (RNN unit) is progressively fed each time slice of audio-frequency spectrogram. Finally, the height predictor module produces the 1D length of the air column of the target containers. The blue rectangular denotes a Fully-connected (FC) layer following with a batch normalization layer and a rectified linear unit.

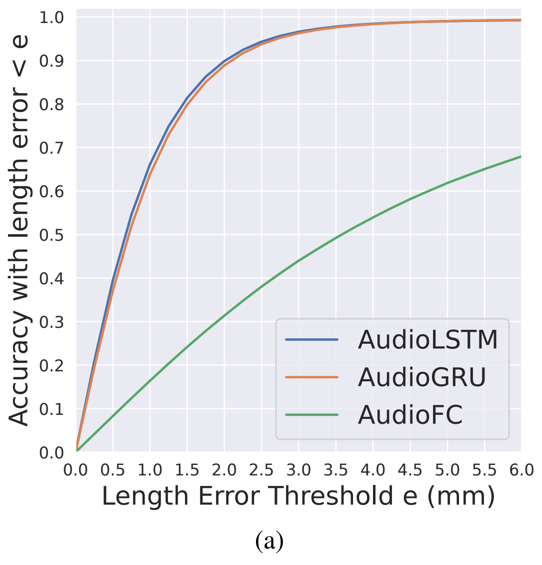
air column length prediction error is below a threshold; 2) the average height prediction error on each target container (including on all the containers).

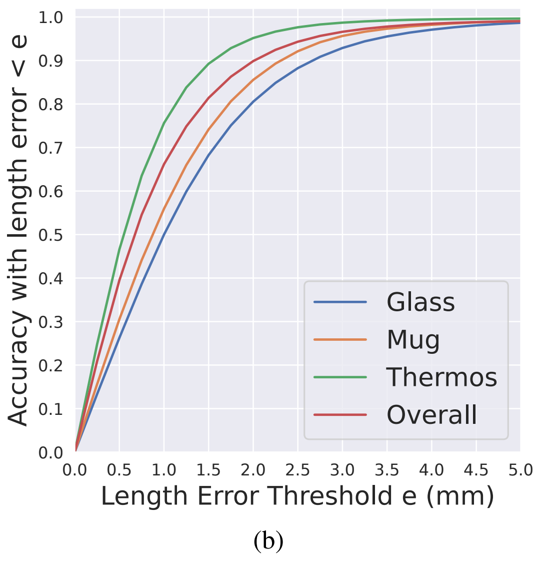
The evaluation results on the testing set are shown in Figure 6.8(a) and Figure 6.8(c), which indicate that the recurrent architectures are significantly superior to the AudioFC baseline. Both recurrent architectures achieve 90% accuracy below an absolute 2 mm length error, and the absolute mean length errors are below 1.5 mm. The results not only show that the recurrent architectures can integrate the prior knowledge from audio sequences, but also verify that the audio vibration is eligible to infer the liquid height between different target containers. On comparison of these two recurrent architectures, the AudioLSTM architecture slightly outperformed AudioGRU architecture on a single target container and all target containers, demonstrating its advantage of having more trainable parameters when trained with our large training set.

Figure 6.8(b) manifests the results of AudioLSTM model trained on the training set with only a single target container or all target containers (shown as Overall). As shown in this plot, on the Thermos cup it outperforms the other two target containers and the glass shows the worst performance. This is mainly due to the different materials of these containers. For example, the stainless steel material makes the crispest sound. Furthermore, we can see that the model trained on our complete training set with all types of containers achieves the best accuracy, which suggests that it benefits a lot from a larger dataset and could embody a better-generalized ability than any models that trained on only a part of the dataset.

6.5 Robot Experiments

In this section, we evaluate the adaptability and robustness of our audio-based perception method in pouring experiments with a UR5 robot. In the experiment setup, we fixed the pouring trajectory of the robot and the source container position in the robot's gripper. To improve the quality of the recorded pouring sound, we fixed the center of the gripper right above the top center of the scale at a height of approximate 310 mm. Unfortunately, the high position of the gripper gives rise to a high possibility of spillage during pouring. We solved this problem by equipping the source container used for pouring with a thin





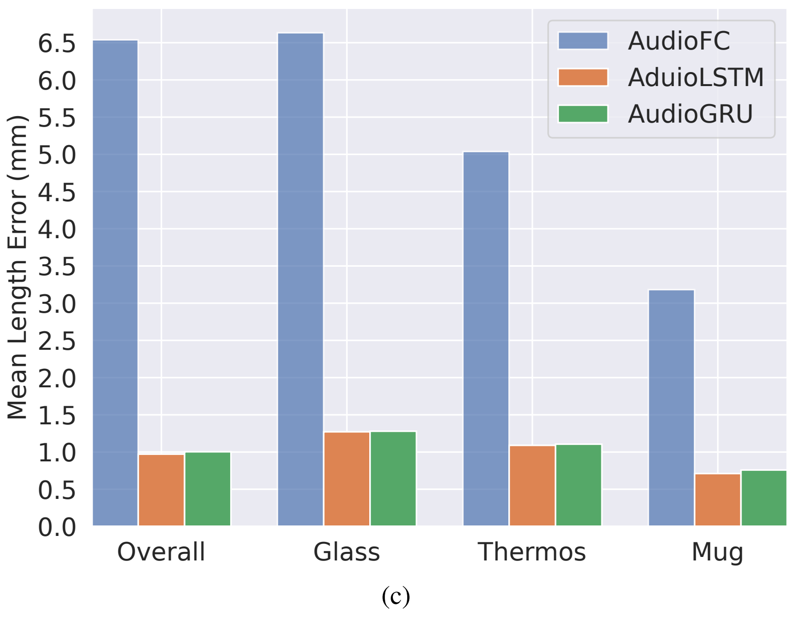


Figure 6.8: Evaluation results of AudioFC, AudioLSTM and AudioGRU models tested on the human pouring dataset. Evaluation results of the fractions of audio sequence whose the length of the air column's errors are below a threshold (a) by all models on our overall test data and (b) only by AudioLSTM model tested on overall test data and each single target container's dataset. (c) Comparison of length errors on the different target containers evaluated by all models.

spout shown in Figure 6.5. Accordingly, the pouring speed was drastically reduced due to the spout, compared to the dataset with human pouring. Therefore, we re-collected a small dataset of 30 trials for each target container using the source container with a spout.

We first selected the AudioLSTM model which was trained on our original dataset with all target containers to estimate the real-time length of the air column. Then, we fine-tuned this AudioLSTM model on our new dataset collected with a spout on the

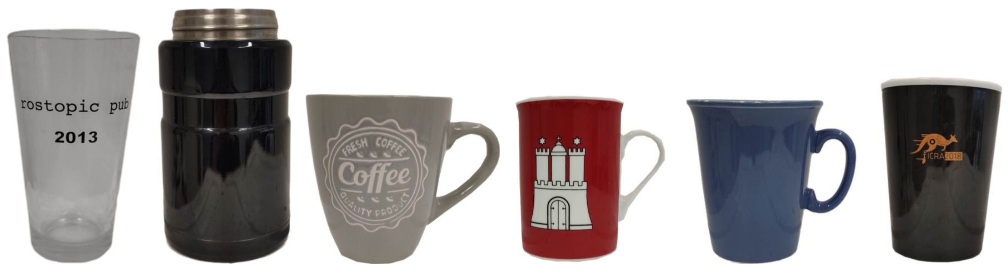


Figure 6.9: The target containers used in the audio-based pouring. From left to right, the first three target containers are the target containers in our datasets: a glass, a Thermos (stainless steel cup), and a mug; the latter three target containers are the unseen containers used in robotic experiments: a red mug, a blue mug and a plastic cup.

source container. Finally, the refined model was employed as the feedback command, which terminated the pouring immediately once the desired length of the air column had reached. The average computation time of one feedback loop is 21 ms, of which nearly 20 ms are spent on processing the spectrograms (a desktop machine with a 20-core Intel i9-7900X CPU and two 1080Ti GPUs).

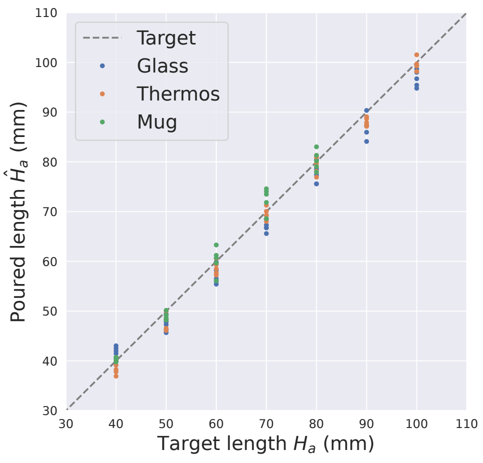
To verify the generalization ability and robustness to robotic pouring tasks, we design four groups of robotic experiments: evaluation on different target containers, different microphone positions, different initial liquid heights and different types of liquid.

6.5.1 Evaluation of Different Target Containers

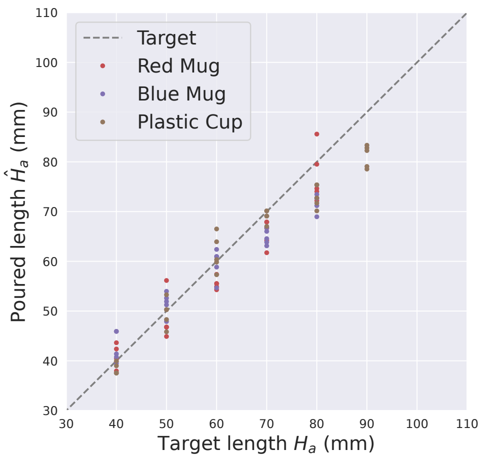
In this experiment, we kept the distance between the target containers and the microphone the same as in our original dataset. During the robotic pouring, we varied the target length of the air column between [40 mm, 50 mm, 60 mm, 70 mm, 80 mm] for three existing target containers in our dataset and three unseen target containers in Figure 6.9. The height of the red mug, blue mug and plastic cup respectively are 97 mm, 94 mm and 103 mm. Owing to the different height of each target container, we also tested 90 mm and 100 mm targets for three target containers in our dataset, and a 90 mm target for the plastic cup. The water was poured for five times to each considered height of each target container.

Quantitative results in Figure 6.10(a) indicate that our audio-based perception system can exploit the useful audio features even though there are huge differences between the human setup (used in our dataset) and the robot setup (used in our robotic evaluations), such as the pouring trajectories and the degree of noise of sthe pouring environment. From Figure 6.10(a) and Figure 6.10(b), we can see that when the target length of the air column is shorter (*i.e.*the liquid height is higher), the estimated length of the air column gradually become more accurate. It suggests that our model works well in height estimation, and the model will be more sensitive when the target length H_a is relatively high. Since there is more likely to spill out with higher liquid height, this property provides our AP-Net* the extra ability to prevent the manipulator from spilling out the liquid.

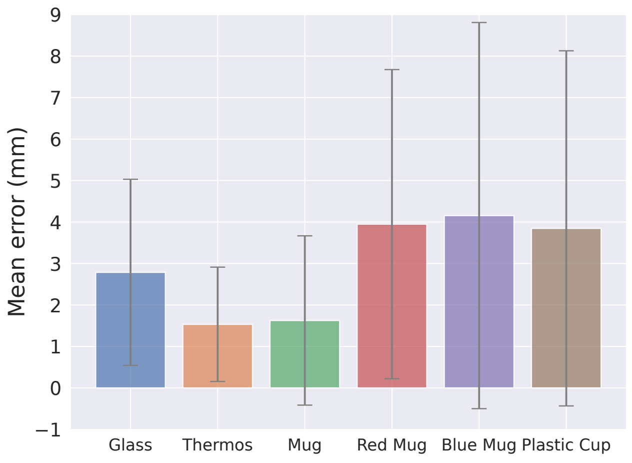
For numerical results, Figure 6.10(c) quantifies that the absolute mean errors and the



(a)



(b)



(c)

Figure 6.10: Controlling results of pouring water to a desired length of air column by our trained LSTM model in robotic experiments. (a) Length estimation of target containers from our dataset. (b) Length estimation of unseen target containers. (c) Comparison on the absolute mean error and standard deviation of different target containers among all desired heights.

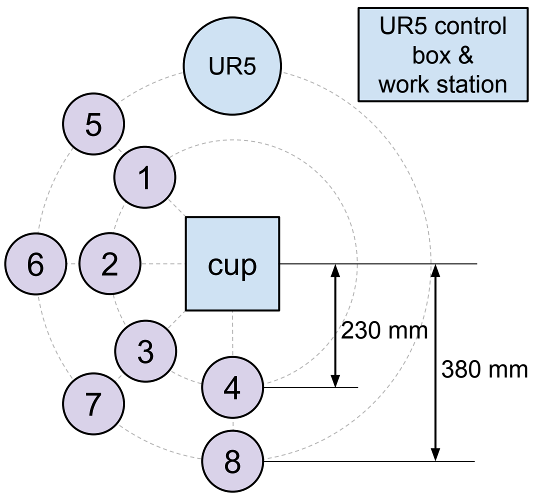
standard deviations of the liquid height are both below 3 mm among all target containers used during training and below 4.5 mm among the unseen target containers. Additionally, we converted the height error of each cup to weight error in this experiment as shown in Table 6.1. In previous work on robotic pouring, Schenck *et al*. [101] reported a mean error of 38 ml and Do *et al*. [27] achieved a mean volume error 22.53 ml over three different target containers. Compared to their results, the robotic pouring with our audio-based perception system can achieve higher precision.

6.5.2 Evaluation of Varying Microphone Positions

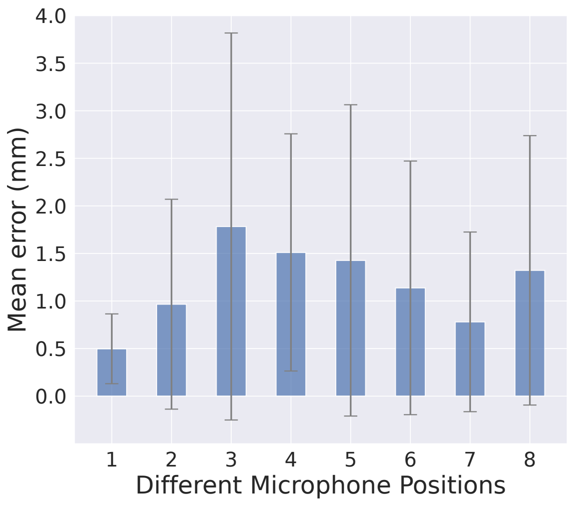
We compared the performance of our method on eight different positions of the bottom center of the microphone as shown in Figure 6.11(a). Position 1 is the position that we

Table 6.1: Absolute mean weight and standard deviations errors converted from the length of the air column error of the robot experiment

Glass	Thermos	Mug	Red Mug	Blue Mug	Plastic Cup
9.54 ± 7.81 ml	9.91 ± 8.48 ml	13.79 ± 11.04 ml	7.92 ± 7.14 ml	6.42 ± 6.31 ml	10.72 ± 8.70 ml



(a)



(b)

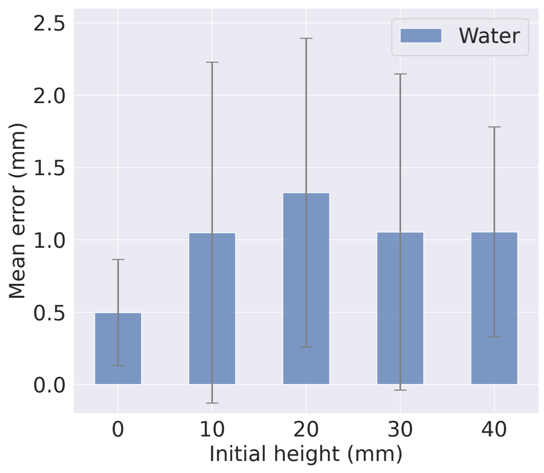
Figure 6.11: Evaluation results of eight microphone positions. (a) Schematic diagram of eight microphone positions relative to the target containers, the UR5 robot position, and the control box of UR5 robot (which is the major noise source). (b) Evaluation results of eight microphone positions shown in (a).

placed the microphone in our dataset. The distances between the target containers and the bottom center of the microphone at positions 1, 2, 3, 4 and at positions 5, 6, 7, 8 are 230 mm and 380 mm respectively. We used the mug and the desired length of air column at 40 mm, as in the later two groups of evaluations. Then we poured water at each microphone position for five times. Since the robot poured from the right side of the target container, all tested microphone positions were on the left side to avoid a collision between the microphone and the robot.

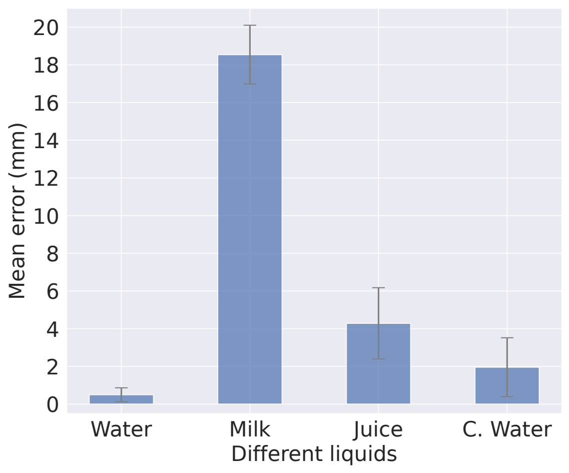
As shown in Figure 6.11(b), the estimation results indicate that our method generalizes well to the different positions of microphone due to the highly consistent of mean height error among all tested positions. In particular, the distance between the noisy robot control box and the microphone does not have a significant influence on the performances, which further suggests the robustness of AP-Net* against noise.

6.5.3 Evaluation of Varying Initial Liquid Height

In this experiment, we selected the initial liquid height of target containers from the set: [10 mm, 20 mm, 30 mm, 40 mm], and poured liquid from each initial length for five times. Other experiment settings are the same as the evaluation of different microphone positions. The results of the absolute mean error and standard deviation are listed in



(a)



(b)

Figure 6.12: Evaluation results of (a) varying initial height and (b) different liquids. Where C. Water represents carbonated water.

Figure 6.12(a). This demonstrates that our AP-Net* is stable among these different experiment settings, which indicates that it generalizes well to the different initial liquid height in robotic pouring.

6.5.4 Evaluation of Different Types of Liquid

To analyze the influence of liquid type on our perception method, we also conducted pouring experiments with pure water (which is used in other experiments), carbonated water, 1.8% fat milk and orange juice with pulp. These types of liquid have different physical properties regarding density, thickness, and viscosity. We used the same experiment setting as in evaluations of different microphone positions and poured each type of liquid for five times. Figure 6.12(b) demonstrates that our model is able to generalize to common household liquids like carbonated water and orange juice. However, it has failed to work well on milk as its higher viscosity makes the sound weaker and induces difficulties to record and analysis. As the milk and juice are with higher viscosity than pure and carbonated water, it turns out that our the generalization performances of our AP-Net* are negatively correlated to the viscosity of the liquid.

Through these experiments above on the generalization performances of AP-Net*, we verify that our method is able to handle different target containers, microphone positions, initial heights and some liquid types in robotic pouring. And we also see a restriction of our method as it cannot be applied to liquid with high viscosity.

6.6 Discussion and Summary

This chapter presents a real-time perception system used for estimating the liquid height in robotic pouring. We offer a multimodal pouring dataset including audio-frequency recordings, liquid real-time weight, force and torque feedback, video and motion trajectories. With this dataset, we develop a robust audio-based perception model named AP-Net*. AP-Net* takes an audio sequence as input and produces the estimation of the length of the air column of the target containers to represent the height of liquid. Especially, using the length of the air column of the target as label can deal with the risk to overfill without any prior knowledge of the target container. Model evaluations on our dataset suggest that the acoustic sequences provide rich information of liquid height and achieve high precision on our tests. Various robotic experiments with a focus on generalization and precision performances demonstrate that our proposed AP-Net* generalizes well to different experiment settings while the precision can still be guaranteed. The dataset and associated software are public and are available[1].

In future work, to facilitate robot behavior in human-robot interaction scenarios, we plan to extend our approach to more noisy environments including human voice and make the distance between the source and target container changeable. Making use of the force, torque, or visual data from our multimodal dataset and designing a novel neural network based on multiple modalities would be a promising approach to improve the robustness of height estimation in robotic pouring.

[1]`https://lianghongzhuo.github.io/AudioPouring`

Chapter 7

Robust Robotic Pouring using Audition and Haptics

Robust and accurate estimation of liquid height lies as an essential part of pouring tasks for service robots. Since AP-Net proposed in Chapter 6 cannot work well in a noisy environment, therefore, this chapter proposes a multimodal pouring network (MP-Net) that is able to robustly predict liquid height by conditioning on both audition and haptics input. MP-Net is trained on a self-collected multimodal pouring dataset. This dataset contains 300 robotic pouring recordings with audio and force/torque measurements for three types of target containers. We also augment the audio data by inserting robot noise. We evaluated MP-Net on our collected dataset and a wide variety of robot experiments. Both network training results and robot experiments demonstrate that MP-Net is robust against noise and adapts to the task and environment. Moreover, we further combine the predicted height and force data to estimate the shape of the target container.

7.1 Introduction

Pouring a specific amount of liquid into a container is an important manipulation skill for service robots in applications such as bartending or housekeeping and also in industrial environments. Compared to humans who can effortlessly pour the liquid into a container without spilling, predicting the liquid height in the target container is still challenging for a robot. For the robust and accurate perception of robot pouring, recent approaches mainly rely on vision [102, 83, 100]. They take the RGB image as input, then infer the liquid height, liquid amount, even pouring trajectories. However, vision fails under occlusions or in the dark. Using audio [20, 123, 71] is a second option for robot pouring perception, but the performance of this method will degrade in a noisy environment. Besides, if the robot is equipped with force/torque sensing, the amount of liquid poured out from the source container can be measured directly [50], but the liquid height in a target container can only be predicted if the initial fill level and the shape of the container are known. These drawbacks in existing perception methods suggest that to estimate the liquid height precisely, it is necessary to find a robust approach that still works in conditions where one modality becomes incomplete or deteriorated.

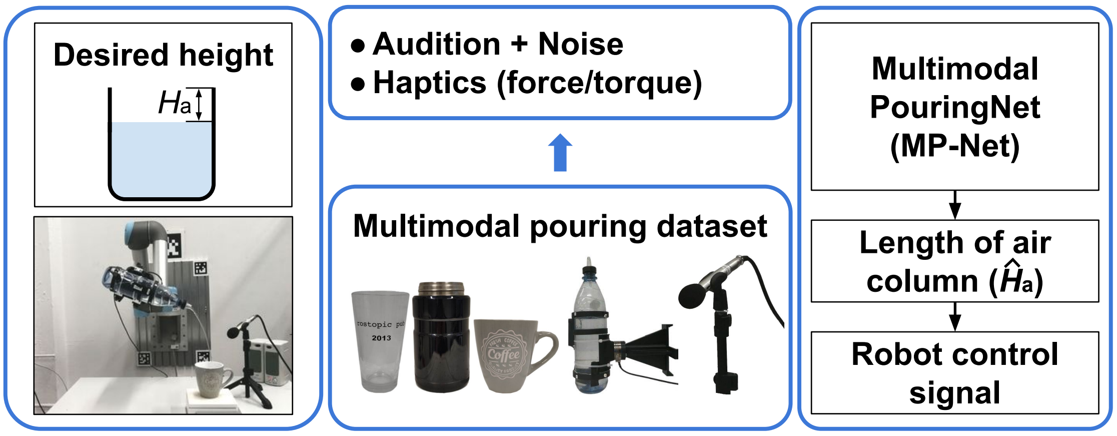

Desired height
Ha
• Audition + Noise
• Haptics (force/torque)
Multimodal pouring dataset
Multimodal PouringNet (MP-Net)
Length of air column (Ĥa)
Robot control signal

Figure 7.1: The multimodal pouring pipeline. (Left) The goal of our pouring task is to pour the desired amount of liquid into a target container. The generated sounds depend on the height of the remaining air column (H_a) in the container. (Center) We collected a multimodal robotic pouring dataset, which contains 300 pouring sequences with audio and force/torque sensor data for training our network. A scale recorded the weight of the poured liquid. We then calculated H_a to serve as ground truth. (Right) By giving the audio and haptic as input, our Multimodal PouringNet (MP-Net) predicts $\hat{H}_a$ as output. This output is then used to control the robotic pouring.

Therefore, we propose to tackle the issues of robust robotic pouring by a multimodal perception method. This multimodal system should be robust in a wild environment, for example, against different levels of noise, types of noise, or light conditions. The perception system should be able to generalize to different target containers, liquid types, and initial liquid height in the target containers.

In this chapter, we develop a novel multimodal neural network called MP-Net that takes preprocessed audio spectrograms and force/torque data as the input and estimates the length of the air column in the target container. This idea is inspired by human experiences, as humans can infer whether the target container is almost full from the pouring sound and use their proprioceptive haptic feedback to estimate how much liquid is poured out from the source container [52, 121]. Figure 7.1 illustrates the proposed perception pipeline.

The main structure of MP-Net is a RNN that implicitly integrates the prior knowledge that the liquid level rises monotonically during the pouring. A multimodal pouring dataset with audio and haptics was built up to train this model. To further improve the adaptability to the noisy environment of MP-Net, we augmented the audio data by adding different noise levels of robot noise. The benefits of force/torque data and audio data augmentation are verified in both our dataset and a wide range of robotic experiments, respectively.

To sum up, our main contributions are:

1. We propose a multimodal neural network MP-Net to estimate the liquid height robustly and to enhance the precision and generalization ability of robotic pouring. Not only the haptic input but also the audio data augmentation facilitate to produce a robust model that can generalize broadly.

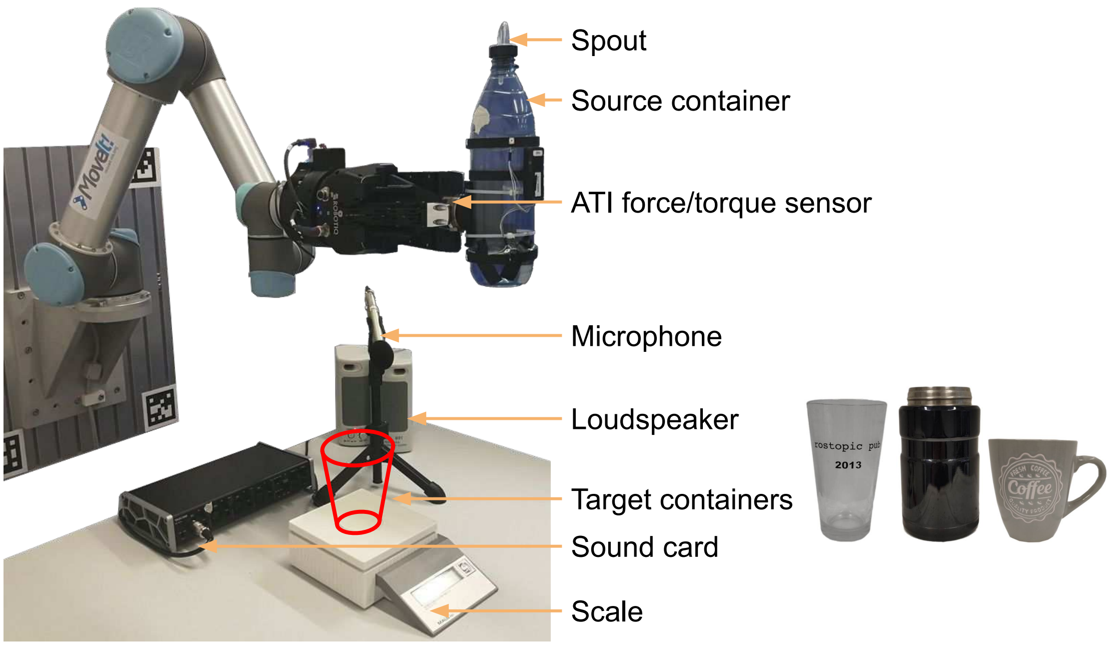


Figure 7.2: Pouring setup used to collect our multimodal dataset and to perform the robot experiment. The three target containers used to collect the dataset are shown on the right. We put these containers on a calibrated scale to get the real-time label of the liquid height. A force/torque sensor is mounted on the source container to collect force and torque information while pouring, and a microphone was set on the table to collect the pouring audio. Loudspeakers were used to create ambient noise for the robot experiments.

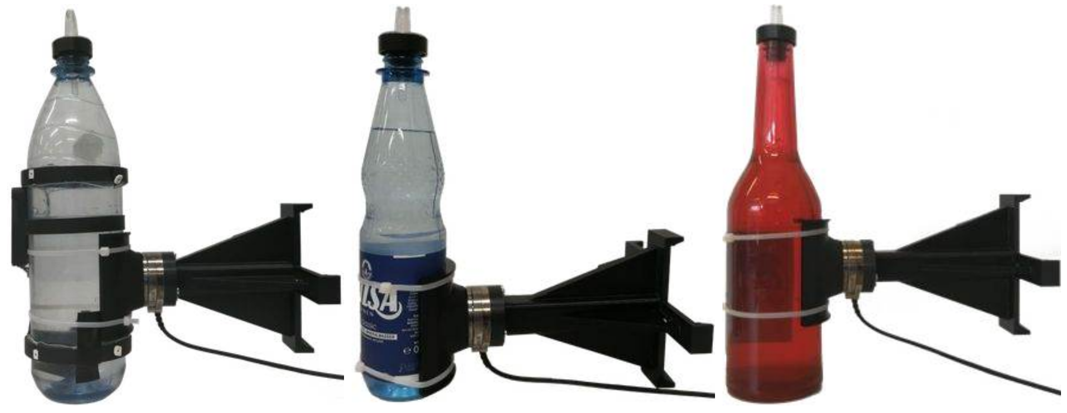
2. System assessment and comparison across eight robotic experiments, *e.g.*, pouring water into different containers, in various noisy environments with different noise levels and noise sources, revealed that high accuracy could be achieved by our proposed method.

3. The shape of the target container can be reconstructed by combining the force data and real-time height prediction.

7.2 Multimodal Pouring Dataset

7.2.1 Dataset Collection Setup

To create the multimodal dataset for the height estimation task, we designed a robot pouring setup, as shown in Figure 7.2. It consists of a UR5 robot arm with a Robotiq 3-finger adaptive robot gripper, and a custom 3D-printed bottle holder with an embedded ATI force/torque sensor. A standard plastic bottle (S1 in Figure 7.3) is used as the source container, with an optional bottle spout to limit liquid flow. The target container is placed on a MAULlogic digital scale which measures the combined weight of the container and the liquid inside. Audio is recorded with a microphone in an environment with only UR5 robot ego-noise. The dataset setup contains three different target containers with three



(a) S1 (b) S2 (c) S3

Figure 7.3: S1, S2, S3 represent three different source containers where S1 is the container used to collect dataset, S2, and S3 are novel source containers.

different materials and heights. The properties of these containers are shown in Table 7.1 with ID 1-3.

Unlike our previous work that used human pouring sequences as training data [71], we use the robot to perform the pouring task. The main reason is that the haptic data coming from human pouring is massively different from the robot pouring data, which makes it hard to transform human pouring to robotic pouring.

We set a fixed pouring trajectory and place the mouth center of the source container 310 mm above the scale plane. In this manner, we collected 100 trials each for three different target containers. For each trial, we recorded both audio and force/torque data starting just before the scale reading begins to change, and stopping after the reading of the scale becomes stable again. The lengths of one pouring recording varied from 24-40 seconds according to the capacity of the different target containers.

7.2.2 Data Analysis

Audio-frequency data

To better extract audio information from the microphone, we resampled all audio data from 44.1 kHz to 16 kHz and computed spectrograms with a window length of 32 ms and 50% overlap without zero-padding. For network training, we randomly chose 4 seconds of audio clips from one complete pouring audio sequence. Using fixed length training samples allows for batch processing for better GPU performance, and random starts correspond to different initial liquid amounts, so that the network can generalize to partially filled containers.

Force/torque data

The force/torque sensor we use is ATI Mini45 (500 Hz). As the weight of the source container is far below the maximum payload of this sensor (580 N in Fx/Fy), the raw

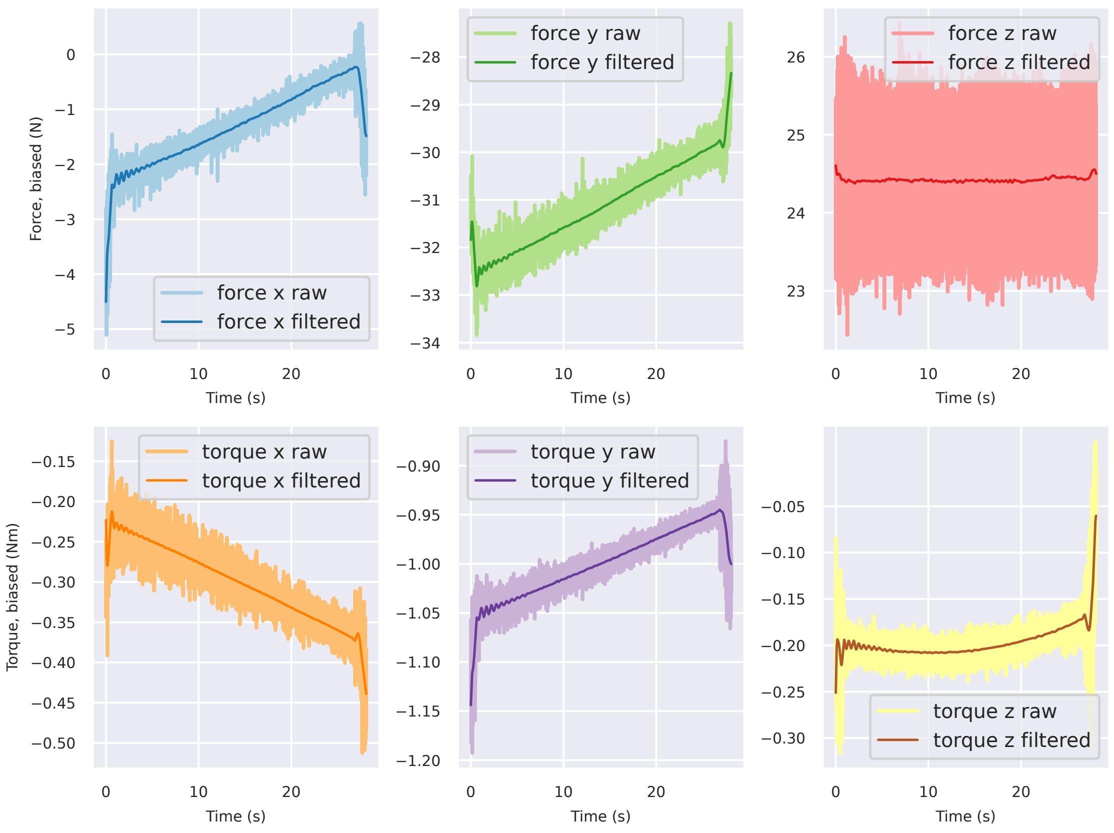


Figure 7.4: A sample of force/torque data (N/Nm) in our pouring dataset. In each sub-figure, we show the raw sensor data (light line) and the filtered data using a Butterworth low pass filter (dark line). See the text for details.

data is rather noisy but becomes usable after simple low-pass filtering. See Figure 7.4 for typical raw force/torque data (in N/Nm) during a sample pouring sequence. In each sub-figure, we plot the noisy raw signal from the sensor shown as a light color together with a low-pass filtered signal shown as a dark color. The change of the bottle weight during pouring can clearly be seen in the force/torque readings for the x- and y-axis, while the z-axis measurements show only little change. This is due to the specific orientation of the sensor in our 3D-printed holder, dictated by the cable routing. While the sensor z-axis is almost horizontal (orthogonal to gravity), the x- and y-axes of the sensor are pointing diagonally upwards, so that the raw sensor readings actually increase when pouring from the source container. The pouring motion from the robot slightly rotates the bottle around the z-axis.

Scale data

To get the ground truth of the pouring perception problem, we used a MAULlogic digital scale to measure the weight of the target container (accuracy ± 2 g). Because the publishing frequency of the scale is only 1 Hz, we deployed a linear interpolation (Figure 7.5(a)) to get real-time scale readings. To convert the measured weights into the heights H_a of the air column needed for network training, we sampled 10-15 random

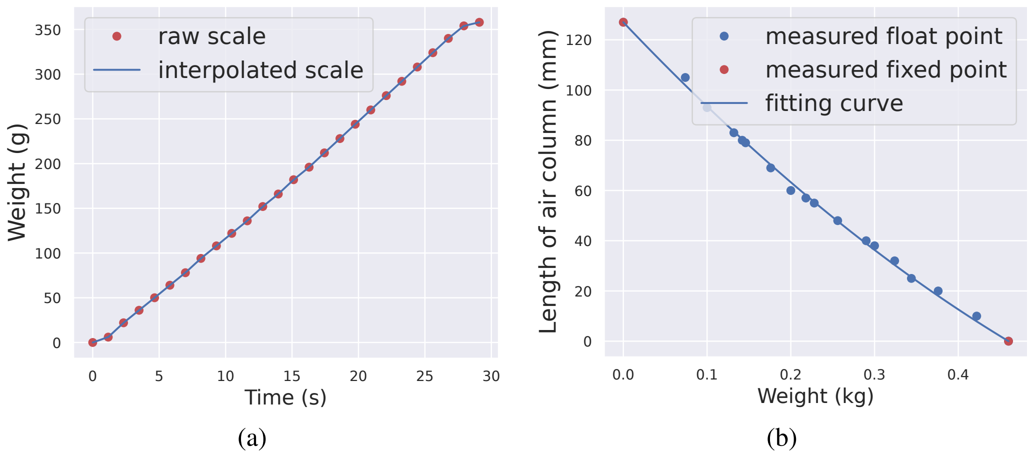


Figure 7.5: (a) Sample scale readings of a pouring sequence in our dataset from bottle ID 1. Each red dot is one weight measurement from the scale. The blue line is the interpolated curve. (b) Converting liquid weights (kg) into length of the air column (mm) for bottle ID 1. The dotted points are the manually measured data, where the red points indicate the empty and completely filled container. The blue line is the fitting curve that calculates from the scale reading to the length of the air column.

amounts of liquid for each target container, up to full container capacity, and manually measured the length of the air column. A polynomial curve fitting is utilized using this data to calculate the approximate air column length for different amounts of liquid and corresponding measured weights (Figure 7.5(b)).

7.2.3 Audio Data Augmentation

Considering that the variability of the audio data is significant enough, models trained on it will better generalize in different audio conditions. Therefore, we augmented the audio data in our dataset by adding noise. We recorded an ego noise from a humanoid PR2 robot, which is commonly used for research in household environments. In detail, we employed the signal-to-noise ratio in decibels ($\mathrm{SNR_{dB}}$) as our reference on how much noise was added.

$$\mathrm{SNR_{dB}} = 10\log\left(\frac{A_{\mathrm{signal}}}{\alpha_{\mathrm{nf}} \times A_{\mathrm{noise}}}\right)^2 \tag{7.1}$$

$$\alpha_{\mathrm{nf}} = \frac{A_{\mathrm{signal}}}{A_{\mathrm{noise}}} \times 10^{\frac{\mathrm{SNR_{dB}}}{-20}} \tag{7.2}$$

$$\mathrm{audio} = \mathrm{signal} + \mathrm{noise} \times \alpha_{\mathrm{nf}} \tag{7.3}$$

Equations 7.1-7.3 illustrate how noise was added, where A_{signal} and A_{noise} are the Root Mean Square (RMS) amplitude for the recorded pouring audio and noise, respectively. α_{nf} is the ratio of the input noise.

We mixed the original audio sequences with noise of different levels ranging from -20 to $+20$ $\mathrm{SNR_{dB}}$ with a step size of $5\,\mathrm{dB}$. The number of generated audio clips,

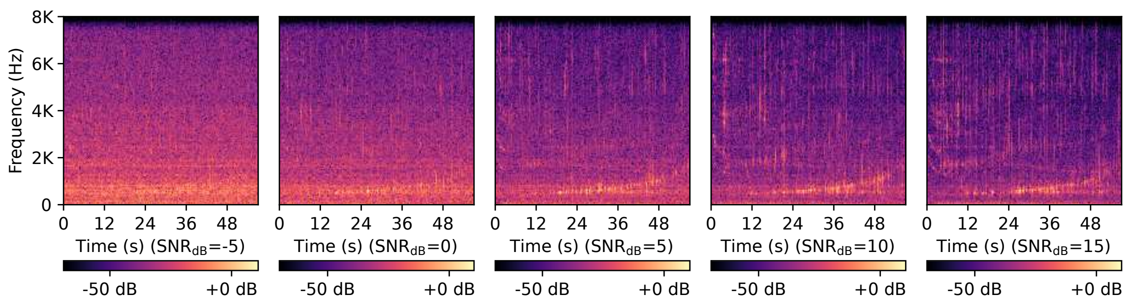


Figure 7.6: Examples of audio spectrograms that add different noise levels of the same audio signal in the dataset. While the $\mathrm{SNR_{dB}}$ is high (*e.g.*, $\mathrm{SNR_{dB}} = 15$, right), which indicates the audio signal is rather clear, we can clearly see rising frequency curves between 256 Hz-2048 Hz. But when the $\mathrm{SNR_{dB}}$ is low (*e.g.*, $\mathrm{SNR_{dB}} = -5$, left), there is no meaningful structure of the resonance frequency.

of 4 seconds length each, is proportional to the length of a pouring trial. Figure 7.6 shows examples of mixed audio spectrograms for different $\mathrm{SNR_{dB}}$. When $\mathrm{SNR_{dB}} > 0$, one high-energy and rising curve between 256 Hz and 2048 Hz is clearly visible during pouring. This curve represents the resonance frequency of the air. But when $\mathrm{SNR_{dB}} \leq 0$, the spectrogram has little structure of the resonance frequency, which indicates this audio sequence contains more noise than signal.

7.3 Multimodal Pouring Network

Our goal is to design a robust network architecture to acquire the liquid filling height by making sense of audition and haptics together. Audio vibration results in the changes in the resonance frequency of the air in the target container. Force/torque changes result from the liquid poured out of the source container, which yields additional guidance about the liquid in the target container, especially when the audio signal is deteriorated.

The pouring process is a typical sequential problem and the predicted length has a temporal relationship. Intuitively, we choose a RNN [16] as our model architecture. The multimodal network architecture MP-Net is shown in Figure 7.7. In detail, we utilize the LSTM unit to process the multimodal inputs. We mix audio clips with noise using Equation (7.3) and transform it into a $257 \times n$ matrix using STFT. The haptic data is processed to a $6 \times 8 \times n$ matrix (6 F/T sensor channels, and 8 F/T samples arrive during each 16 msec STFT interval) and then concatenated with the audio data to a $305 \times n$ matrix for input. Here, n is the number of the time slice, and we get $n = 251$ when using audio clips of 4 seconds duration from our dataset. Then each time slice of the fusion data is progressively fed into the encoder module (2-layer LSTM unit) to a layer of 56 recurrent features A_h.

Besides, it is crucial to find a well-suited ground truth in supervised learning. Due to the fact that when the length of the air column gets shorter in an organ pipe, the air vibrates faster and the resonance frequency of the air increases [38, 122], it is more indicative to choose the length of the air column H_a as the ground truth of our model instead of the liquid height. Thus, the height predictor (a 2-layer multilayer perceptron)

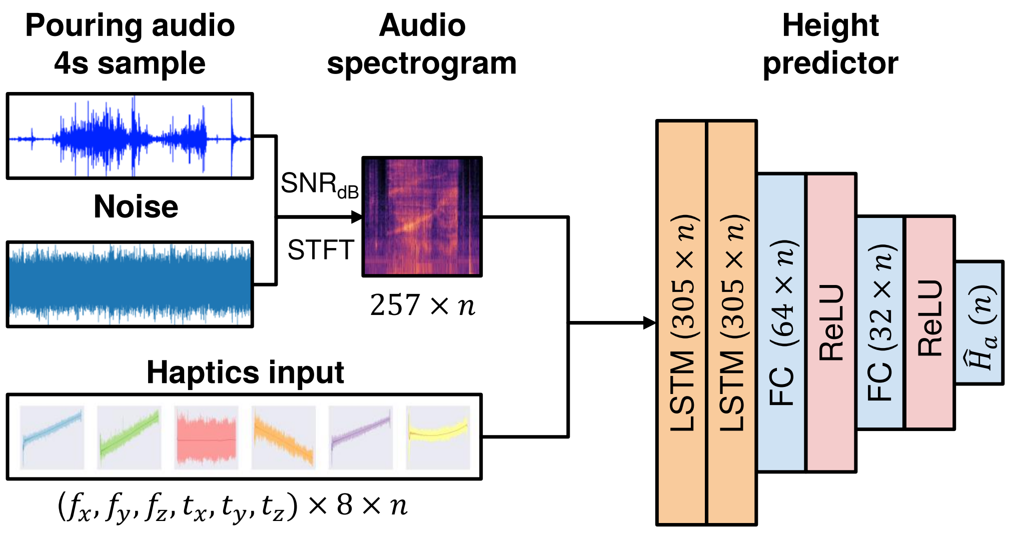


Figure 7.7: MP-Net architecture. The raw audio data is transformed into a spectrogram with 257 descriptors. Then the encoder module (a recurrent neural unit) is progressively fed each time slice of the audio-frequency spectrogram and the corresponding haptic data. Finally, the height predictor module produces the 1D length of the air column of the target containers. The blue rectangle denotes a FC layer followed by a batch normalization layer and a rectified linear unit. Here, $n = 251$ for a 4 s pouring sample.

in MP-Net takes the recurrent vector A_h as input and performs a regression of the length of the air column H_a. The height predictor is supervised with a Mean Squared Error (MSE) loss $\mathcal{L}_{height}$

$$\mathcal{L}_{height} = \|\hat{H}_a - H_a\|^2. \tag{7.4}$$

In addition, an auxiliary $\mathcal{L}_{mono}$ is introduced to enforce the decrease of the estimated length of the air column over time t

$$\mathcal{L}_{mono} = \sum_t \left[\max(0, (\hat{H}_{a_{t+1}} - \hat{H}_{a_t}))\right]. \tag{7.5}$$

Overall loss

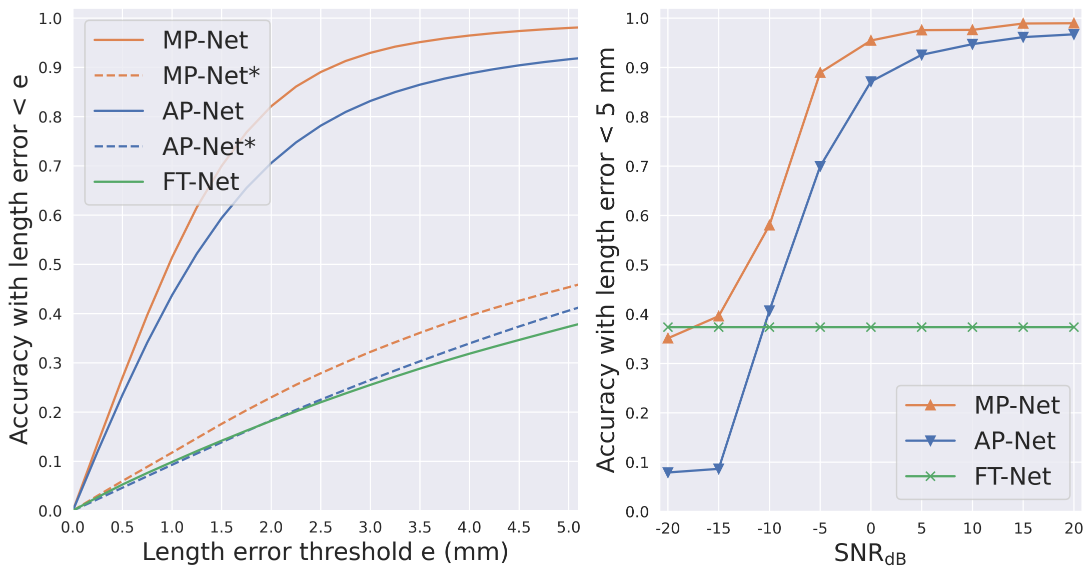
Combining with $\mathcal{L}_{height}$ and $\mathcal{L}_{mono}$, the complete training objective for MP-Net is defined by $\mathcal{L}_{mp}$

$$\mathcal{L}_{mp}(\theta) = \mathcal{L}_{height} + \alpha \cdot \mathcal{L}_{mono}, \tag{7.6}$$

where α is a hyperparameter for balancing these two loss functions. In our implementation, we set it to 0.01 for the best performance via some preliminary experiments.

7.4 MP-Net Evaluation

We examined our proposed multimodal network MP-Net against the following baseline methods:



(a) (b)

Figure 7.8: Network evaluation results of MP-Net and four baselines. (a) The fraction of the input sequence whose length prediction error is below a threshold between five models. (b) The fraction of the input sequence whose length prediction error is below 5 mm between MP-Net, AP-Net, FT-Net. MP-Net and AP-Net trained and tested on nine datasets, where the only difference between each dataset is the noise level of audio data.

1. MP-Net*: the same network architecture as MP-Net but trained without adding noise to the audio data.
2. AP-Net: only audio branch of MP-Net and training on augmented audio data with noise.
3. AP-Net*: same network architecture as AP-Net without audio augmentation [71].
4. FT-Net: only force/torque branch of MP-Net that only takes force/torque data as input.

All the models mentioned above were trained on our whole dataset which contains pouring samples collected from three different target containers.

First, we evaluated MP-Net and four baselines using the fraction of the input sequences whose length prediction error $|\hat{H}_a - H_a|$ is below a threshold e. We trained MP-Net and AP-Net on noisy audio, with noise levels from the $\mathrm{SNR_{dB}}$ set [0, 5, 10, 15, 20]. Except for FT-Net, MP-Net and the other three baseline models were then tested on audio data with $\mathrm{SNR_{dB}} = 5$ noise level. Figure 7.8(a) shows that MP-Net has a stable advantage of about 8% over AP-Net over the full range of tested length error thresholds, which indicates that the force data provide a positive gain to the training result. The performances of MP-Net* and AP-Net* are both poor compared to MP-Net and AP-Net, respectively, which indicate that augmenting audio data also contributes a lot to the networks. FT-Net was not trained on audio and has learned to predict the air column length averaged over all containers.

To compare the robustness of MP-Net, AP-Net, FT-Net with regard to audio noise, we evaluate the percentage of input sequences whose length prediction error stays below 5 mm, when both the training and the testing data were augmented with the same noise levels. Each model was trained and tested on audio data augmented with 9 different $\mathrm{SNR_{dB}}$ levels from the set [-20, -15, -10, -5, 0, 5, 10, 15, 20] respectively. Figure 7.8(b) illustrates that the accuracy of MP-Net and AP-Net increases with $\mathrm{SNR_{dB}}$. The accuracy curve of FT-Net is straight because force/torque data is independent of audio noise. We can see that MP-Net has at least 27% higher accuracy than AP-Net when $\mathrm{SNR_{dB}} \leq -15$, and has 8% higher accuracy when $\mathrm{SNR_{dB}} = 0$. Note that the performance of AP-Net degrades quickly for $\mathrm{SNR_{dB}} \leq -15$, which indicates that the noise interferes with the audio and the network cannot gain useful information from it anymore. Nevertheless, with the help of force/torque data, MP-Net can still perform comparably to FT-Net in such a noisy situation. This phenomenon proves that MP-Net has learned to predict liquid height by conditioning on both audio and haptic input.

7.5 Robotic Experiments

To verify and compare the reliability and robustness of the proposed MP-Net in robotic pouring tasks, we carried out eight evaluation experiments on different target containers, different pouring heights, different source containers, different noise levels, varying positions of the noise source, different initial liquid heights, different types of liquid, and different noise sources.

All experiments used both MP-Net and AP-Net to compare their real-world performance, except for the different source containers experiments. To adapt to different noisy environments, MP-Net and AP-Net were trained on noisy audio data with noise levels $\mathrm{SNR_{dB}}$ from 0-20. The experiment setup was the same as in our dataset collection setup and the initial water level of the source container was random. MP-Net took the latest 4 s sequence of audio and haptic data. While the sequence length of the input signal was less than 4 s, all the input signals were fed into the network. Once the estimated length of the air column was smaller than the desired one, the robot immediately stopped pouring. To compensate for the liquid poured out during the stopping motion, a corresponding correction was applied.

7.5.1 Evaluation of Different Target Containers

We used nine different target containers for robot experiments of which ID 1-3 are in the training dataset, ID 4-9 are unseen. Table 7.1 illustrate the properties of these nine containers. We kept the distance between the target containers and the microphone the same as in our original dataset. Then we set the desired length of the air column to 60 mm. We synthesized a $\mathrm{SNR_{dB}} = 5$ audio signal by playing the PR2 robot noise from a pair of loudspeakers located at positions 1&1 as shown in Figure 7.12(a). Here, 1&1 means that both loudspeakers were placed at position 1. We repeated the experiments five times for each container.

Table 7.1: The properties of nine target containers

ID	1	2	3	4	5	6	7	8	9
Figure									
Material	Glass	C*	Steel	C*	C*	Plastic	Plastic	Enamel	Glass
Height (mm)	127	99	150	97	94	103	115	78	135
Volume (ml)	460	428	766	310	298	416	408	382	358

*C represents for Ceramics.

Table 7.2: The pouring volume (ml) error at $\mathrm{SNR}_{\mathrm{dB}} = 5$, $H_a = 60\,\mathrm{mm}$

ID	1	2	3	4	5	6	7	8	9
MP-Net	15.8 ± 14.0	11.5 ± 8.1	6.3 ± 6.1	16.6 ± 19.0	8.2 ± 9.3	12.8 ± 11.5	23.7 ± 18.7	8.5 ± 9.6	13.9 ± 4.4
AP-Net	23.1 ± 8.2	7.5 ± 8.2	6.0 ± 4.7	25.3 ± 14.5	5.2 ± 2.3	14.3 ± 10	46.6 ± 70.6	41.6 ± 10.4	26.6 ± 45.4

Figure 7.9(a) displays that the absolute mean errors of the liquid height are below 8 mm and the standard deviations are below 4 mm for both MP-Net and AP-Net among the known target containers. Container 3 performs best for both networks due to the stainless steel material which makes the crispest sound. Furthermore, we converted the height error of each cup to a weight error, shown in Table 7.2. We can see that AP-Net performs well on known containers and even outperforms MP-Net a little on containers 2 and 3. Regarding the unseen containers, especially for container 7 and 9, MP-Net exhibits a stronger generalization ability than AP-Net in a noisy environment.

7.5.2 Evaluation of Different Pouring Heights

In this experiment, we placed the source container at four different heights from the set [310, 260, 210, 160] mm respectively. The height of the source container is the vertical distance from the mouth center of the source container to the scale plane. We carried out five robot experiments on each height using target container 2 when $\mathrm{SNR}_{\mathrm{dB}} = 5$. Figure 7.9(b) indicates that our algorithm performs well for the three higher heights but not at the lowest height. As the pouring height decreases, the volume of the pouring sound also decreases. With the source container at height 160 mm the pouring sound is

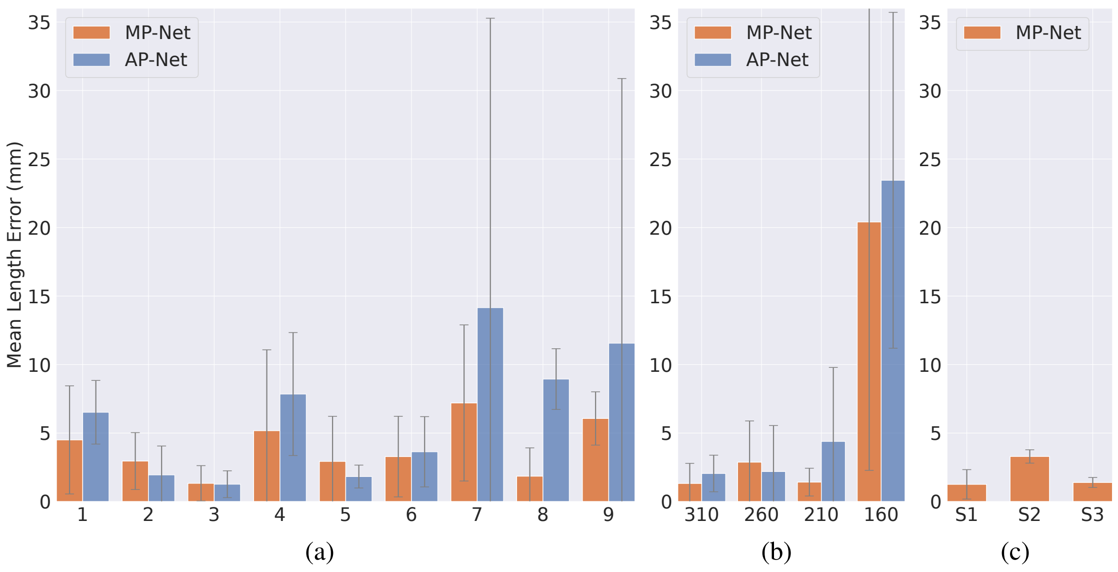


Figure 7.9: In the following robot experiments, the $\mathrm{SNR_{dB}}$ were set to 5 dB. Robot experiment result (a) on pouring water into different target containers ($H_a = 60\,\mathrm{mm}$, ID 1-3 are in the training dataset, ID 4-9 are novel), (b) on four different pouring heights (pouring height 1 was the height for dataset collection, $H_a = 40\,\mathrm{mm}$), (c) on one known source container S1 and two novel source containers S2, S3 ($H_a = 40\,\mathrm{mm}$).

too low, and haptics only is not sufficient for the network to perceive the height of the air column.

7.5.3 Evaluation of Different Source Containers

The source container influences the flow-rate and the initial force/torque data. Therefore, we tested two novel source containers S2 (44 g) and S3 (484 g), which are shown in the left side of Figure 7.2, to compare with the container S1 (64 g) used for network training and all other experiments. We kept the pouring height at 310 mm, and the other experimental setup was the same as in the evaluation of the different pouring height. Figure 7.9(c) suggests that different source containers hardly affect network performance.

7.5.4 Evaluation of Varying Noise Conditions

To further verify the performance of our MP-Net model in different noise conditions, we implemented a set of experiments using target container 2 under six different noise levels of [-5, 0, 5, 10, 15, 20] $\mathrm{SNR_{dB}}$. The loudspeakers were again located at positions 1&1. For each $\mathrm{SNR_{dB}}$ level, we tested five different target lengths of air column, namely [40, 50, 60, 70, 80] mm. We carried out five robot experiments on each audio and target length condition.

As visualized in Figure 7.11, MP-Net has a substantial advantage over AP-Net when $\mathrm{SNR_{dB}} = -5$, which further indicates the advantages of multimodal fusion. In this experiment, we also tested AP-Net* under different noise conditions. AP-Net* performed

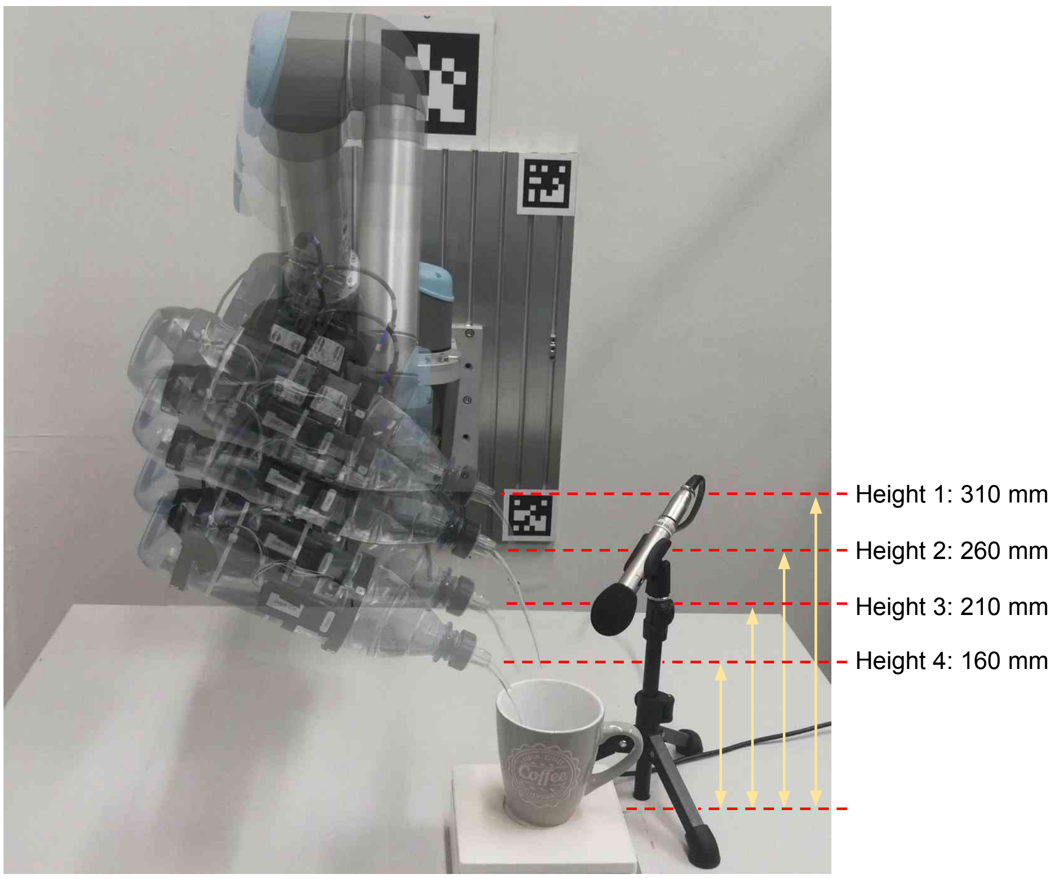


Figure 7.10: Visualization of different pouring heights.

well while $\mathrm{SNR_{dB}} \geq 10$, but when $\mathrm{SNR_{dB}} < 10$, the robot either stopped pouring immediately or overfilled the target containers. Therefore we did not list the experiment results of AP-Net* tested on audio with $\mathrm{SNR_{dB}} < 10$.

7.5.5 Evaluation of Varying Positions of Noise Source

To assess whether MP-Net is sensitive to the direction of the noise source, we set up six different position combinations [1&1, 2&2, 3&3, 4&4, 2&4, 1&3] of the two loudspeakers. The two loudspeakers played a synthetic $\mathrm{SNR_{dB}} = 5$ noise signal at each position. We used the target container 2 and a desired air column length of 40 mm. The other experimental setup was the same as in the evaluations of the different target containers. Then at each combined position of the two loudspeakers, we poured water five times. As shown in Figure 7.12(b), when the loudspeakers are at position 1&1, both models perform best as the loudspeakers are behind the microphone. MP-Net generalizes better than AP-Net to the different positions of the loudspeakers due to the lower consistent mean height errors among all tested positions.

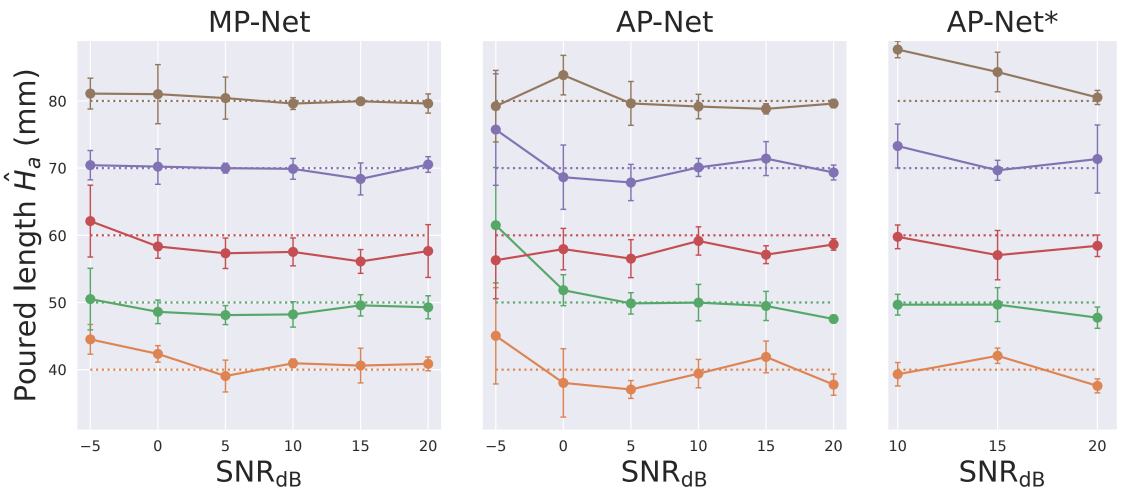
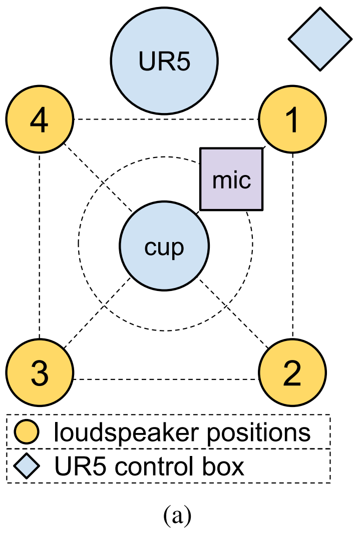


Figure 7.11: Robot experiment results of the performance of MP-Net, AP-Net, and AP-Net* in environments with different levels of noise (measured by $\mathrm{SNR}_{\mathrm{dB}}$). We evaluate with five different target liquid heights and the results are demonstrated in five different colors, respectively. The dashed lines represent the desired lengths of the air column, while the solid dots (with error bars) show the actual ones when the pouring terminates.



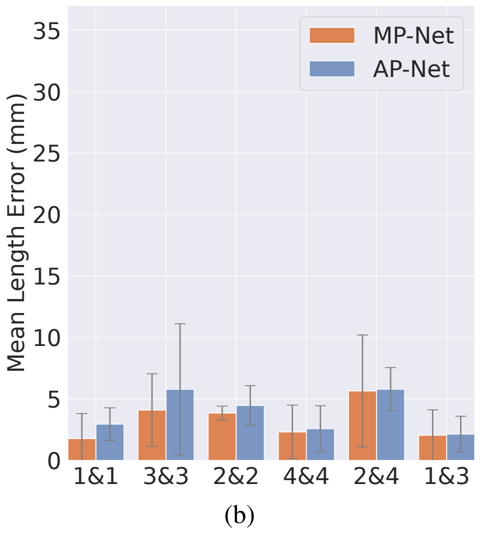


Figure 7.12: Evaluation of varying positions of noise source. (a) Schematic diagram of four different loudspeaker positions relative to the target containers, the UR5 robot position, and the control box of the UR5 robot (top view). (b) Evaluation results of six combinations of two loudspeakers positions shown in (a).

7.5.6 Evaluation of Varying Initial Liquid Height

In this experiment, we poured water into the target container 2, starting from five different initial liquid heights [0, 10, 20, 30, 40] mm. We tested five times from each initial level. We put two loudspeakers at 1&1 positions and kept the other test setups the same as in the evaluations of the varying direction of noise sources. The results in Fig-

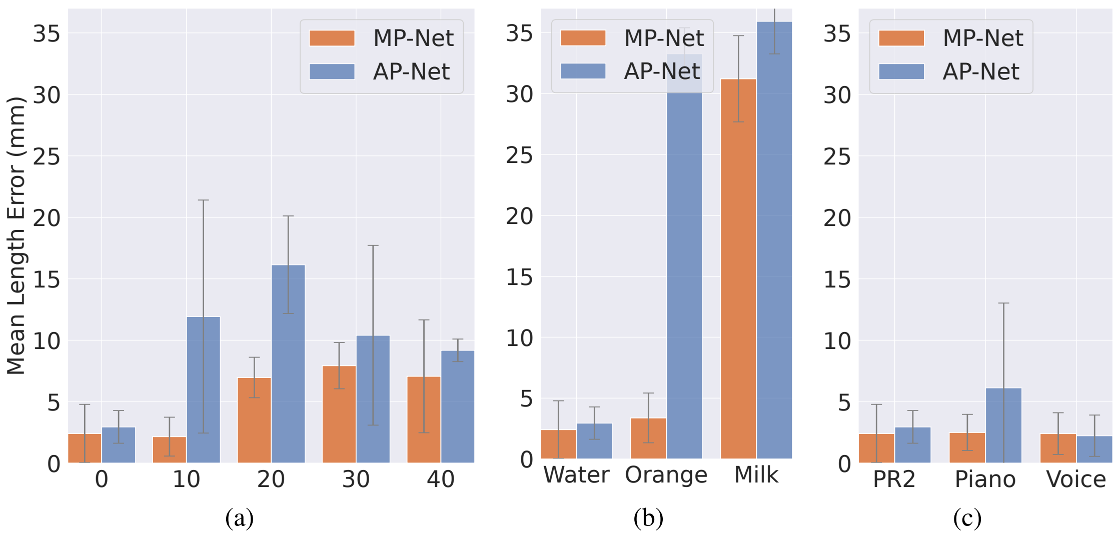


Figure 7.13: Evaluation results of (a) varying initial heights of the source container, (b) different types of liquids and (c) different types of noise sources. In these experiments, the $\mathrm{SNR}_{\mathrm{dB}}$ was set to 5 dB. The target height H_a was set to 40 mm.

ure 7.13(a) demonstrate that MP-Net is again more robust than AP-Net. Force and torque data yield a meaningful indication of how much water was poured out.

7.5.7 Evaluation of Different Types of Liquid

We conducted pouring experiments with different liquids: pure water, orange juice and 1.8% fat milk. These liquid have different physical properties regarding density and viscosity. We used the same experimental setting as in the evaluations of different microphone positions and poured each type of liquid for five times. As manifested in Figure 7.13(b), MP-Net can generalize to common household liquids like water and orange juice while AP-Net cannot handle the task of pouring orange juice well under $\mathrm{SNR}_{\mathrm{dB}} = 5$. However, similar to [71], due to the high viscosity of milk which makes the pouring sound very low, both models cannot generate correct height predictions.

7.5.8 Evaluation of Different Types of Noise Sources

We also assessed our model with three noise types: PR2 robot noise, human voices and a continuous piece of piano music. The human voice is represented by discrete sounds of a man counting numbers in English. We poured water under each type of noise five times. All experimental settings were the same as in the evaluations of different types of liquid. Figure 7.13(c) shows that MP-Net is not affected by different noise types, but the accuracy of AP-Net has a small fluctuation under a musical disturbance.

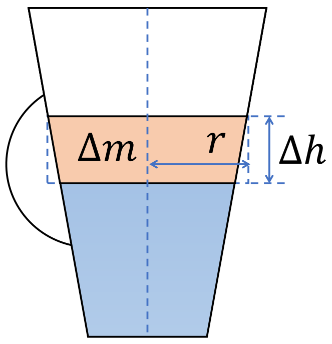


Figure 7.14: Schematic diagram of a symmetric container. In a specified time interval Δt, the change in mass Δm can be determined by the F/T sensor and the change in height Δh can be derived through MP-Net. Then the radius r at each height can be calculated to form an edge-profile of this container.

7.5.9 Shape Prediction of Target Containers

In this section, we applied MP-Net to predict the shape of symmetric target containers. In this case, the edge profile is sufficient to describe the shape of the containers, which is determined by the correlation between height and radius [55]. Figure 7.14 shows a volume profile filled with liquid of density ρ, where ΔV, Δm, Δh are the poured liquid volume, and the weight and liquid height differences during a time interval Δt respectively. Assuming that Δt is very small, then Δm can be calculated by approximating the shape of ΔV as a cylinder,

$$\begin{aligned} \Delta m &= \rho \Delta V \\ &= \rho \pi r^2 \Delta h \end{aligned} \tag{7.7}$$

We can determine Δm by the (f_x, f_y, f_z) values from the force/torque sensor and Δh through our neural network output $\hat{H}_a$. In the robot experiments, the frequency of Δm and $\hat{H}_a$ was 500 Hz and 12 Hz, respectively. To get a smooth and accurate estimation of the container shape, we used a quadratic function to fit the scatter points,

$$r(h) \approx \sqrt{\frac{\Delta m}{\Delta h} \frac{1}{\rho \pi}} \tag{7.8}$$

For target containers 1, 2, 4, 5, 6, 7, we conducted five trials of the experiment in which the robot pours into these target containers. We recorded the real-time estimation of $\hat{H}_a$ and force data into a rosbag. When the target container was filled to about 90% of its total height, we stopped the pouring and the recording. Using the data from these rosbags, we calculated the edge-profiles of the target containers. In Figure 7.15, the thick black curves are the ground truth profiles, while five colored curves around black curves depict the experimental results. The magenta area in the middle of each target container visualizes the mean error of the radius prediction. As expected, the mean radius estimation error is highest for an empty container, when our recursive network cannot yet rely

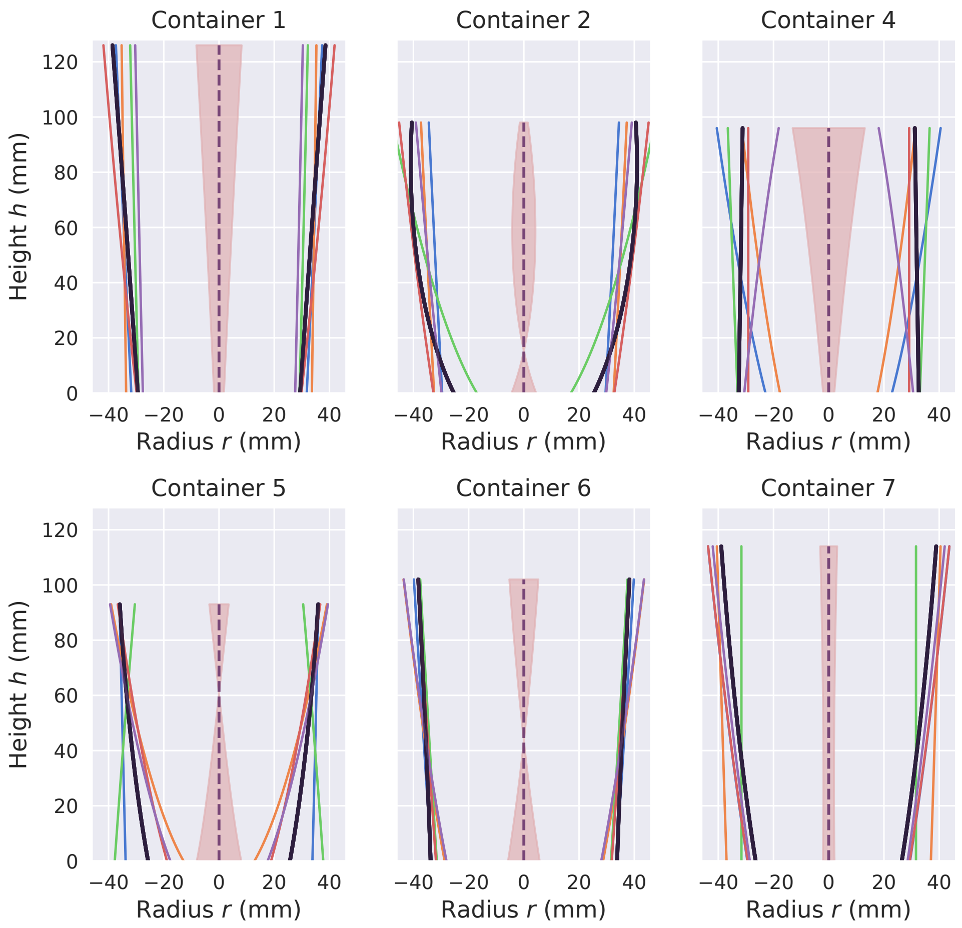


Figure 7.15: Prediction result of estimating the target container shape. The black curve is the ground truth, while the five different colored curves are the estimated target container shape in five trials. The mean error of the estimated container radius at different heights is plotted in the middle of each subplot (the shaded magenta area).

on its memory but stabilizes as the liquid level rises. Due to the restriction to quadratic functions, the reconstruction works best for containers with low edge curvature (such as containers 1, 7).

7.6 Discussion and Summary

In this chapter, we motivate the need for robust robotic pouring by combining audio and haptic information. We recorded a robot pouring dataset that includes 300 complete pouring sequences with audio and force/torque data. We propose a novel audio-haptic recurrent deep network (MP-Net) trained on this dataset that predicts liquid height in real-time.

The multimodal perception algorithm is systematically tested across four baselines and a wide range of robotic pouring experiments. The results substantiate that MP-Net is quite robust against noise and against changes in different tasks and varying environments. Finally, the multimodal nature of our network lets us reconstruct the shape of the target container. The dataset and associated software are public and are available[1].

One limitation of our approach is the poor generalization to liquids like milk or fruit juices, which would be considered quite similar to water by many humans, while the pouring noises are actually quite different. Training on different liquid types would improve network performance, but MP-Net will still fail in situations where the auditory signal is too weak. Another issue is our use of raw force/torque data as the network input, which changes significantly for different grasp types and pouring motions. This could be resolved by training on many grasps, or simply by feeding the preprocessed weight data into the network.

For future work, using audio and haptic information for dynamic control of robotic pouring would be an exciting research direction.

[1]`https://lianghongzhuo.github.io/MultimodalPouring`

Chapter 8

Conclusion and Future Work

This thesis studied two important robotic tasks in everyday life, grasping and pouring. The two robotic tasks can then be used to compose a complex service robot task, *i.e.*, serving a human user drinks.

In detail, the target defined in Section 1.2 can be concluded as five sub-goals which are discussed in Section 8.1. Section 8.2 presents the limitation of this thesis, and Section 8.3 lists the future work the author will do to enhance robotic perception and robotic skill learning.

8.1 Achievements

The drink-serving robot task can be divided into five sub-tasks:

- **Improved grasp pose generation method.** An improved grasp pose generation method is proposed. The proposed grasp pose generation improves the grasp generation to generate more grasp candidates in the graspable region of cluttered space. The versatile grasp candidates give the grasp evaluate network more chance to find better grasps. The robotic experiment also proves that our improved grasp candidates generation method can achieve an overall better grasp success rate in a fair comparison using the same grasp evaluation network.

- **Grasp evaluation network to improve grasp evaluation results.** With a handy yet powerful grasp generation algorithm. The grasp evaluation network PointNet-GPD is proposed to rank the grasp quality. The grasp evaluation network is based on PointNet, which takes pure point cloud as input and a local frame grasp representation. Robot experiments show that PointNetGPD can better evaluate grasp candidates even in a dense clutter with single-viewed point cloud input. A further experiment shows that a shape completion network in the grasp generation loop can also improve the grasp success rate.

- **Multifingered grasping based on RL and multimodal inputs.** The multifingered grasping is also studied. We train the whole multifingered grasp skill in simulation and conduct real robot experiments. The agent in the simulator takes tactile, torque, and joint proprioception as observation and is trained by the PPO algorithm. The action space is simplified with PCA dimension reduction to simplify the task. The real robot experiment shows that multimodal inputs can help learn multifingered grasping, and the agent trained in the simulation can successfully transfer to the real robot.

- **Audio-based network for robot audio pouring height estimation.** To pour liquid without the help of a vision sensor, audio perception is studied to make sense and retrieve real-time liquid height perception. With the help of a large human pouring dataset that contains multimodal information, we use the audio sensing and a LSTM network for liquid height perception. The robot experiment shows that audio can get a precise liquid height perception in a quiet environment.
- **Multimodal robot pouring height perception based on audio and force/torque sensing.** To further make the audio pouring perception robust to environment noise and other disturbance, a multimodal pouring height perception network is proposed that uses audio and force/torque. With the augmentation of the audio training set with noise and the help of force/torque data, the multimodal network can make liquid height perception even in a noisy environment.

The main takeaway of this thesis is that the author solved both the robotic grasping (Chapter 3, 4, 5) and the pouring perception task (Chapter 6, 7) using multimodal neural networks. This proves that multimodal information can be more robust to environmental uncertainty. In the end, by combining the above five achievements, a demo of the PR2 robot grasp bottle and pouring liquid is shown in Figure 8.1.

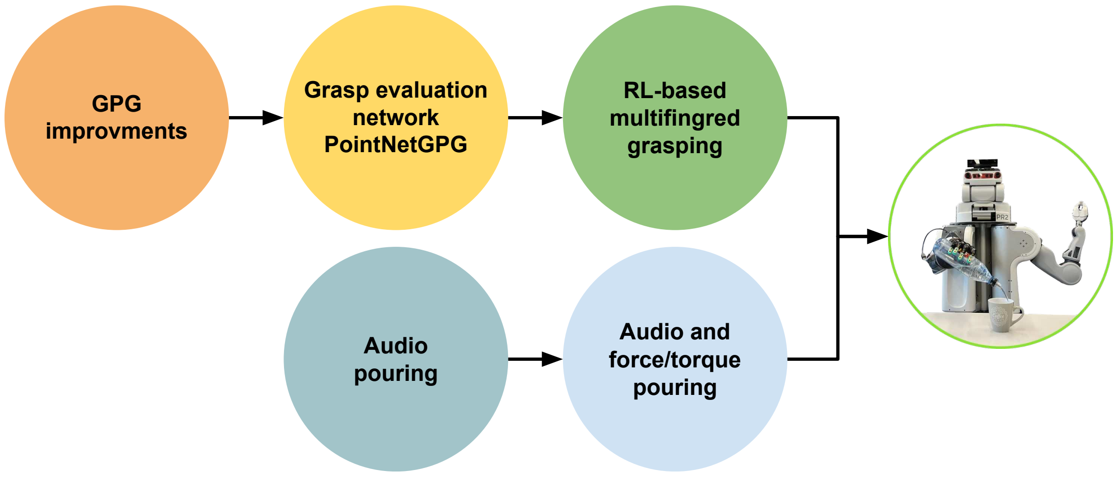

GPG improvments
Grasp evaluation network PointNetGPG
RL-based multifingred grasping
Audio pouring
Audio and force/torque pouring

Figure 8.1: Combining five achievements made by this thesis, the PR2 robot serves a drink to a user. The whole task includes grasping and pouring.

8.2 Limitations

Even though this thesis has accomplished many achievements, there is still space for improvement. These limitations will prevent a more autonomous drinking system not addressed in this thesis.

The semantic understanding of the environment is missing. *Semantics* is the concepts represented by the things in the real world. With semantic understanding, the robot can find the position of the target container and source container autonomously without having to hard code it.

Our multifingered grasping work handles grasp very well, but using a dexterous hand to do tool-using is much more interesting and valuable, such as using a screwdriver. Furthermore, the simulator used for RL agent training is CPU-based. Changing to a GPU-based simulator like Isaac Gym would accelerate the training speed and try more hyperparameters for better training results.

In the robotic pouring height estimation part, the audio-based method highly relies on the input audio quality. Even though we have improved the audio-based network with a multimodal network, an obvious limitation is that these audio-based methods cannot be utilized when the pouring motion generates only weak sound.

8.3 Future Work

There are a lot can be done to do to enhance the thesis topic:

- **Add semantic understanding for the service robot.** To make the robot understand what to do and how to do it by letting the robot understand the environment would be interesting. The service robot would be more intelligent if it could understand where it is located, what task the user wants it to do, and how.

- **Make the service robot learning more interesting and challenging tasks.** After grasping the object properly, the service robot is more useful if it can play with the grasped object. Learning in-hand dexterous manipulation skills would be very useful, as shown in Figure 8.2. In the figure, we plan to make the robot do more challenging tasks like using a power drill and screwdriver, playing the guitar and wipe the table. Some tasks are hard to accomplish using only one arm, it is necessary to study the problem of dual arm manipulation.

- **Add more sensor modalities in robotic pouring estimation.** Our current robotic pouring estimation method lacks vision sensing. It would be beneficial for a robot to estimate the liquid height using vision sensing and other modalities, such as pouring liquid with high viscosity.

- **Integrate a container classification network in height estimation.** As different liquid types have different sounds, training a classification network to classify liquid type from the audio is also possible. This result may also help the generalization of the liquid height estimation network.

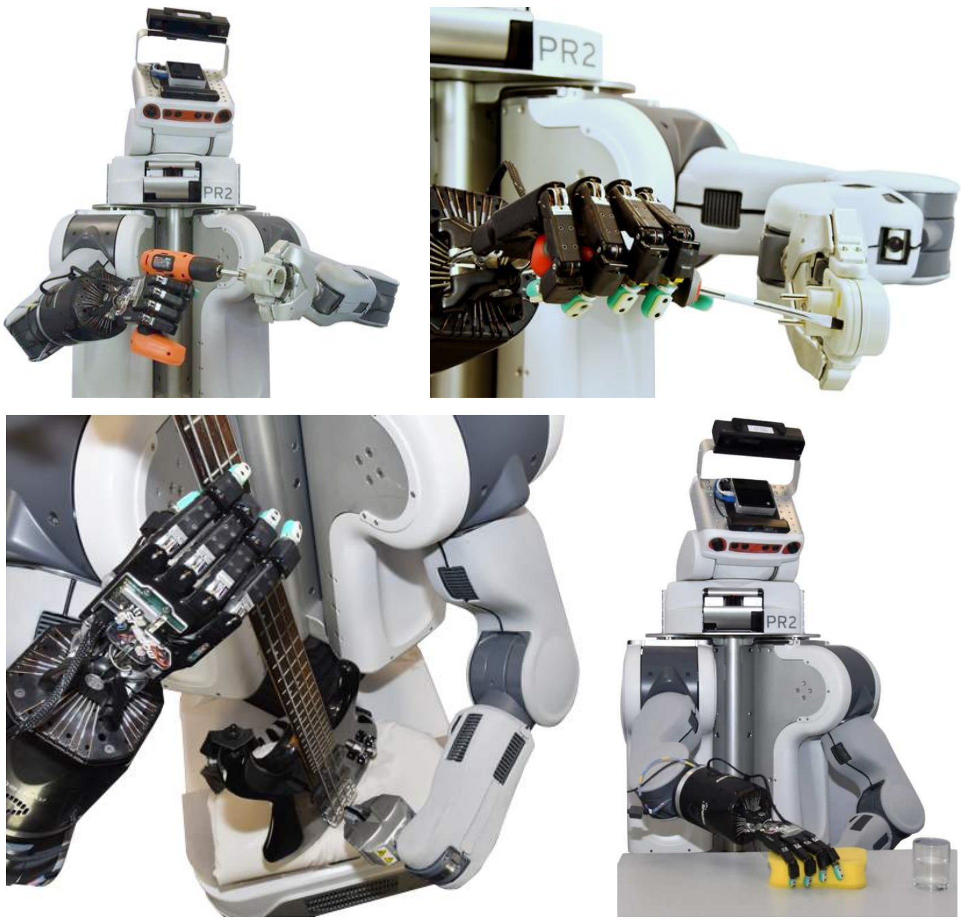


Figure 8.2: Examples of the future work of this thesis, robotic tool using, guitar playing and wipe the table.

- **Learning robot pouring motion that can adjust with environmental change.** Current robotic pouring motion assumes that the target container position is fixed. Using the force, motion trajectories, and visual data from our multimodal dataset and studying the complementarity and interaction between multiple modalities in robotic pouring would also be an exciting direction of future research.

Appendix A

List of Abbreviations

CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
DDPG	Deep Deterministic Policy Gradient
DoF	Degree of Freedom
FC	Fully-connected
GRU	Gated Recurrent Unit
GUI	Graphical User Interface
GWS	Grasp Wrench Space
HMM	Hidden Markov Model
ICP	Iterative Closest Point
IL	Imitation Learning
LfD	Learning from Demonstration

LSTM	Long Short-Term Memory
MDP	Markov Decision Process
MLP	Multi-Layer Perception
MSE	Mean Squared Error
NLP	Natural Language Processing
PCA	Principal Component Analysis
PID	proportional-integral-derivative
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
RMS	Root Mean Square
RNN	Recurrent Neural Network
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
STFT	Short-Time Fourier Transform
URDF	Unified Robot Description Format

Appendix B

Publications

Below are all the publications published during my PhD study, sorted by publication date.

[1] **Hongzhuo Liang**, Lin Cong, Norman Hendrich, Shuang Li, Fuchun Sun, and Jianwei Zhang. Multifingered grasping based on multimodal reinforcement learning. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):1174–1181, Apr. 2022.

[2] Lin Cong, **Hongzhuo Liang**, Philipp Ruppel, Yunlei Shi, Michael Görner, Norman Hendrich, and Jianwei Zhang. Reinforcement learning with vision-proprioception model for robot planar pushing. *Frontiers in Neurorobotics*, Jan. 2022.

[3] Wenkai Chen, **Hongzhuo Liang**, Zhaopeng Chen, Fuchun Sun, and Jianwei Zhang. Improving object grasp performance via transformer-based sparse shape completion. *Journal of Intelligent & Robotic Systems*, 104(45), 2022.

[4] **Hongzhuo Liang**, Chuangchuang Zhou, Shuang Li, Xiaojian Ma, Norman Hendrich, Timo Gerkmann, Fuchun Sun, and Jianwei Zhang. Robust robotic pouring using audition and haptics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10880–10887, Nov. 2020.

[5] Lin Cong, Michael Görner, Philipp Ruppel, **Hongzhuo Liang**, Norman Hendrich, and Jianwei Zhang. Self-adapting recurrent models for object pushing from learning in simulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5304–5310, 2020.

[6] Shuang Li, Jiaxi Jiang, Philipp Ruppel, **Hongzhuo Liang**, Xiaojian Ma, Norman Hendrich, Fuchun Sun, and Jianwei Zhang. A mobile robot hand-arm teleoperation system by vision and IMU. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10900–10906, 2020.

[7] Jinpeng Mi, **Hongzhuo Liang**, Nikolaos Katsakis, Song Tang, Qingdu Li, Changshui Zhang, and Jianwei Zhang. Intention-related natural language grounding via

object affordance detection and intention semantic extraction. *Frontiers in Neurorobotics*, 14:1–12, 2020.

[8] **Hongzhuo Liang**, Xiaojian Ma, Shuang Li, Michael Görner, Song Tang, Bin Fang, Fuchun Sun, and Jianwei Zhang. PointNetGPD: Detecting grasp configurations from point sets. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3629–3635, Nov. 2019.

[9] Shuang Li, Xiaojian Ma, **Hongzhuo Liang**, Michael Görner, Philipp Ruppel, Bing Fang, Fuchun Sun, and Jianwei Zhang. Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 416–422, May. 2019.

[10] Zhen Deng, Haojun Guan, Rui Huang, **Hongzhuo Liang**, Liwei Zhang, and Jianwei Zhang. Combining model-based Q-learning with structural knowledge transfer for robot skill learning. *IEEE Transactions on Cognitive and Developmental Systems*, 11(1):26–35, 2019.

[11] **Hongzhuo Liang**, Shuang Li, Xiaojian Ma, Norman Hendrich, Timo Gerkmann, Fuchun Sun, and Jianwei Zhang. Making sense of audio vibration for liquid height estimation in robotic pouring. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5333–5339, 2019.

[12] Shuang Li, **Hongzhuo Liang**, and Jianwei Zhang. Path planning for wheeled mobile service robots based on improved genetic algorithm. In *Shanghai International Symposium on Human-Centered Robotics (HCR)*, pages 249–252, 2018.

[13] **Hongzhuo Liang**, Shuang Li, Michael Görner, and Jianwei Zhang. Generating robust grasps for unknown objects in clutter using point cloud data. In *Shanghai International Symposium on Human-Centered Robotics (HCR)*, pages 298–301, 2018.

Appendix C

Acknowledgements

Wenkai
Yihong
Michael
Jianwei
Yuyang
Chao
Qingdu
Jiaxi
Xiaojian
Eugen
Jinpeng
Shuang
Chaobin
Philipp
Dads
Jianzhi
Chuang
Bernd
Hannes
T
A
M
S
UHH
Sam
Lu
Hao
Niklas
Di
Daniel
Bo
Yunlei
Dan
Yanzi
Ge
Song
Moms
Lin
Hamburg
Dieter
Marc
Junhu
Stefan
Zhaopeng
Wiebke
Aimen
Andreas
Zhen
German
Yannick
Norman
Timo
Florens
Margarete
Denis

Bibliography

[1] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. My lips are concealed: Audio-visual speech enhancement through obstructions. In *INTERSPEECH*, pages 4295–4299, 2019. `doi:10.21437/Interspeech.2019-3114`.

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research (IJRR)*, 39(1):3–20, 2020. `doi:10.1177/0278364919887447`.

[3] Tim Baier-Löwenstein. *Lernen der Handhabung von Alltagsgegenständen im Kontext eines Service-Roboters*. PhD thesis, Universität Hamburg, 2008. URL: `https://ediss.sub.uni-hamburg.de/handle/ediss/2192`.

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

[5] Alexandre Bernardino, Marco Henriques, Norman Hendrich, and Jianwei Zhang. Precision grasp synergies for dexterous robotic hands. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 62–67, 2013. `doi:10.1109/ROBIO.2013.6739436`.

[6] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis — a survey. *IEEE Transactions on Robotics (T-RO)*, 30(2):289–309, 2013. `doi:10.1109/TRO.2013.2289018`.

[7] Christoph Borst, Max Fischer, and Gerd Hirzinger. Grasping the dice by dicing the grasp. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 4, pages 3692–3697, 2003. `doi:10.1109/IROS.2003.1249729`.

[8] Samarth Brahmbhatt, Ankur Handa, James Hays, and Dieter Fox. ContactGrasp: Functional multi-finger grasp synthesis from contact. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2386–2393, 2019. `doi:10.1109/IROS40897.2019.8967960`.

[9] Sascha Brandi, Oliver Kroemer, and Jan Peters. Generalizing pouring actions between objects using warped parameters. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 616–621, 2014. `doi:10.1109/HUMANOIDS.2014.7041426`.

[10] Peter Brook, Matei Ciocarlie, and Kaijen Hsiao. Collaborative grasp planning with multiple object representations. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2851–2858, 2011. doi:10.1109/ICRA.2011.5980490.

[11] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In *IEEE International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015. doi:10.1109/ICAR.2015.7251504.

[12] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[13] Yevgen Chebotar, Karol Hausman, Zhe Su, Gaurav S Sukhatme, and Stefan Schaal. Self-supervised regrasping using spatio-temporal tactile features and reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1960–1966, 2016. doi:10.1109/IROS.2016.7759309.

[14] Wenkai Chen, Hongzhuo Liang, Zhaopeng Chen, Fuchun Sun, and Jianwei Zhang. Improving object grasp performance via transformer-based sparse shape completion. *Journal of Intelligent & Robotic Systems*, 104(45), 2022. doi:10.1007/s10846-022-01586-4.

[15] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3D object detection network for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6526–6534, 2017. doi:10.1109/CVPR.2017.691.

[16] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, pages 103–111, 2014.

[17] HeeSun Choi, Cindy Crump, Christian Duriez, Asher Elmquist, Gregory Hager, David Han, Frank Hearl, Jessica Hodgins, Abhinandan Jain, Frederick Leve, Chen Li, Franziska Meier, Dan Negrut, Ludovic Righetti, Alberto Rodriguez, Jie Tan, and Jeff Trinkle. On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. *Proceedings of the National Academy of Sciences*, 118(1), 2021. doi:10.1073/pnas.1907856118.

[18] Matei Ciocarlie, Corey Goldfeder, and Peter K. Allen. Dimensionality reduction for hand-independent dexterous robotic grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3270–3275, 2007. doi:10.1109/IROS.2007.4399227.

[19] 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. doi:doi.org/10.1177/0278364909105606.

[20] Samuel Clarke, Travers Rhodes, Christopher G. Atkeson, and Oliver Kroemer. Learning audio feedback for estimating amount and flow of granular material. In *Conference on Robot Learning (CoRL)*, pages 529–550, 2018.

[21] David Coleman, Sachin Chitta, and Nikolaus Correll. Reducing the barrier to entry of complex robotic software: a MoveIt! case study. *Journal of Software Engineering for Robotics*, pages 3–16, 2014.

[22] Jack Collins, Shelvin Chand, Anthony Vanderkop, and David Howard. A review of physics simulators for robotic applications. *IEEE Access*, 9:51416–51431, 2021. doi:10.1109/ACCESS.2021.3068769.

[23] Enric Corona, Albert Pumarola, Guillem Alenya, Francesc Moreno-Noguer, and Gregory Rogez. GanHand: Predicting human grasp affordances in multi-object scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5031–5041, 2020. doi:10.1109/CVPR42600.2020.00508.

[24] Zhen Deng. *Integrating Perception and Optimization for Dexterous Grasping and Manipulation*. PhD thesis, Universität Hamburg, 2019. URL: https://ediss.sub.uni-hamburg.de/handle/ediss/6122.

[25] Zhen Deng, Bin Fang, Bingwei He, and Jianwei Zhang. An adaptive planning framework for dexterous robotic grasping with grasp type detection. *Robotics and Autonomous Systems*, 140:103727, 2021. doi:10.1016/j.robot.2021.103727.

[26] Rosen Diankov and James Kuffner. OpenRAVE: A planning architecture for autonomous robotics. *Robotics Institute, Pittsburgh, PA, Technical Report CMU-RI-TR-08-34*, 2008.

[27] Chau Do and Wolfram Burgard. Accurate pouring with an autonomous robot using an RGB-D camera. In *Intelligent Autonomous Systems (IAS)*, pages 210–221, 2019. doi:10.1007/978-3-030-01370-7_17.

[28] Chau Do, Camilo Gordillo, and Wolfram Burgard. Learning to pour using deep deterministic policy gradients. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3074–3079, 2018. doi:10.1109/IROS.2018.8593654.

[29] Chau Do, Tobias Schubert, and Wolfram Burgard. A probabilistic approach to liquid level detection in cups using an RGB-D camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2075–2080, 2016. doi:10.1109/IROS.2016.7759326.

[30] Chenyu Dong, Masaru Takizawa, Shunsuke Kudoh, and Takashi Suehiro. Precision pouring into unknown containers by service robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5875–5882, 2019. doi:10.1109/IROS40897.2019.8967911.

[31] Guoguang Du, Kai Wang, Shiguo Lian, and Kaiyong Zhao. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *Artificial Intelligence Review*, 54(3):1677–1734, 2021. doi:10.1007/s10462-020-09888-5.

[32] Christof Elbrechter, Jonathan Maycock, Robert Haschke, and Helge Ritter. Discriminating liquids using a robotic kitchen assistant. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 703–708, 2015. doi:10.1109/IROS.2015.7353449.

[33] Yongxiang Fan, Te Tang, Hsien-Chung Lin, and Masayoshi Tomizuka. Real-time grasp planning for multi-fingered hands by finger splitting. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4045–4052, 2018. doi:10.1109/IROS.2018.8594369.

[34] Thomas Feix, Javier Romero, Heinz-Bodo Schmiedmayer, Aaron M. Dollar, and Danica Kragic. The grasp taxonomy of human grasp types. *IEEE Transactions on Human-Machine Systems*, 46(1):66–77, 2016. doi:10.1109/THMS.2015.2470657.

[35] Fanny Ficuciello, Pietro Falco, and Sylvain Calinon. A brief survey on the role of dimensionality reduction in manipulation learning and control. *IEEE Robotics and Automation Letters (RA-L)*, 3(3):2608–2615, 2018. doi:10.1109/LRA.2018.2818933.

[36] Fanny Ficuciello, A. Migliozzi, G. Laudante, Pietro Falco, and Bruno Siciliano. Vision-based grasp learning of an anthropomorphic hand-arm system in a synergy-based control framework. *Science Robotics*, 4(26), 2019. doi:10.1126/scirobotics.aao4900.

[37] Fanny Ficuciello, Damiano Zaccara, and Bruno Siciliano. Synergy-based policy improvement with path integrals for anthropomorphic hands. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1940–1945, 2016. doi:10.1109/IROS.2016.7759306.

[38] Anthony P. French. In vino veritas: A study of wineglass acoustics. *American Journal of Physics*, pages 688–694, 1983.

[39] Eric Gaba. Minimal surface curvature planes — Wikipedia, the free encyclopedia, 2006. [Online; accessed 20-December-2021]. URL: https://en.wikipedia.org/wiki/File:Minimal_surface_curvature_planes-en.svg.

[40] Shane Griffith, Vladimir Sukhoy, Todd Wegter, and Alexander Stoytchev. Object categorization in the sink: Learning behavior-grounded object categories with water. In *ICRA Workshop on Semantic Perception, Mapping and Exploration*, 2012.

[41] Di Guo, Tao Kong, Fuchun Sun, and Huaping Liu. Object discovery and grasp detection with a shared convolutional neural network. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2038–2043, 2016. doi:10.1109/ICRA.2016.7487351.

[42] Di Guo, Fuchun Sun, Bin Fang, Chao Yang, and Ning Xi. Robotic grasping using visual and tactile sensing. *Information Sciences*, 417:274–286, 2017. doi:10.1016/j.ins.2017.07.017.

[43] Junhu He. *Robotic In-hand Manipulation with Push and Support Method*. PhD thesis, Universität Hamburg, 2017. URL: https://ediss.sub.uni-hamburg.de/handle/ediss/7359.

[44] Junhu He and Jianwei Zhang. Push resistance in in-hand manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2488–2493, 2014. doi:10.1109/IROS.2014.6942901.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi:10.1109/CVPR.2016.90.

[46] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *IEEE International Conference on Computer Vision (ICCV)*, pages 858–865, 2011. doi:10.1109/ICCV.2011.6126326.

[47] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. doi:10.1109/MSP.2012.2205597.

[48] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[49] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *IEEE/CVF Conference on computer vision and pattern recognition (CVPR)*, pages 2261–2269, 2017. doi:10.1109/CVPR.2017.243.

[50] Yongqiang Huang and Yu Sun. Learning to pour. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7005–7010, 2017. doi:10.1109/IROS.2017.8206626.

[51] Yongqiang Huang and Yu Sun. A dataset of daily interactive manipulation. *The International Journal of Robotics Research (IJRR)*, pages 879–886, 2019. doi:10.1177/0278364919849091.

[52] Sakiko Ikeno, Ryo Watanabe, Ryuta Okazaki, Taku Hachisu, Michi Sato, and Hiroyuki Kajimoto. Change in the amount poured as a result of vibration when pouring a liquid. In *Haptic Interaction*, pages 7–11. Springer, 2015.

[53] Stephen James, Marc Freese, and Andrew J. Davison. PyRep: Bringing V-REP to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

[54] Liyiming Ke, Jingqiang Wang, Tapomayukh Bhattacharjee, Byron Boots, and Siddhartha Srinivasa. Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6185–6191, 2021. doi:10.1109/ICRA48506.2021.9561662.

[55] Monroe Kennedy, Karl Schmeckpeper, Dinesh Thakur, Chenfanfu Jiang, Vijay Kumar, and Kostas Daniilidis. Autonomous precision pouring from unknown containers. *IEEE Robotics and Automation Letters (RA-L)*, 4(3):2317–2324, 2019. doi:10.1109/LRA.2019.2902075.

[56] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations (ICLR)*, 2015.

[57] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference for Learning Representations (ICLR)*, 2014.

[58] David Kirkpatrick, Bhubaneswar Mishra, and Chee-Keng Yap. Quantitative Steinitz's theorems with applications to multifingered grasping. *Discrete & Computational Geometry*, 7(3):295–318, 1992. doi:10.1007/BF02187843.

[59] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154, 2004. doi:10.1109/IROS.2004.1389727.

[60] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1097–1105, 2012.

[61] Visak Kumar, Tucker Hermans, Dieter Fox, Stan Birchfield, and Jonathan Tremblay. Contextual reinforcement learning of visuo-tactile multi-fingered grasping policies. In *NeurIPS Workshop on Robot Learning: Control and Interaction in the Real World*, 2019.

[62] Joshua D. Langsfeld, Krishnanand N. Kaipa, Rodolphe J. Gentili, James A. Reggia, and Satyandra K. Gupta. Incorporating failure-to-success transitions in imitation learning for a dynamic pouring task. In *Workshop on Compliant Manipulation: Challenges and Control, Chicago, IL*, 2014.

[63] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi:10.1109/5.726791.

[64] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020. doi:10.1126/scirobotics.abc5986.

[65] Michelle A. Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8943–8950, 2019. doi:10.1109/ICRA.2019.8793485.

[66] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research (IJRR)*, 34(4-5):705–724, 2015. doi:10.1177/0278364914549607.

[67] Jiaxin Li, Ben M Chen, and Gim Hee Lee. SO-Net: Self-organizing network for point cloud analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9397–9406, 2018. doi:10.1109/CVPR.2018.00979.

[68] Miao Li, Kaiyu Hang, Danica Kragic, and Aude Billard. Dexterous grasping under shape uncertainty. *Robotics and Autonomous Systems*, 75:352–364, 2016. doi:10.1016/j.robot.2015.09.008.

[69] Hongzhuo Liang, Lin Cong, Norman Hendrich, Shuang Li, Fuchun Sun, and Jianwei Zhang. Multifingered grasping based on multimodal reinforcement learning. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):1174–1181, 2022. doi:10.1109/LRA.2021.3138545.

[70] Hongzhuo Liang, Shuang Li, Michael Görner, and Jianwei Zhang. Generating robust grasps for unknown objects in clutter using point cloud data. In *Shanghai International Symposium on Human-Centered Robotics (HCR)*, pages 298–301, 2018. URL: https://dl.acm.org/doi/10.5555/3281667.3281730.

[71] Hongzhuo Liang, Shuang Li, Xiaojian Ma, Norman Hendrich, Timo Gerkmann, Fuchun Sun, and Jianwei Zhang. Making sense of audio vibration for liquid height estimation in robotic pouring. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5333–5339, 2019. doi:10.1109/IROS40897.2019.8968303.

[72] Hongzhuo Liang, Xiaojian Ma, Shuang Li, Michael Görner, Song Tang, Bin Fang, Fuchun Sun, and Jianwei Zhang. PointNetGPD: Detecting grasp configurations from point sets. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3629–3635, 2019. `doi:10.1109/ICRA.2019.8794435`.

[73] Hongzhuo Liang, Chuangchuang Zhou, Shuang Li, Xiaojian Ma, Norman Hendrich, Timo Gerkmann, Fuchun Sun, and Jianwei Zhang. Robust robotic pouring using audition and haptics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10880–10887, 2020. `doi:10.1109/IROS45743.2020.9340859`.

[74] Raymond R. Ma, Lael U. Odhner, and Aaron M. Dollar. Dexterous manipulation with underactuated fingers: Flip-and-pinch task. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3551–3552, 2012. `doi:10.1109/ICRA.2012.6225348`.

[75] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio, and Ken Goldberg. Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017. `doi:10.15607/RSS.2017.XIII.058`.

[76] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1957–1964, 2016. `doi:10.1109/ICRA.2016.7487342`.

[77] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac Gym: High performance GPU-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[78] Carolyn Matl, Robert Matthew, and Ruzena Bajcsy. Haptic perception of liquids enclosed in containers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7142–7149, 2019. `doi:10.1109/IROS40897.2019.8968528`.

[79] Hamza Merzić, Miroslav Bogdanović, Daniel Kappler, Ludovic Righetti, and Jeannette Bohg. Leveraging contact forces for learning to grasp. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3615–3621, 2019. `doi:10.1109/ICRA.2019.8793733`.

[80] Jinpeng Mi, Hongzhuo Liang, Nikolaos Katsakis, Song Tang, Qingdu Li, Changshui Zhang, and Jianwei Zhang. Intention-related natural language grounding

via object affordance detection and intention semantic extraction. *Frontiers in Neurorobotics*, 14:1–12, 2020. `doi:10.3389/fnbot.2020.00026`.

[81] Andrew T. Miller and Peter K. Allen. GraspIt! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004. `doi:10.1109/MRA.2004.1371616`.

[82] Douglas Morrison, Peter Corke, and Jürgen Leitner. EGAD! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation. *IEEE Robotics and Automation Letters (RA-L)*, 5(3):4368–4375, 2020. `doi:10.1109/LRA.2020.2992195`.

[83] Roozbeh Mottaghi, Connor Schenck, Dieter Fox, and Ali Farhadi. See the glass half full: Reasoning about liquid containers, their volume and content. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1889–1898, 2017. `doi:10.1109/ICCV.2017.207`.

[84] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-DOF GraspNet: Variational grasp generation for object manipulation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2901–2910, 2019. `doi:10.1109/ICCV.2019.00299`.

[85] Van-Duc Nguyen. Constructing force-closure grasps. *The International Journal of Robotics Research (IJRR)*, 7(3):3–16, 1988. `doi:10.1177/027836498800700301`.

[86] Zherong Pan and Dinesh Manocha. Motion planning for fluid manipulation using simplified dynamics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4224–4231, 2016. `doi:10.1109/IROS.2016.7759622`.

[87] Daehyung Park, Zackory Erickson, Tapomayukh Bhattacharjee, and Charles C. Kemp. Multimodal execution monitoring for anomaly detection during robot manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 407–414, 2016. `doi:10.1109/ICRA.2016.7487160`.

[88] Kunal J. Pithadiya, Chintan K. Modi, and Jayesh D. Chauhan. Selecting the most favourable edge detection technique for liquid level inspection in bottles. *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)*, 3:34–44, 2011.

[89] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE/CVF Conference of Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. `doi:10.1109/CVPR.2017.16`.

[90] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, pages 1–6, 2009.

[91] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6620–6629, 2017. doi:10.1109/CVPR.2017.701.

[92] Maximo A. Roa, Max J. Argus, Daniel Leidner, Christoph Borst, and Gerd Hirzinger. Power grasp planning for anthropomorphic robot hands. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 563–569, 2012. doi:10.1109/ICRA.2012.6225068.

[93] Eric Rohmer, Surya P. N. Singh, and Marc Freese. CoppeliaSim (formerly V-REP): a versatile and scalable robot simulation framework. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1321–1326, 2013. doi:10.1109/IROS.2013.6696520.

[94] Leonel Rozo, Pablo Jiménez, and Carme Torras. Force-based robot learning of pouring skills using parametric hidden Markov models. In *IEEE International Workshop on Robot Motion and Control*, pages 227–232, 2013. doi:10.1109/RoMoCo.2013.6614613.

[95] Philipp Ruppel, Norman Hendrich, Sebastian Starke, and Jianwei Zhang. Cost functions to specify full-body motion and multi-goal manipulation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3152–3159, 2018. doi:10.1109/ICRA.2018.8460799.

[96] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4, 2011. doi:10.1109/ICRA.2011.5980567.

[97] Hannes P Saal, Jo-Anne Ting, and Sethu Vijayakumar. Active estimation of object dynamics parameters with tactile sensors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 916–921, 2010. doi:10.1109/IROS.2010.5649191.

[98] Ricardo Sanchez-Matilla, Konstantinos Chatzilygeroudis, Apostolos Modas, Nuno Ferreira Duarte, Alessio Xompero, Pascal Frossard, Aude Billard, and Andrea Cavallaro. Benchmark for human-to-robot handovers of unseen containers with unknown filling. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):1642–1649, 2020. doi:10.1109/LRA.2020.2969200.

[99] Marco Santello, Martha Flanders, and John F. Soechting. Postural hand synergies for tool use. *Journal of Neuroscience*, 18(23):10105–10115, 1998. doi:10.1523/JNEUROSCI.18-23-10105.1998.

[100] Connor Schenck and Dieter Fox. Reasoning about liquids via closed-loop simulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017. doi:10.15607/RSS.2017.XIII.014.

[101] Connor Schenck and Dieter Fox. Visual closed-loop control for pouring liquids. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2629–2636, 2017. doi:10.1109/ICRA.2017.7989307.

[102] Connor Schenck and Dieter Fox. Perceiving and reasoning about liquids using fully convolutional networks. *The International Journal of Robotics Research (IJRR)*, 37(4-5):452–471, 2018. doi:10.1177/0278364917734052.

[103] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017. doi:10.15607/RSS.2017.XIII.050.

[104] Lin Shao, Fabio Ferreira, Mikael Jorda, Varun Nambiar, Jianlan Luo, Eugen Solowjow, Juan Aparicio Ojea, Oussama Khatib, and Jeannette Bohg. UniGrasp: Learning a unified model to grasp with multifingered robotic hands. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):2286–2293, 2020. doi:10.1109/LRA.2020.2969946.

[105] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[106] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3D object detection in RGB-D images. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 808–816, 2016. doi:10.1109/CVPR.2016.94.

[107] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(56):1929–1958, 2014.

[108] Julia Starke, Christian Eichmann, Simon Ottenhaus, and Tamim Asfour. Human-inspired representation of object-specific grasps for anthropomorphic hands. *International Journal of Humanoid Robotics (IJHR)*, 17(2):1–20, 2020. doi:10.1142/S0219843620500085.

[109] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. doi:10.1109/CVPR.2015.7298594.

[110] Minija Tamosiunaite, Bojan Nemec, Aleš Ude, and Florentin Wörgötter. Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives. *Robotics and Autonomous Systems*, 59(11):910–922, 2011. doi:10.1016/j.robot.2011.07.004.

[111] Alessandro Tasora, Radu Serban, Hammad Mazhar, Arman Pazouki, Daniel Melanz, Jonathan Fleischmann, Michael Taylor, Hiroyuki Sugiyama, and Dan Negrut. Chrono: An open source multi-physics dynamics engine. In *High Performance Computing in Science and Engineering*, pages 19–49. Springer International Publishing, 2016. `doi:10.1007/978-3-319-40361-8_2`.

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

[113] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research (IJRR)*, 36(13-14):1455–1473, 2017. `doi:10.1177/0278364917735594`.

[114] Andreas ten Pas and Robert Platt. Using geometry to detect grasp poses in 3D point clouds. In *Robotics Research: Volume 1*, pages 307–324. Springer International Publishing, 2018. `doi:10.1007/978-3-319-51532-8_19`.

[115] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. `doi:10.1109/IROS.2017.8202133`.

[116] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012. `doi:10.1109/IROS.2012.6386109`.

[117] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1082–10828, 2018. `doi:10.1109/CVPRW.2018.00143`.

[118] Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter K. Allen. Shape completion enabled robotic grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2442–2447, 2017. `doi:10.1109/IROS.2017.8206060`.

[119] Jacob Varley, Jonathan Weisz, Jared Weiss, and Peter K. Allen. Generating multi-fingered robotic grasps via deep learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4415–4420, 2015. `doi:10.1109/IROS.2015.7354004`.

[120] Yikai Wang, Wenbing Huang, Fuchun Sun, Tingyang Xu, Yu Rong, and Junzhou Huang. Deep multimodal fusion by channel exchanging. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 4835–4845, 2020.

[121] Connie Weadon. Pouring in the dark. *Future Reflections*, 10(3), 1991.

[122] Emile S. Webster and Clive E. Davies. The use of Helmholtz resonance for measuring the volume of liquids and solids. *Sensors*, 10(12):10663–10672, 2010. `doi:10.3390/s101210663`.

[123] Justin Wilson, Auston Sterling, and Ming C. Lin. Analyzing liquid pouring sequences via audio-visual neural networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7702–7709, 2019. `doi:10.1109/IROS40897.2019.8968118`.

[124] Thomas Wimböck, Benjamin Jahn, and Gerd Hirzinger. Synergy level impedance control for multifingered hands. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 973–979, 2011. `doi:10.1109/IROS.2011.6094555`.

[125] Bohan Wu, Iretiayo Akinola, Jacob Varley, and Peter K. Allen. MAT: Multi-fingered adaptive tactile grasping via deep reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

[126] Chaozheng Wu, Jian Chen, Qiaoyu Cao, Jianchi Zhang, Yunxin Tai, Lin Sun, and Kui Jia. Grasp proposal networks: An end-to-end solution for visual learning of robotic grasps. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 13174–13184, 2020.

[127] TzYing Wu, JuanTing Lin, TsunHsuang Wang, ChanWei Hu, Juan Carlos Niebles, and Min Sun. Liquid pouring monitoring via rich sensory inputs. In *European Conference on Computer Vision (ECCV)*, pages 335–351, 2018.

[128] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. Unseen object instance segmentation for robotic environments. *IEEE Transactions on Robotics (T-RO)*, 37(5):1343–1359, 2021. `doi:10.1109/TRO.2021.3060341`.

[129] Akihiko Yamaguchi, Christopher G. Atkeson, and Tsukasa Ogasawara. Pouring skills with planning and learning modeled from human demonstrations. *International Journal of Humanoid Robotics (IJHR)*, 12(03):1–40, 2015. `doi:10.1142/S0219843615500309`.

[130] Xinchen Yan, Jasmine Hsu, Mohi Khansari, Yunfei Bai, Arkanath Pathak, Abhinav Gupta, James Davidson, and Honglak Lee. Learning 6-DOF grasping interaction via deep geometry-aware 3D representations. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3766–3773, 2018. `doi:10.1109/ICRA.2018.8460609`.

[131] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker Jr, Rodriguez Alberto, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6D pose estimation in the Amazon picking challenge. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1386–1383, 2017. doi:10.1109/ICRA.2017.7989165.

[132] Kevin Zhang, Mohit Sharma, Manuela Veloso, and Oliver Kroemer. Leveraging multimodal haptic sensory data for robust cutting. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 409–416, 2019. doi:10.1109/Humanoids43949.2019.9035073.

[133] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, 2018. doi:10.1109/CVPR.2018.00472.

Erklärung der Urheberschaft

Ich versichere an Eides statt, dass ich die vorliegende Dissertation im Bereich der Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Insbesondere wurden dabei keine nicht im Quellenverzeichnis benannten Internet-Quellen verwendet. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

[illegible]

Hamburg, Jan. 27, 2022	Hongzhuo Liang
Ort, Datum	Unterschrift

Erklärung zur Veröffentlichung

Ich erkläre hiermit mein Einverständnis zur Einstellung dieser Dissertation in den Bestand der Bibliothek.

[illegible]

Hamburg, Jan. 27, 2022 — Hongzhuo Liang
Ort, Datum — Unterschrift