Chubu University
Department of Robotic Science and Technology

Development of Object Detection and Grasp Position Estimation in a Commercial Robotic Dishwasher System

Ph.D. Thesis Report

Presented by

Suraj Prakash Pattar

Supervisor

Hironobu Fujiyoshi

January 2023

Abstract

Artificial intelligence (AI) and robotics research have the potential to revolutionize various industries and tasks. However, deploying these technologies in real-world scenarios can be challenging due to issues such as the need for large amounts of annotated data and the difficulty of adapting to different environments and lighting conditions. In this thesis, we present two main contributions towards making it easier to deploy AI and robotics research in real-world scenarios.

Firstly, we propose the use of a shallow network for the detection of the grasp position of an object. Unlike other networks that require a large amount of computation and training time, our shallow network is trained with synthetic images and is able to detect the grasp position efficiently. This is particularly useful for tasks where quick and precise grasp positioning is essential, such as in automated assembly lines or pick-and-place operations.

Secondly, we introduce the use of mobile robots and invisible markers to automatically collect a large amount of annotated data for object detection and grasp position estimation. The invisible markers, which are only visible under UV light, allow us to generate a large amount of real data in a short amount of time and accurately annotate it. This is a significant improvement over synthetic data, which may not generalize well to real-world scenarios, and manual data collection, which is time-consuming and labor-intensive.

In addition to these main contributions, we also discuss the use of key-point estimation for 6D pose estimation and a deep-learning method for dealing with harsh lighting conditions in depth data acquisition. Overall, this thesis presents practical solutions for deploying AI and robotics research in real-world scenarios, enabling the efficient and effective use of these technologies in a variety of applications.

Acknowledgements

I am deeply grateful to my supervisor, Professor Hironobu Fujiyoshi, for providing me with the opportunity to pursue my Ph.D. under their guidance and mentorship. Their unwavering support and encouragement throughout this journey has been invaluable, and I feel fortunate to have had such a knowledgeable and supportive supervisor. Their expertise and insight have greatly influenced my research, and I have learned so much from them. I am grateful for the time and effort they have dedicated to helping me grow as a researcher and scholar.

I would also like to extend my sincere thanks to Professor Tsubasa Hirakawa for their guidance and support during my time as a Ph.D. candidate. Their insights and expertise were invaluable, and I am grateful to have had the opportunity to learn from such a knowledgeable and respected scholar. Their contributions to my research and overall development as a researcher have been immeasurable, and I am deeply grateful for their support.

I am also deeply grateful to all of the members of my lab, including Ms. Miya, Mr. Araki, and Mr. Kousuke, for their constant help and support during my time as a Ph.D. candidate. The collaborative and supportive environment in the lab was crucial to my success, and I am grateful to have been a part of such a wonderful team. Their assistance and guidance in navigating the challenges of Ph.D. life were invaluable, and I am thankful to have had such supportive colleagues. I would like to extend my sincere thanks to each and every one of them for their contributions to my research and overall development as a researcher.

I would also like to express my sincere gratitude to Connected Robotics Inc. for their support and understanding during my time as a Ph.D. candidate. The flexibility and accommodations provided by Mr. Sawanobori allowed me to balance my work and research commitments. The constant help from my colleagues, including Kan San, Kaeru San, and Sam, helped me to overcome obstacles, and I am deeply grateful for their support. Their encouragement and understanding were crucial to my success, and I am thankful to have had such a supportive work environment.

I would also like to extend my heartfelt thanks to my girlfriend, Asuka, for her unwavering love and support throughout this journey. Her artistic touch in illustrations and her expertise in designing engaging slides for my presentations greatly improved the impact of my research presentations. Her constant encouragement, understanding, and belief in me were instrumental in helping me navigate the challenges of Ph.D. life.

Finally, I would like to thank my family and friends for their constant love and support throughout this journey. Their encouragement and understanding were crucial to my success, and I am deeply grateful to have such a supportive network of loved ones. This journey would not have been possible without their help and understanding, and I am thankful to have such wonderful people in my life.

Contents

List of Figures

List of Tables

List of abbreviations

2D	Two-dimensional
3D	Three-dimensional
6D	Six-dimensional
AI	Artificial Intelligence
AIST	Advanced Industrial Science and Technology
DOF	Degree of Freedom
ROS	Robot Operating System
UE4	Unreal Engine 4
UV	Ultraviolet

Listings

People think of education as something that they can finish. - **Isaac Asimov**.

Chapter 1

Introduction

1.1 Motivation

Robots using artificial intelligence (AI) are starting to enter the space of humans. Researchers are working to make these robots more responsive and accurate in their recognition abilities. This thesis aims to explore how to apply research in robotics and AI in the real world.

While research in robotics and AI is advancing quickly, real-world applications of these robots are slow to develop. There are not many intelligent and autonomous robots available on the market that have been practically deployed. One reason for this could be that researchers do not fully understand the challenges of applying their research in real-world situations, and there is insufficient collaboration between academia and end-users [7], [8].

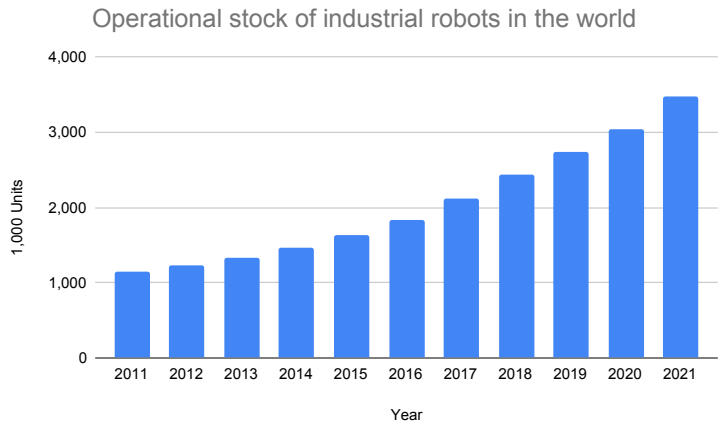
In this thesis, we will investigate the challenges of implementing a computer vision system, specifically object detection and grasp position estimation, for a dishwashing system in a busy restaurant kitchen. We will focus on how to achieve high accuracy while using limited resources, which is essential for the successful deployment of a robotic system. We will also examine ways to speed up the data collection process, a crucial step for supervised machine learning. Additionally, we will discuss a major challenge for computer vision systems in real-world environments: harsh lighting conditions. Throughout our research, we will emphasize the difficulties of deploying a commercial system that is designed to operate reliably in the real-world.

We define the real-world as an environment that is not restricted or modified to facilitate robot operation. The real-world is often complex, dynamic, chaotic, and unpredictable [9]. Despite extensive testing, the real-world can differ from the test setup, causing algorithms to fail [10].

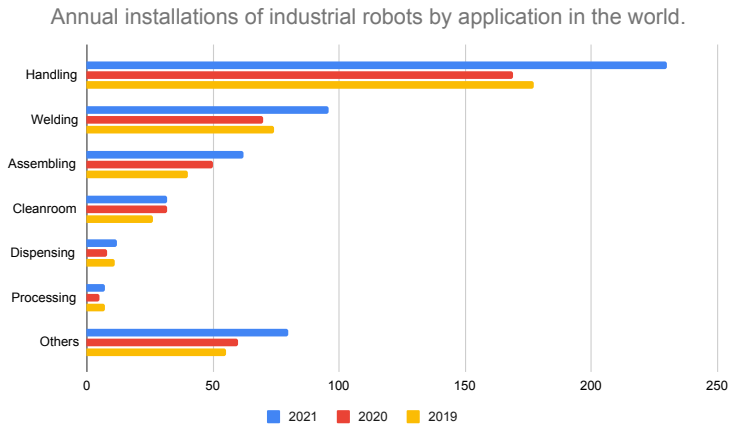
In the rest of this chapter, we will begin by introducing industrial robots and exploring the reasons for their widespread adoption in factories. We will then move on to service robots and examine the obstacles for their deployment. Next, we will examine service robots specifically in restaurants. We will then describe our problem domain, which is the dishwashing system, and provide examples of other implementations.

1.1.1 Robots and AI in the Real World

The word "robot" was first coined by the author Karel Čapek in his play Rossum's Universal Robots (R.U.R). It originally referred to forced labor. Today, robots are present in many areas of our world, assisting humans in manufacturing [11], handling hazardous materials [12], exploring space [13], and other tasks that are tedious or dangerous.



(a)



(b)

Figure 1.1: Trends in industrial robots [2].

Industrial Robots

Traditionally, robots were first used in manufacturing industries [14]. Their adoption in factories has been steadily increasing, as shown in Fig. 1.1 (a) [2]. They assist in automating dangerous tasks for humans, such as welding, handling heavy objects, painting, assembly, mechanical cutting, grinding, and other applications. Fig. 1.1 (b) shows the most popular applications for robots, based on the number of installations.

Because industrial robots can be given designated spaces away from people, they can be easily installed in production lines for safety. Industrial robots are typically large, made of stiff materials, strong, and energy-intensive. Additionally, work in factories is commonly divided into small, specialized tasks, making it easy to assign robots to specific tasks that are carefully divided into small, pre-determined steps. Factories can also be divided into separate areas where humans are allowed and where they aren't, allowing the installation of safety sensors to detect any unwanted presence in the robots' work area.

Another advantage of industrial robots is that they only need to be programmed once for a specific task. The programming, or motion sequence, of the robot does not need to be changed as long as the environment and the product it is working on remain the same. Programming these robots does not require expert skill, and any overhead costs in training are worth it as long as the products are mass-produced.

Recently, robotic systems have been developed that use a perception module to find the

(a)

(b)

(c)

Figure 1.2: Representative sample of industrial robots.

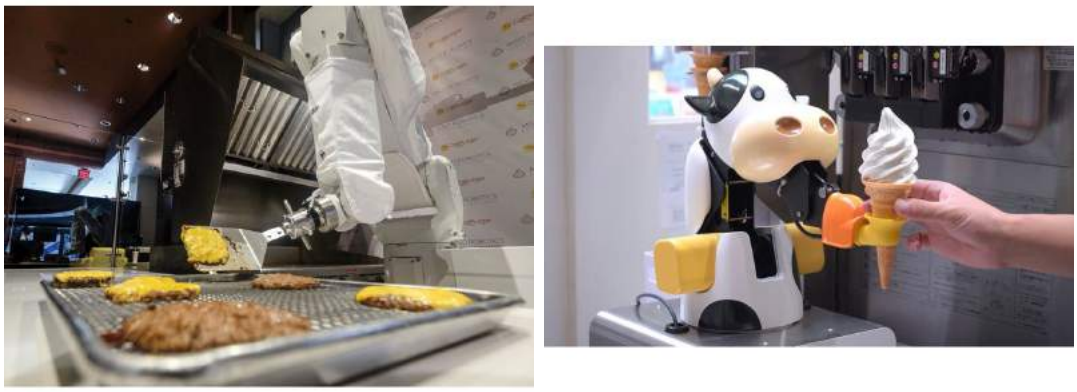
location of graspable objects, commonly referred to as the bin-picking problem [15, 16, 17]. Even in these scenarios, the type of objects that the perception module observes is highly restricted in shape, dimensions, and lighting conditions. It is also easier to restrict unwanted objects in the robot's work-space.

Lighting in factories can also be precisely controlled to meet the requirements of the vision system [18]. Because humans will not be in contact with the robots or interact with the environment designed for robots, the lighting settings do not need to take human needs into account. Traditional computer vision algorithms can meet many of the vision needs of industrial robots under these favorable conditions [19, 15, 16, 17, 20]. In contrast to the black boxes of modern deep-learning-based computer vision algorithms, these algorithms can be fine-tuned manually and the pre-processing steps are investigateable.

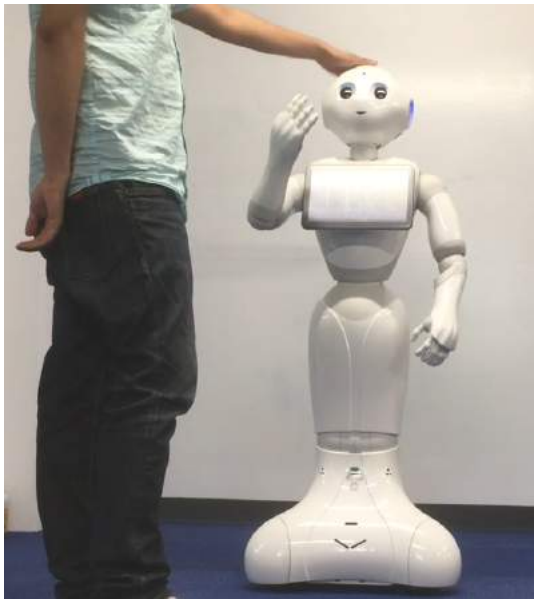
The cost of industrial robots is a significant point to consider. A typical industrial robot can range in price from $50,000 to 400,000$, including the cost of the robot, controllers, teach pendants, and application-specific peripherals. The reasons for such a wide range in cost depend on the customization required for a particular application.

Additionally, the cost of industrial cameras ranges from 1,000 USD to 10,000 USD. These costs are considered reasonable and affordable in large factories as the increase in production output and profits outweigh the costs. Industries that utilize these robots are typically large organizations with access to a large budget [21].

Service Robots



(a) (b)



(c)

Figure 1.3: Representative sample of service robots.

A service robot is defined as a robot that performs useful tasks for humans or equipment excluding industrial automation applications [22]. The field of service robotics covers a wide range of applications, the majority of which have distinctive designs and various levels of automation, from fully tele-operated to fully autonomous operation. Figure 1.3 shows a representative sample of service robots in action.

As time has progressed, service robots are entering spaces occupied by humans. To work alongside humans, these robots are typically smaller, made of softer materials, more economical, and more aware of their environments. Since the environment these robots operate in is designed for humans, they need to be more adaptive and intelligent than their industrial counterparts. They are equipped with various sensors on-board for gathering information from their environment, such as vision, sound, sonar, lidar, and haptic sensors.

These sensors need to be more intelligent than those used in industrial robots as the range of data they are exposed to is much larger. However, to keep the overall cost of the robot low,

Table 1.1: Industrial three-d camera vs regular three-d camera.

	Ensenso	**ZED**
Price (USD)	10,000	450
Dimension	175 x 50 x 52	124.5 x 30.5 x 26.5
Operating Temperature (C)	0 - 40 °C	0 - 45° C
IP Rating	IP 65	NA
Depth Range	Adjustable	0.15 - 12 m
Connectivity	GigE	USB

their costs need to be much lower. To balance these challenges, the algorithms for processing sensor data need to be more advanced.

The challenges for a service robot are specific to its environment and application. For example, a legged robot designed for inspection must be able to traverse rugged terrain. A mobile robot used in a busy restaurant must be able to carry objects while also being aware of its surroundings, including humans, floor conditions, and challenging lighting conditions. In the following subsection, we will provide more information about the specific challenges of the restaurant kitchen environment that are relevant to our use case.

Kitchen environment

Kitchen environments present a unique set of challenges when deploying robotic systems. While they share some similarities with factory environments, such as the use of fixed robots and sensors that perform repetitive tasks, there are also significant differences. In a kitchen, the environment is not as well controlled as it is in a factory. Humans can still move around freely, and the lighting is set according to their needs. This means that the occasional foreign object may enter the environment, and vision systems may be affected by various noises such as food particles, water, smoke, and shadows.

Unlike factories, the volume of work in a kitchen is not consistent. There may be times when there is a large workload in a short period, followed by periods of low workload. To meet these demands, the system needs to be efficient and use the minimum resources necessary to complete the task.

Additionally, the footprint of the robots and sensors needs to be small, as they must operate in tightly constrained spaces due to the limited operating space in a restaurant. Their operating range should be between a few centimeters to a few meters.

1.2 Dishwasher system

In this section, we will provide a brief overview of our target application, which is dishwashing in a restaurant. Dishwashing is one of the most tedious tasks in a restaurant and can cause back and shoulder pain in workers when done for extended periods of time. In addition, commercial dishwashers emit a significant amount of heat and steam which can also be detrimental to the worker's skin condition.

Furthermore, the chemicals used for dishwashing can be harmful when in contact for long periods. This is also a cost department for the restaurant because it is crucial for a restaurant to clean the dishes after their use, but it does not contribute to the profits of the restaurant. This makes it an excellent candidate for automation using robots.

Before we proceed to automate the dishwashing process, let's take a look at the steps involved in dishwashing at a typical restaurant.

- Gather the dishes or containers from the tables to the cleaning room.
- Discard any leftover food or drinks in the container for disposal or recycling.

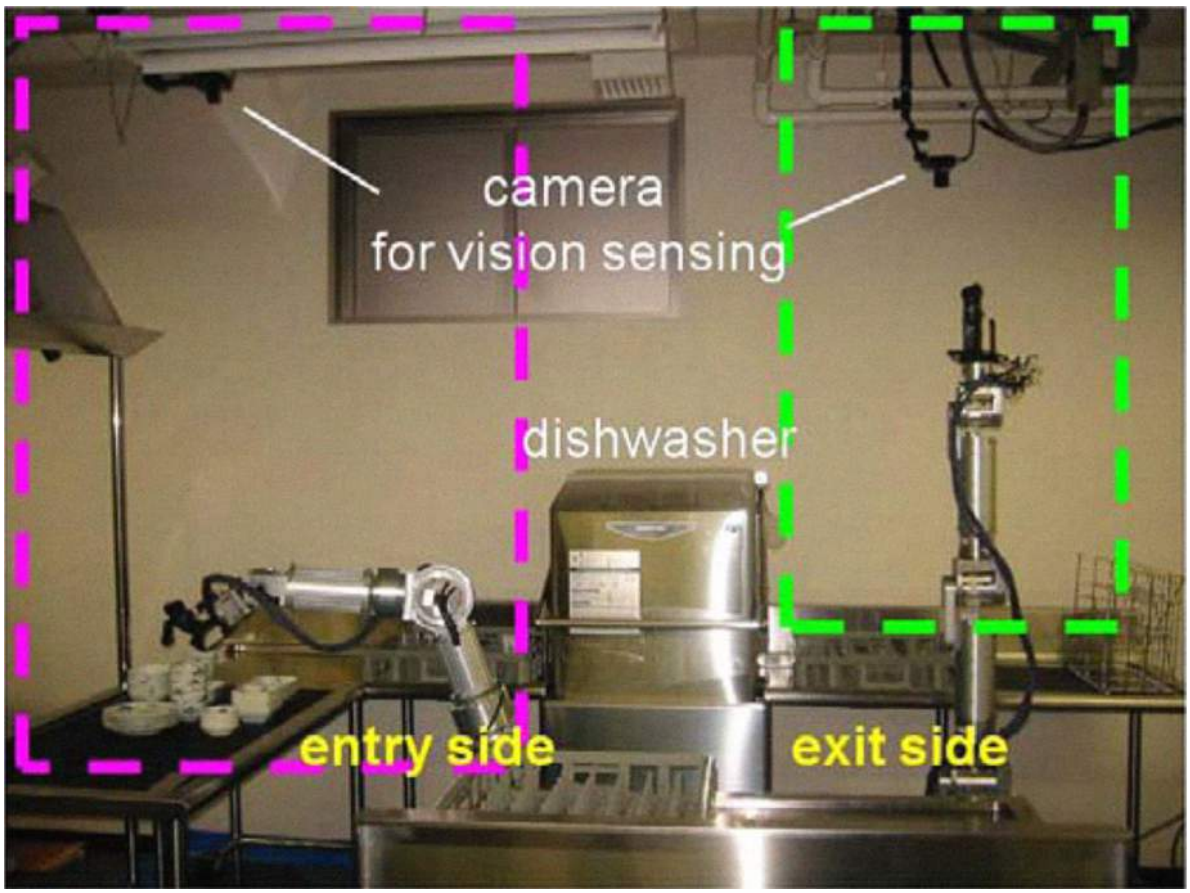


Figure 1.4: Representative systems for dishwasher implementation [3].

- Sort the dishes that need to be brushed vigorously to remove any sticky materials, such as rice.
- Sort small items such as spoons and forks into a carrier rack for easy handling.
- Place the dishes in the dishwasher rack in a way that maximizes the number of dishes that can be cleaned in a single cycle.
- Place the rack inside the dishwasher and wait for the cleaning cycle to be completed.
- Once the dishwashing cycle is completed, sort the dishes according to their type, shape, and size for easy redistribution in the restaurant.

When automating the dishwashing process using robots and a perception system, one needs to consider the following points.

- The cost of the overall system should be low.
- The system should have low latency.
- It should be able to adapt to low-volume and high-volume workloads and perform reliably.
- The environment cannot be constrained as it needs to work together with humans in tight spaces.

There have been several approaches to tackle the task of dishwashing in Japanese kitchens, as reported in works such as Kosuge et al. [3], Fukuzawa et al. [23], and Lo et al. [4]. One

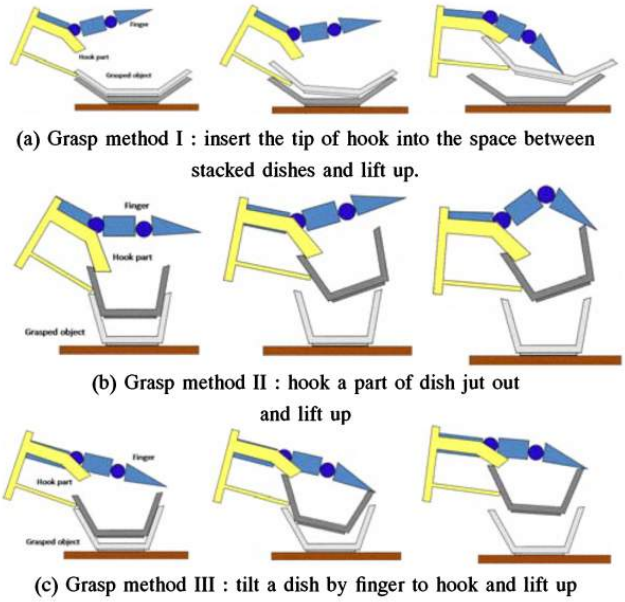


Figure 1.5: Hook type gripper used by Kosuge et al. [3].

such system is shown in Fig. 1.4. The system used in our work, as described by Lo et al., will be discussed in more detail in Chapter 2.

In the system depicted in Fig. 1.4, we can see that it has two robots on either side of the dishwasher. These two 7-DOF robots are installed on the floor space in front of the layout. The robots are equipped with hook-structured hands, as shown in Fig. 1.5. The cameras are fixed to the ceiling of the room.

Kosuge et al. [3] mention the use of a vision system to detect the position of the dishes and the appropriate grasping position. However, the details of a sophisticated vision system that would be capable of accurately estimating the grasp positions, as shown in Fig. 1.5, are not available.

Observing the system, we discover that it does not take into consideration all the variables that exist in an actual restaurant setting. For example, the dishes in the system are placed the right way up and stacked neatly. However, in a real restaurant, the dishes would usually contain some leftovers or sticky materials, as seen in Fig. 1.6. In such cases, it is difficult for the gripper to grasp the object without dirtying itself or failing to grasp it entirely.

(a) (b)

Figure 1.6: Sample of dirty dishes in a real restaurant.

1.2.1 Gripper Selection

(a) Parallel-plate gripper. (b) Three jaw gripper. (c) Jamming gripper.

(d) Dexterous hand gripper. (e) Single suction gripper. (f) Multi-suction gripper.

Figure 1.7: Representative sample of types of grippers.

In the previous section, we briefly described an example of an automated dishwasher system and the type of gripper used. In this section, we will describe the types of grippers available and their relative advantages and disadvantages.

Parallel-plate gripper: These are the most common grippers found in robotics research [24, 25]. For smaller jobs, these grippers are frequently found in manufacturing environments as well. The end-effector has two flat-edged parallel jaws that open and close, securing the part and firmly maintaining its stability. However, when it comes to handling asymmetrical shapes and sizes, these kinds of grippers are rigid.

Three-jaw gripper: These are a variant of the parallel plate gripper with three jaws. Three fingers or jaws on a three-finger gripper close around the object to hold it in place. These are frequently used for cylinder-shaped or round objects. They have more flexibility in grasping irregular shapes compared to parallel-plate grippers.

Jamming gripper: These grab and hold the object using granular materials, like coffee beans or plastic beads, enclosed in a soft outer shell. It is a flexible option that is best suited for producing irregularly shaped items because it can be altered to fit different sizes and shapes. Jamming grippers function by lightly touching the object, followed by the application of air pressure, which causes the internal grains of the gripper to "jam," resulting in a secure grip. When the air is let out, the granular material "unjams," releasing the object.

Dexterous hand gripper: These are a type of robotic end-effector that mimics the functionality of a human hand and consists of several fingers that are individually actuated. They offer a wide range of grasping and manipulation tasks, able to handle a variety of object shapes and sizes, perform fine manipulation, and provide a high level of compliance and adaptability. However, they can be more complex, costly, and require advanced control algorithms to operate effectively. They are also prone to wear and tear and may not be suitable for high-speed or high-volume automation tasks.

Suction gripper: These can pick up objects using the difference between a vacuum and atmospheric pressure. However, one limitation of suction grippers is that they need a relatively flat surface to ensure a reliable grip. In the absence of flat surfaces, the suction cup might leak air, resulting in a failed grip. They are best suited for objects with smooth and flat surfaces.

When selecting a gripper for any real-world application, one needs to consider factors such as the price, complexity of motion, payload, and cleanability. The term "payload" describes the total weight, including the gripper, that the robot arm is capable of supporting.

Electric grippers such as parallel-plate grippers, three-jaw grippers, and dexterous hand grippers are popular in research since they allow precise control and are usually equipped with force sensors. The addition of a force sensor allows one to understand the forces necessary for each task.

However, when compared to pneumatic grippers such as the suction grippers shown in Fig. 1.7 (e) and (f), electric grippers provide less gripping force and tend to be more expensive. Additionally, electric grippers are harder to clean, and it can be difficult to use them in environments where they are exposed to water.

Pneumatic grippers are also easier to control since they usually have only two modes, either to grip or not grip. This simplicity makes them a great choice for use in restaurants where simplicity and ease of control are important. Additionally, their low cost and easy cleanability make them a cost-effective and practical option for use in these environments.

1.3 Machine Learning in Production

In this section, we will describe the common pitfalls that a machine-learning model might encounter when deployed in production. To begin with, we will first describe the machine learning approach used in research.

In research, machine learning models are typically trained on a fixed dataset. This is done to establish a baseline when evaluating a model. However, the majority of machine learning (ML) research has been focused on models, and the most popular datasets have been used for routine ML tasks without considering the size, complexity, or fidelity of these datasets to the underlying problem. This has led to the saturation of dataset-driven criteria for model quality and major issues with data cascades in real-world applications as a result of ignoring the fundamental significance of datasets, as reported by Mazumder et al. [26]. These issues have hindered the advancement of research in this field.

1.3.1 Challenges for Machine Learning in Production

In production, one needs to address software engineering issues as well, such as latency, compute resources (CPU/GPU/memory), real-time or batch processing, and security.

The data preparation phase of an ML project typically takes the most extended amount of time, up to 70 %. When selecting an algorithm, robustness - that is, how it responds to unforeseen circumstances - and interpretability are crucial considerations, as reported by Mayr et al. [27] and Whang et al. [28]. Thus, it is important to consider the real-world requirements and constraints of the problem at hand when selecting an algorithm and evaluating its performance.

1.3.2 Data Drift and Concept Drift

Data-drift can occur when the dataset used to train your model does not mimic the data that you receive in production. This is when your model starts to act strange and underperform. The difference between training and production data can be caused by multiple factors. In

our use case, this can occur when the environmental conditions of the testing area versus the deployed area are vastly different from each other.

Data is subject to change over time. In predictive models that assume a static relationship between input and output variables, this can lead to poor and degrading predictive performance. In the field of machine learning, this problem of changing underlying relationships in data is referred to as concept drift. In our use case, this can occur when the target objects are changed without retraining the model, resulting in a model that is no longer accurate. It is important to continuously monitor and retrain the model to ensure that it is able to adapt to these changes in the data.

1.4 Objective, Research Questions and Contributions

In the previous sections, we briefly described the current gaps in modern research activities in the development of robotic systems aimed at deployment in real-world scenarios. Therefore, the main objective of this thesis is:

To increase the deployability, robustness, and reliability of intelligent robotic systems, which can successfully perform under real-world conditions.

We shall now go over the fundamental research questions this thesis is based upon. We will also briefly describe our major contributions towards answering those research questions. Based on the above objective, we define our research questions as follows.

- **Research Question 1** Can the existing deep-learning models for object detection and estimating the grasp position be deployed in real world situations and perform reliably?
- **Research Question 2** How can we simplify the above models for a modular architecture and easier debugging?
- **Research Question 3** How to train such large deep-learning models to ensure robustness, large variey of datawithout spending lot of time on data collection?
- **Research Question 4** Can we decrease the amount of time required to collect data for such models?
- **Research Question 5** How to address the issue of non-ideal lighting conditions in the real world?

1.5 Organization

This thesis has two main outcomes: 1) Single suction grasp detection for symmetric objects using shallow networks trained with synthetic data; and 2) Automatic data collection for object detection and grasp-position estimation with mobile robots and invisible markers. The initial chapters of the thesis report are based on these two journal publications and are presented chronologically. The thesis report is organized as described below.

Chapter 2 introduces our system along with the software architecture. We also present the reasoning behind the various design choices.

Chapter 3 aims to address the problem of deploying a grasp detection system in a real-world system within its constraints and compares its performance with other systems deployed in research and industry. This chapter will address the research questions one to three stated above.

Chapter 4 aims to address the problem of data collection for an object detection and grasp position detection system used in the real world. We will provide our answer to research question four in this chapter.

Chapter 5 presents an extension of our work in chapter 4 to predict the 6D position of the objects.

Chapter 6 presents the lighting problem in uncontrolled, real-world environments and our attempts to solve it. We will provide our attempt to answer research question five in this chapter.

Chapter 7 presents a summary of the performed work along with its limitations. We present the conclusions and potential future lines of inquiry.

Chapter 2

System Architecture and Design Concept of Our Dishwasher System

2.1 Introduction

In the previous chapter, we briefly described the general architecture of a dishwasher system found in the literature. This chapter introduces the system architecture and the comprising components for the development and experimental evaluation of the dishwasher system. We will briefly describe the overall system design and the hardware and software components involved in our system.

We will begin by describing the process workflow, where we will go over the sequence of steps performed and their contribution to the overall system. This will include the tasks performed by the robots, the data flow, and the control flow.

Next, we shall describe each major hardware and software component in detail. This will include the robotic arms, grippers, cameras, and the software stack used for image processing, control, and communication. We will also describe the decision process for their selection and how they need to be modified for our particular use case.

Additionally, we will also describe the design choices for the system, such as the choice of gripper, and the reasons behind those choices. Overall, this chapter aims to provide a comprehensive understanding of the system architecture and the design choices that were made for our dishwasher system.

2.2 System Design

Our dishwasher system includes two 7-DOF collaborative robots located next to a commercial dishwasher, as shown in Fig. 4.2. A human worker helps with bringing the dishes from tables, emptying the food inside them, and placing them upside down in the pre-wash area (the area near Robot-1 in Fig. 4.2). The post-wash area is where the washed dishes are placed according to their type.

Both the pre-wash and post-wash areas have two levels. The upper level in the pre-wash area is reserved for trays and racks for small items such as spoons, cups, and forks. It is equipped with weight sensors to detect the presence of these items.

Two 3D stereo-vision cameras are placed over the lower level of the pre-wash area to ensure maximum coverage. These cameras capture images of the pre-wash area, which are later processed by the vision module for detecting the dishes and their 3D position in relation to the robot. We also use the in-hand camera of the second robot to detect the 2D location of the dishes after the dishwasher cycle.

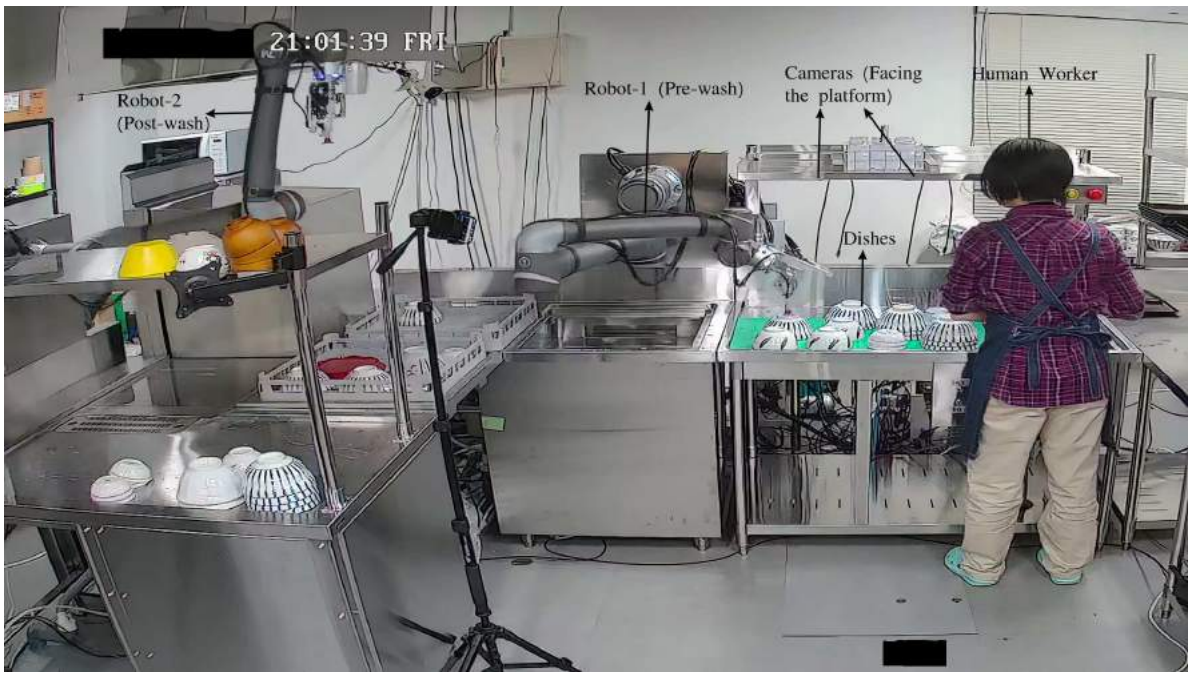


Figure 2.1: Dishwasher setup

The robots are fixed to fixtures such that the z-axis of the base is parallel to the floor. This configuration allows the robots to reach the maximum distance when picking and placing the objects. When not in use, the robots can be folded and kept out of distance from human workers. This is useful during servicing the dishwasher, low-volume washing when the robots are not necessarily required, or when cleaning the whole setup after a day's work.

The layout is L-shaped so as to make most of the small space allocated for the system in a restaurant. The L-shaped layout has numerous advantages as mentioned below.

- It is easily adaptable.
- Efficient for small spaces.
- Provides easy access to all work areas for human operators.
- Highly suitable for corner spaces in a room.
- It allows for easy separation of the items into the process and the finished results.

The above design considerations were made so that the dishwasher system was capable of processing a large number of dishes in a short period of time. At a typical restaurant, dishwashing is a sporadic activity. It is carried out at a high pace during peak hours, typically at breakfast, lunch, or dinner, depending on the particular restaurant. Table 2.1 shows the number of dishes washed at our target restaurant and the time required to wash them on two of the most busiest days [4].

Table 2.1: Time spent washing dishes by human workers. Table adopted from Lo et al. [4].

Day	**No. of dishes washed**	**Total time spent on washing dishes (h)**	**Time spent per dish (s)**
1	1344	2.83	7.57
2	1039	2.19	7.59

Figure 2.2: Dishwashing motion sequence.

2.3 Process Worfklow

In this section, we shall briefly describe the steps carried out by our dishwasher system.

1. Throw away the leftover food in the dishes and place them upside down. In a later iteration of the system, this step is automated by a dish-flipper instead of needing a human worker as shown in Fig. 2.2 (a) $\sim$ (h)[29].

2. Capture the image of the pre-wash area, recognize the items and estimate their location relative to the robot base in three dimensions as shown in Fig. 2.3 (a) $\sim$ (b).

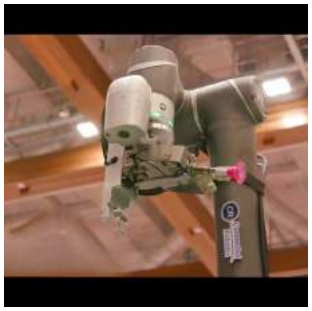
3. Plan the order of picking up the dishes to effectively utilize space in the rack and minimize the number of washes needed as shown in Fig. 2.4.

4. Pick and proceed to either wash the dishes with a brush or place them in the rack according to the type of the dish as shown in Fig.2.2 (i) $\sim$ (o).

(a) Capture images.

(b) Detect objects.

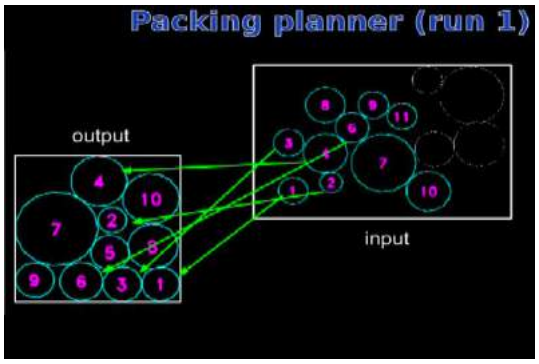
(c) Estimate 3D position with depth image.



(d) Capture images in post-wash.

(e) Detect dishes and their position in 2D.

Figure 2.3: Dishwashing vision process.



(a) Planning of 2D bin-packing.

(b) Placement of dishes for tight packing.

Figure 2.4: Dishwashing packing process [4].

5. Once the items are placed in the rack, push the rack inside the dishwasher and start the wash cycle as shown in Fig. 2.2 (p).

6. Once the wash cycle is completed, pull out the rack and place it under the camera in the post-wash area as shown in Fig. 2.2 (q) $\sim$ (v).

7. Recognize the dishes again, as they can move around during dishwashing as shown in Fig. 2.3 (c) $\sim$ (d).

8. Pick the dishes, sort, and stack them in an assigned place as shown in Fig. 2.2 (w) $\sim$ (z).

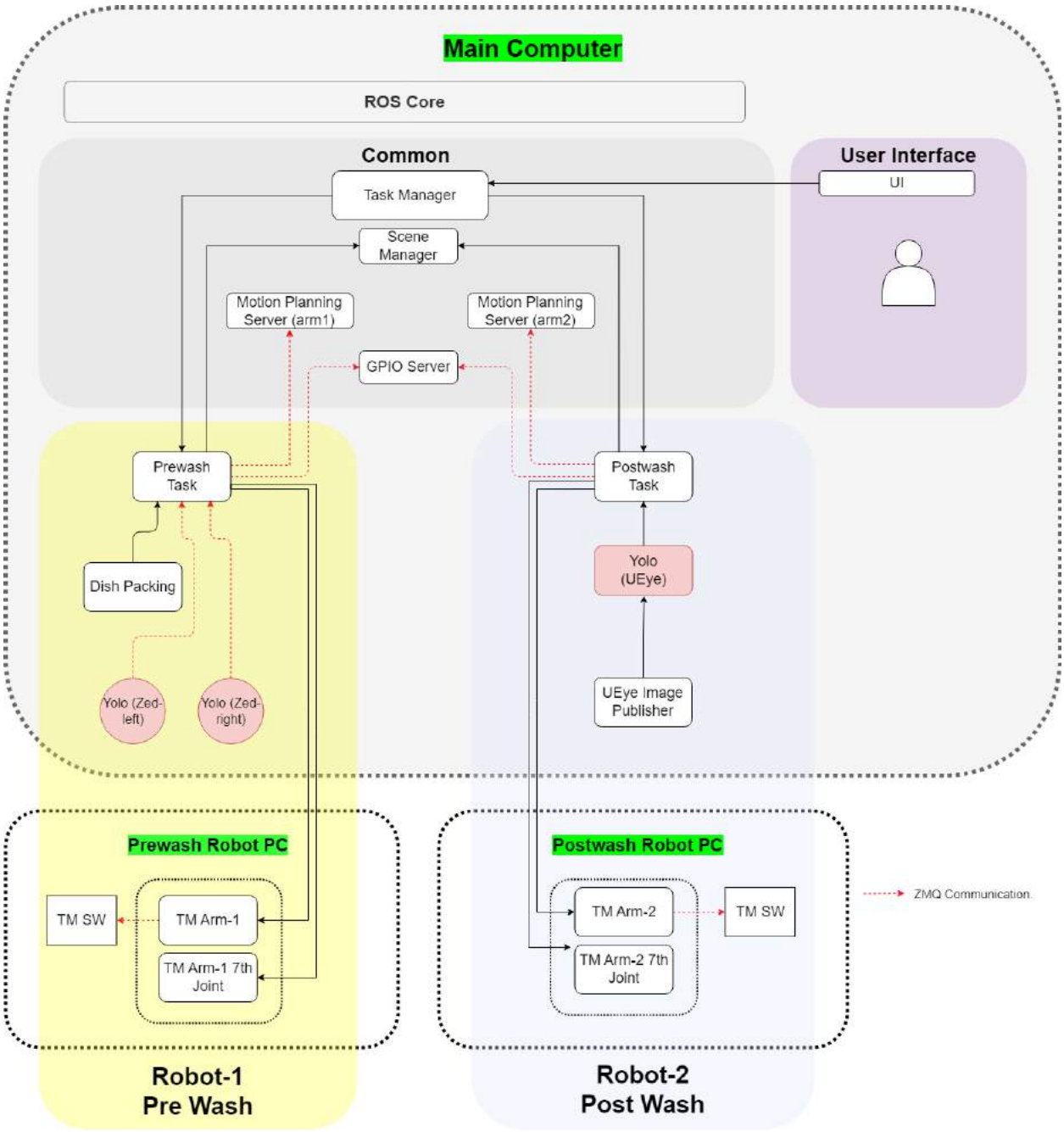


Figure 2.5: Dishwasher system software architecture.

2.3.1 System Architecture

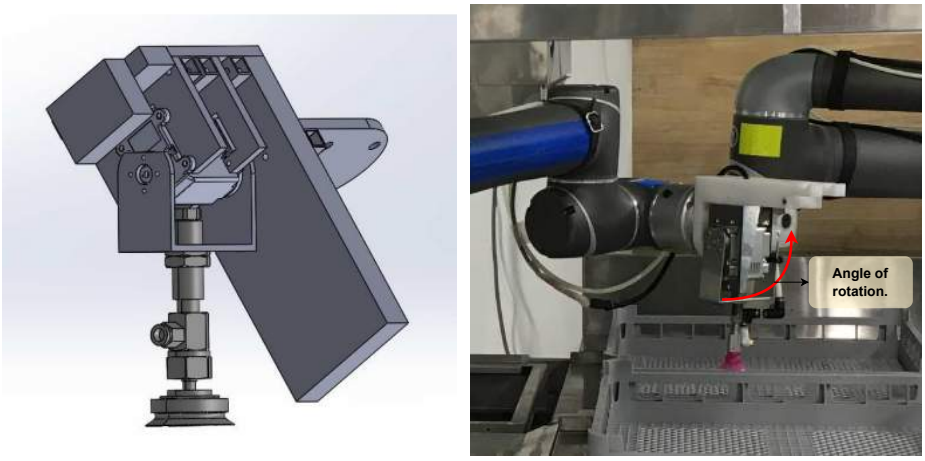
In Fig. 2.5, we can see the software architecture of our system. We have three main controllers, one for each robot and an integrating industrial computer. The industrial computer is a computer intended for industrial purposes. They have higher dependability and precision standards and are generally more expensive than consumer electronics.

The ZMQ communication by the vision module and the rest of the system is represented in red lines. The main controller receives the signals from the vision module, and planning module and plans the task sequence accordingly.

Table 2.2: TM-12 Specifications.

Weight (kg)	32.8
Controller weight (kg)	13.8
Max Reach (mm)	1,194
Max Payload (kg)	12
Typical speed (m/s)	1.3
Repeatability (mm)	±0.1
Ingress Protection	IP54
Communication	RS232, Ethernet, Modbus TCP/RTU
Power Supply	100 to 240 VAC, 50/60 Hz

Figure 2.18: TM-12 robot.



(a) 3D model of additional joint. (b) Robot with 7th axis.

Figure 2.19: Additional rotational joint [4].

2.4 Hardware

2.4.1 Collaborative Robot

In our system, we have two 6-DOF Techman 12(TM) collaborative robots on either side of a commercial dishwasher. The TM-12 robot is shown in Fig. 2.18, and its specifications are described in 2.6. These collaborative robots were chosen for their long reach and high payload for carrying the objects.

Additional Rotational Joint.

The robots are further modified by adding an additional rotation joint as shown in Fig. 2.19. This additional joint allows the robot to manipulate in hard-to-reach spaces and provides extra dexterity to pick up the objects. In addition, we can also attach additional fixtures to this joint which help to push and pull the dishwasher rack.

Table 2.3: IDS UEye XS specifications.

Dimension (mm)	26.5 x 23 x 23.7
Weight (g)	12
Frame rate (FPS)	15
Connectivity	USB 2.0
IP Rating	IP30
Price (USD)	250

Figure 2.18: IDS UEye-XS camera.

Table 2.4: Zed-Mini specifications.

Dimension (mm)	124.5 x 30.5 x 26.5
Weight (g)	63
Depth Technology	Stereo vision
Operating Temperature (in C)	0 - 45
Connectivity	USB 3.0
IP Rating	NA
Price (USD)	450

Figure 2.18: Zed-mini camera.

UEye 2D camera

The TM-12 robot has an IDS's UEye 2D camera attached to its 6th joint in its base model, seen in Fig.2.18. This camera can be useful for observing the robot's workspace. The resolution of this camera is 640x480.

In our use case, we use this 2D camera for object detection and 3D pose estimation of dishes in the post-wash area. In this case, the height in the 3D position is always fixed. We place the robot arm at a fixed height above the rack. Once the camera acquires an image, we can convert the pixel values to real-world measurements with a constant value that is manually measured.

2.4.2 ZED-Mini 3D camera

In the pre-wash area, we use two ZED-Mini 3D cameras. We can see the ZED-Mini (ZED-M camera in Fig. 2.18 along with its specifications in Table 2.19.

These cameras were selected for their small size, close distance depth range, high FPS data streaming, and price. The small size and the close distance depth range are very important in our use case since the cameras need to fit within a few centimeters of space in the hollow part of the upper surface. And since the upper surface is only around 60 70 centimeters from the workspace where the dishes are kept, our depth sensor needs to have a low depth range.

In the case of bigger cameras that have thicknesses greater than 4 centimeters, they risk

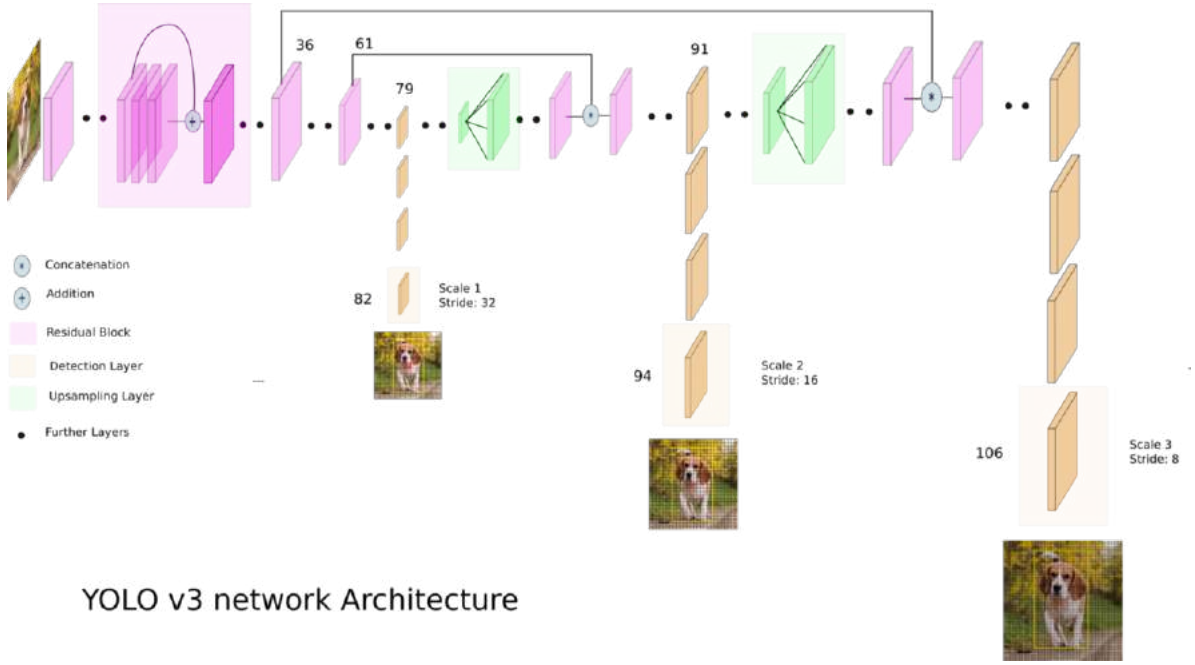


Figure 2.19: YOLOv3 network architecture [5].

becoming an obstacle for the robot. And if the depth range starts at a higher value than 30 - 40 centimeters, we would not be able to get the height of stacked dishes. The camera's low cost is also an important factor for us since using an industrial grade camera such as an Ensenso 3D camera, the price would be ten times greater.

Despite the above advantages, the ZED-M camera has a few caveats as well. It does not have any ingress protection (IP). IP is important in industrial devices since it defines how much exposure to dust or water a device can tolerate. In our use case, the devices are at constant risk of exposure to water. But since we have placed our cameras at a considerable distance from water outlets, we did not face any difficulties.

Another weakness of a stereo-vision camera like ZED-M is that it fails to obtain depth when there is a lack of texture or in harsh lighting conditions. We shall describe this phenomenon more in Chapter 6.

2.5 Software

In this section, we will describe the software components of our system. This will include the object-detection framework, deep-learning framework, and the messaging protocol for communication between different processes.

2.5.1 YOLOv3

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images[30]. The YOLO machine learning algorithm uses features learned by a deep convolutional neural network to detect an object. YOLOv3 network architecture can be seen in Fig. 2.19.

We selected YOLOv3 [5] for our object-detection framework due to its high-speed high-accuracy inference, ability to generalize from synthetic images, fast-training cycles, and low GPU-memory requirement.

2.5.2 Robot Operating System (ROS)

ROS is a collection of open-source robotics middleware. In spite of the fact that ROS is a collection of software frameworks for developing robot software rather than an operating system (OS), it offers services for a heterogeneous computer cluster, including hardware abstraction, low-level device control, the implementation of frequently used functionality, message-passing between processes, and package management.

Although we do not use ROS in our vision system, it is mainly used to control the motion of our robots and maintain communication between different modules.

2.5.3 ZeroMQ

For our messaging protocol, we decided to use ZeroMQ. ZeroMQ, also referred to as MQ, 0MQ, or zmq, resembles an embeddable networking library but performs concurrency framework-like operations. It provides sockets for atomic message transmission over a variety of transports, including in-process, inter-process, TCP, and multicast. With patterns like fan-out, pub-sub, task distribution, and request-reply, sockets can be connected N-to-N. It is quick enough to serve as the product fabric for clustered goods. Scalable multicore applications that were created as asynchronous message-processing tasks are provided by its asynchronous I/O model. It supports the majority of operating systems and has a number of language APIs [31].

2.5.4 NVIDIA Deep Learning Dataset Synthesizer (NDDS)

In order to enable computer vision researchers to export high-quality synthetic images with metadata, NVIDIA created the NDDS UE4 plugin. Images, segmentation, depth, object pose, bounding box, key-points, and customized stencils are supported by NDDS. The plugin also comes with various parts for producing images with a high degree of randomness. Lighting, objects, camera position, poses, textures, distractions, camera path following, and other elements are all randomly generated. These elements work together to make it simple for researchers to produce random scenes for deep neural network training [32]. We use this tool extensively to create synthetic data for training our deep-learning models.

An initial requirement for generating photorealistic synthetic data with this tool is an accurate 3D model of the object. Such models can be either created using a 3D scanner or digitally reconstructing the scanned model with the help of a professional artist.

We decided to use a professional artist to digitally reconstruct the scanned 3D model to provide a high-fidelity 3D model. The process of creating such models is described in Section 3.3.5.

2.5.5 Target objects

In the coming chapters, the subject of our deep-learning algorithms will be the dishes shown in Fig. 2.20. The objects are selected from a selection of dishes provided to us by our target restaurant.

Among the dishes provided to us, we had to reject a few for the following reasons.

- Dishes with an asymmetric shape as shown in Fig. 2.21 (a). Dishes like this were deemed difficult for the packing algorithm mentioned in [4].
- Dishes completely black in color as seen in Fig. 2.21 (b), (d), and (e). It is difficult for stereo-vision cameras to estimate depth for completely black dishes.

(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Figure 2.20: Dish classes.

(a) (b) (c) (d) (e)

Figure 2.21: Rejected dishes.

- Dishes with uneven surface as seen in Fig. 2.21 (d) and (e). Such dishes are harder to stack.

2.6 Summary

In this chapter, we have described our overall system design along with the software architecture. We have also defined each hardware and software component along with the explanation behind their selection.

In the coming chapters, we shall try to answer our research questions defined in Chapter 1. We shall use the system defined in this chapter for all the subsequent work presented in this thesis.

Chapter 3

Single Suction Grasp Detection for Symmetric Objects Using Shallow Networks Trained with Synthetic Data

3.1 Introduction

Grasping detection is an important issue for robotic manipulation. It is one of the critical processes among motion planning and robot movement to handle objects, as these steps depend heavily on accurate grasp prediction. In commercial applications, achieving high accuracy is paramount because imprecise grasping of an object can lead to loss of life or property. Furthermore, it needs to be achieved quickly to maintain efficiency. Over the years, this problem has been tackled with both deep learning and computer vision technologies [33, 34, 35].

The primary application field of robots has traditionally been the factories utilized by the manufacturing industry, which are a kind of static environment on fixed object manipulation. These days, applications are slowly transitioning to dynamic environments where objects can be placed at a random location with random orientation.

One such dynamic environment is a commercial kitchen equipped with dishwashers. A commercial dishwasher system automates the dishwashing process at a restaurant. The robots need to sort the dishes, and perform pre-processing on specific dishes, such as washing with a brush, placing them in the dishwasher rack, and sorting the dishes by type upon washing. Our target objects are dishes that are symmetric in shape and are kept upside down on a pre-wash platform. We make the above two assumptions to restrict our environment, as our focus in this study is a commercial human-robot collaborative dishwasher system used to automate dishwashing in restaurants.

Our setup of a commercial dishwasher system is shown in Fig. 4.2. This system has two main objectives: 1) to process numerous dishes in a short amount of time to prevent any bottlenecks in restaurant operation and 2) to pack the dishwasher rack efficiently to maximize the number of dishes washed in each cycle. Maintaining close to real-time constraints in a commercial system is essential to avoid ambiguity and process failure [36].

To achieve the above goals, the dishes are placed upside down on a rack by human assistants, and the dishes are picked up from the center to pack the rack efficiently. The symmetric circular shape of the dish allows for a more straightforward 2D bin packing method, which is outside the scope of this paper. We need to classify the dish type for sorting and determine if the dish needs to be washed with a brush or only water.

Because the pick-up area is constrained to the center of the dishes, the grasp pose estimations must be accurate for the above dishwashing process to run smoothly. This accuracy is also paramount for bigger dishes because the next step, which is placing the dishes in the rack,

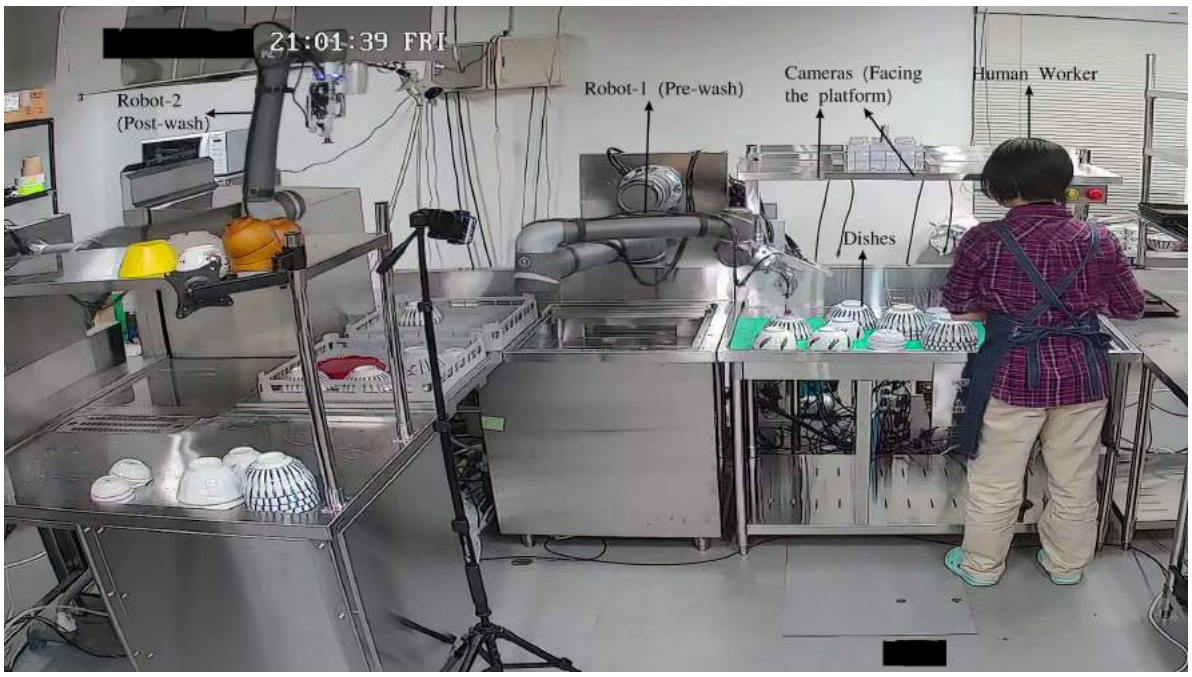


Figure 3.1: Dishwasher setup

depends on whether the robot picked up the dish from right at the center.

We focus on the applicability of deep learning grasp detection in real-world deployable systems with various constraints, including the cost of equipment, speed of operation, and limited processing. To this end, we propose a deep learning grasp detection method suitable for a commercial dishwasher robot system. We use shallow networks that are easy to train, easy to debug, accurate, and require fewer computational resources. We generate the training data in a synthetic gaming engine environment, which helps to reduce the cost of data collection.

In section 3.2 of this paper, we discuss other methods for grasp detection and briefly explain the methods against which we compare our grasp detector. In section 3.3, we describe the shallow convolutional network and the process of generating data using a gaming engine called Unreal Engine (UE4)[37]. We then explain how to use accurate 3D models of the dishes and place them in the UE4 environment to generate large amounts of realistic data in a short period. Also, we show how this method easily generalizes to different backgrounds, objects and grasp positions. In section 4.4, we present the experimental setup and the methodology to test the accuracy of grasp prediction and report the results. We conclude in section 3.5 with a brief summary and mention of future work.

3.2 Related Work

Grasping methods are broadly divided into deep learning methods and model fitting methods [38, 39]. Most grasp networks using deep learning focus on grasp predictions for the parallel-plate robotic gripper, [40, 41, 42, 43], which grasps the object by pressing it from two sides. Grasping with a suction gripper is also widely utilized in industrial use cases [44, 45] and in the Amazon Picking Challenge [46, 47].

Suction grippers need planar surfaces close to the centroid of objects to grasp effectively [48]. Various approaches to estimate the grasp position have been developed, including those using only depth images [44, 49], or making use of point clouds [47, 48], and those using RGB images [40, 50].

Table 3.1: Comparison of related works.

	Method	Input	Gripper Type	Hardware
Delft Amazon Picking Challenge [47]	Model fitting (Super 4PCS + Heuristics)	Point cloud	Single suction gripper + Pinch mechanism	Training + Inference: Nvidia Titan X GPU
Dex-Net 3.0 [48]	End-to-end deep learning (GQ-CNN)	Point cloud	Single suction gripper	Training: Nvidia Titan X GPUs x3 Inference: Intel i5-6400 + Nvidia GeForce 980 GPU
Real-Time Grasp Detection [40]	End-to-end deep learning (CNN)	RGB-D image	Parallel plate gripper	Training + Inference: Nvidia Tesla K20 GPU
Depth Image-Based Grasping [44]	End-to-end deep learning + Visual servoing	Depth image	Two-vacuum-cup suction gripper	Training: Intel i7-6700K CPU + Nvidia 1080Ti GPU Inference: Intel i7-4790 CPU + Nvidia 1080 GPU
FGE [49]	Model fitting (Template matching)	Cropped depth image	Parallel plate gripper	Inference: Intel i7600 CPU

3.2.1 Methods using Point Clouds

Hernandez et al. make use of point clouds for estimating the pose of objects [47]. As point clouds are susceptible to challenging lighting conditions and reflections, they also use heuristics to improve the estimations. Specifically, they estimate the pose of objects using Super 4PCS [51], a global alignment algorithm that needs filtered point clouds and 3D models of objects. The accuracy of this method is low, so it requires user-defined heuristics to estimate the grasp position.

Mahler et al. make use of GQ-CNNs, trained on point clouds, grasps, and grasp reward labels [48]. Their method focuses on the viability of the suction grip itself rather than detecting the grasp position. Namely, it predicts whether the grasp will hold when the object is grasped, moved, or shaken during transportation.

Deep learning networks that take point clouds for input data need powerful GPUs and a long training duration. Mahler et al.'s method [48] takes approximately 12 hours on three NVIDIA Titan-X GPUs. It needs powerful GPUs and processing equipment during inference as well.

3.2.2 Methods using RGB and depth images

Redmon and Angelova proposed a CNN-based method that takes RGB images as input, classifies objects, and predicts multiple grasp positions [40]. As it predicts rectangular grasp positions, it is well suited for objects that can be grasped in multiple ways by a parallel plate gripper. In such end-to-end networks, it is difficult to analyze the cause when the model fails to train. In addition, they require a large amount of training data.

Fast Grasp Evaluator

The fast grasp evaluator (FGE) [49] is a template matching method for depth images that uses only the depth image and templates of the gripper to find the optimum grasping points. Its process flow shown in Fig. 3.2 is as follows.

1. Create a binary image of the depth image based on a pre-set threshold that depends on the expected distance between the object and the camera.

2. Compute the contact region by using the convolution of the shape of a suction gripper template and the above binary image.

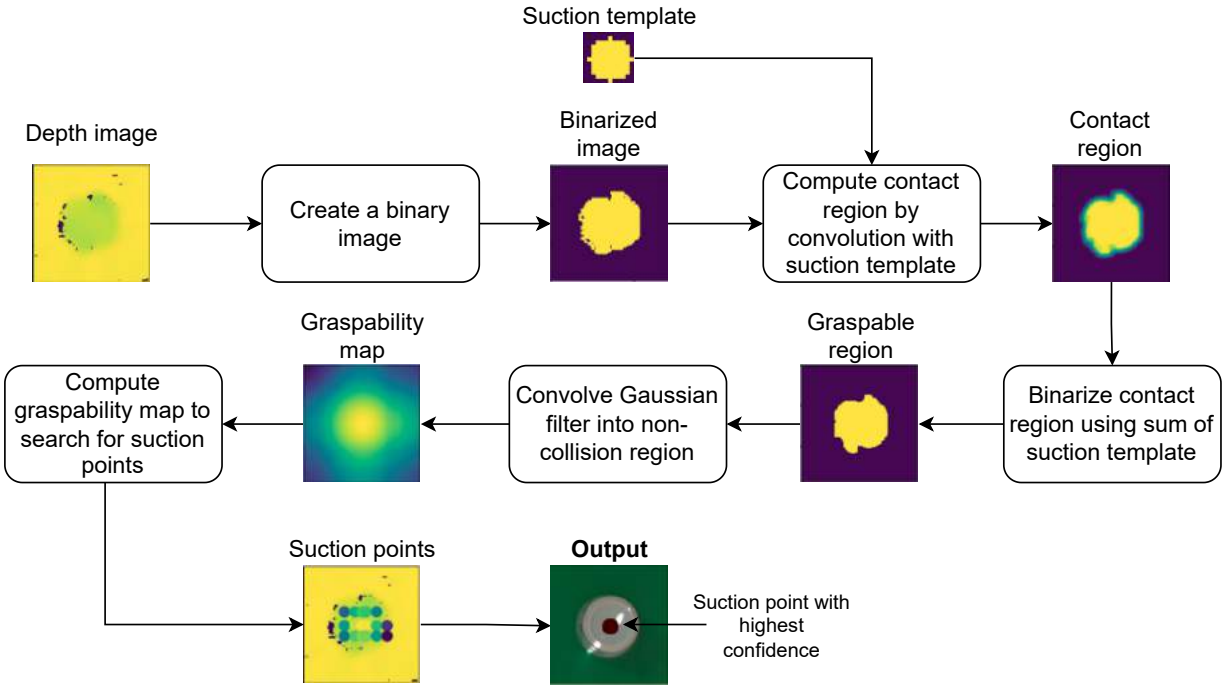


Figure 3.2: Fast grasp evaluator workflow.

3. Binarize the contact region using the sum of the suction template. The non-collision region is the result of the binarized contact region.

4. Compute the graspability map by using the convolution of the non-collision region with a Gaussian filter.

5. Search for suction points in the graspability map, which divide the graspability map into small areas.

6. Provide a list of possible 3D coordinates for the grasping point along with a score indicating the confidence of a successful grasp for each coordinate.

Generally, FGE is used for predicting grasp positions for objects of similar height in the pick-up space. Since our objects are of varying height, we first crop the object based on the output of the object detector and then set the threshold dynamically. As it is a template matching method, to increase the accuracy, we had to slightly expand the bounding box to show the background as well. As a result, we could not place multiple objects close to each other because we wanted to avoid overlapping of their bounding boxes.

One advantage of this method is that it does not need an RGB image to predict the grasp position. However, this increases its reliance on the depth image. Also, we found that its computational cost increases as the number of objects in the scene increases. In addition, as it is a template matching method, the template size and shape need to be pre-defined.

We compare our method against FGE [49] and a manually tuned linear shift algorithm (described in Section 4.4).

Table 3.1 shows a comparison of the methods used and resources needed for the work described in this section.

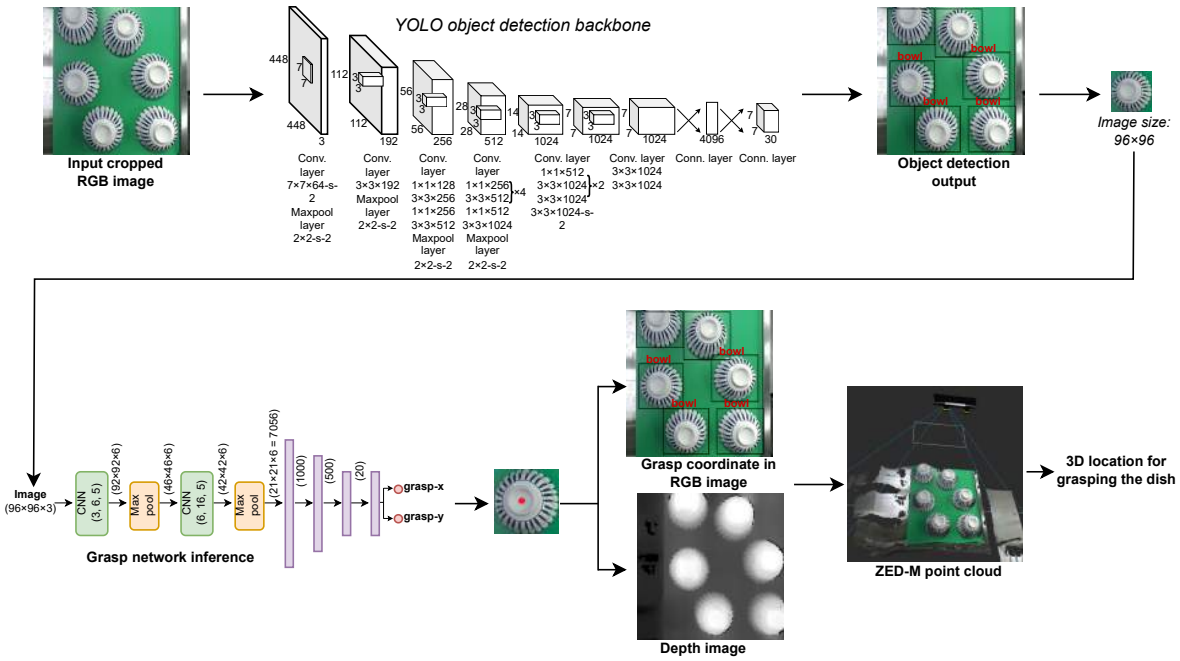


Figure 3.3: Overall workflow.

3.3 Proposed Method

3.3.1 Problems with Single Suction Gripper

Our method focuses on a single suction gripper for symmetric objects, as it has fewer difficulties compared with a parallel-plate robotic gripper. First, a suction gripper can grab only relatively flat surfaces; any surface irregularities (e.g., unevenness or bumpiness) can cause the suction to fail. Second, when using a single suction gripper, the grasping point prediction needs to be very accurate; otherwise, the next step, which is the planning algorithm, will not be able to place the objects in the correct position.

3.3.2 Shallow Networks

Differences from Previous Uses of Shallow Networks

Although shallow networks have been utilized in previous methods, our method has a few key differences. In [41] and [52] shallow networks are used to predict high-ranked candidate grasp rectangles, followed by a deeper network that chooses the optimal grasp points. These networks are cascaded and need to be trained together. The grasp accuracy of the final network relies heavily on the candidates chosen by the preceding network.

In our method, we train the grasp network separately from the object detection network, as they are essentially solving two different problems. In addition, our system does not rely on post-processing or heuristics but rather predicts the grasp position directly from the cropped image.

Advantages of Shallow Networks

Connecting a grasp detector sequentially to an object detector instead of combining the two networks provides us with the following advantages:

- We can train the networks separately for their specific tasks.

- In case of bugs either in object detection or grasp detection, we can separate the modules easily to find the cause of the bug.
- As this is a modular system, we can easily replace the grasp or object detection networks when needed.

A major advantage of having a shallow network is the short training time. With a total of 120,000 images (10,000 images for 12 classes) of size 96×96 pixels, the training time is approximately two hours on an NVIDIA V100 GPU.

Furthermore, as this system is deployed in the real world, we have limited GPU memory during inference. We use a GTX 1080 GPU on which we run three instances of object detection and grasp pose estimation networks for three cameras. Running large deep networks on such a limited system would be impractical. With these settings, we get close to 15 FPS on each instance.

The main reason we chose a shallow network was to maintain the speed of our current operation. Using a larger network would have slowed down the detection, and this would slow down the robotic manipulation speed and eventually slow down the operation of the whole system.

3.3.3 Workflow

As shown in Fig. 3.3, we first pass a cropped RGB image to the YOLOv3 [30] object detector for detecting the dishes. Once a dish is detected, we crop the image and extract only the relevant object. Next, this cropped image is resized to 96×96 and passed to the grasp detector network. Once the grasp position is detected, we calculate its position in the original input image. The grasp coordinates in pixels are then used to find the corresponding 3D point from the point cloud. Then, we approach this 3D position with the robot to pick up the dish.

We use YOLOv3 for object detection and retrain it for our custom objects. We chose this model for its speed of detection and ability to generalize from being trained on synthetic images to inference on real images [30].

We use an object detection model because the dishes need to be classified for sorting and placement. Classifying dishes is a non-trivial task, as the background of the dishes, lighting conditions, and other environmental noises can drastically change in a real-world environment. In addition, the model can be used to detect only the objects that the suction gripper can pick up. Fig. 3.4 shows some sample input scenarios for object detection.

3.3.4 Network Architecture

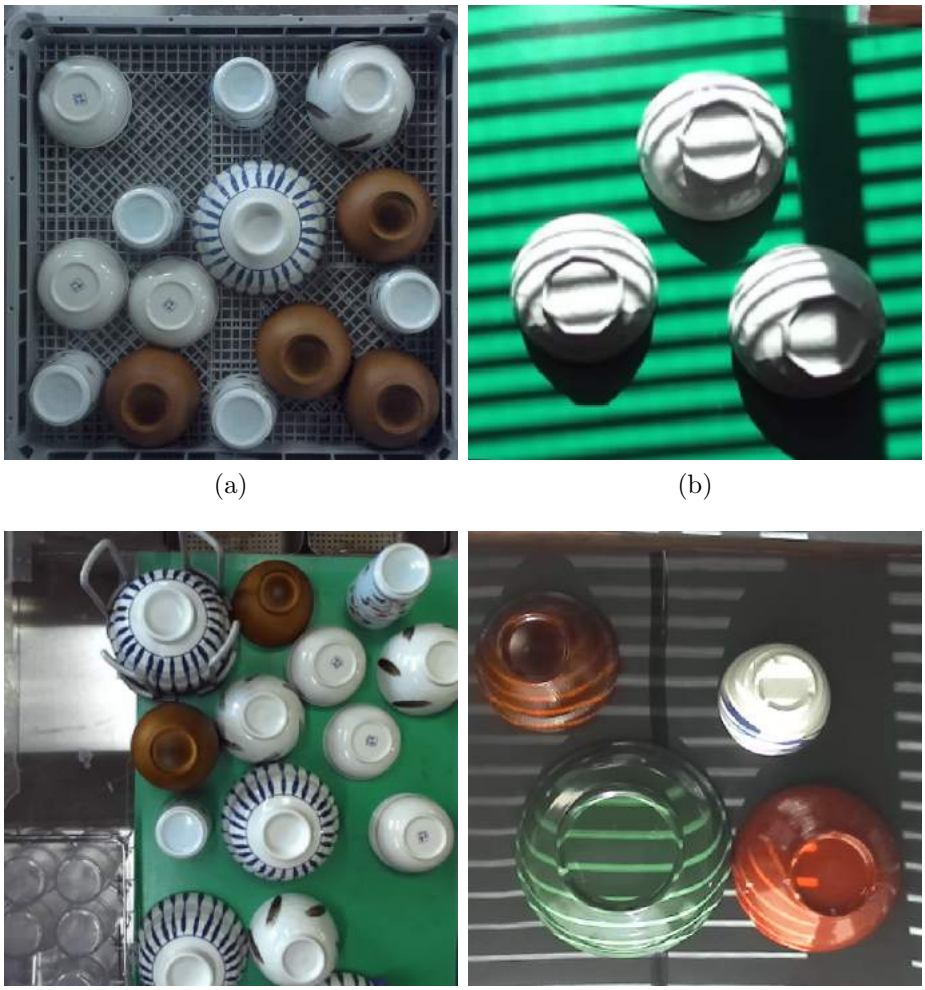
The shallow network used for this task is shown in Fig. 3.5. We treat the problem of finding the grasping point as a regression problem. The input image size is 96×96 pixels, and we use all three channels to avoid losing any information. We have two convolutional layers followed by four fully connected layers to bring down the activations progressively. The output of the final layer is the x and y coordinate of the grasping point.

We make use of the SmoothL1Loss function as it is less sensitive to outliers than MSE loss [53], as shown in following equation:

$$loss(x, y) = \frac{1}{n} \sum_{i=1}^{n} z_i, \tag{3.1}$$

where z_i is given by

$$z_i = \begin{cases} 0.5(x_i - y_i)^2/\beta, & \text{if } |x_i - y_i| < \beta \\ |x_i - y_i| - 0.5 * \beta, & otherwise. \end{cases} \tag{3.2}$$



(a) (b)

(c) (d)

Figure 3.4: Input to object detection.

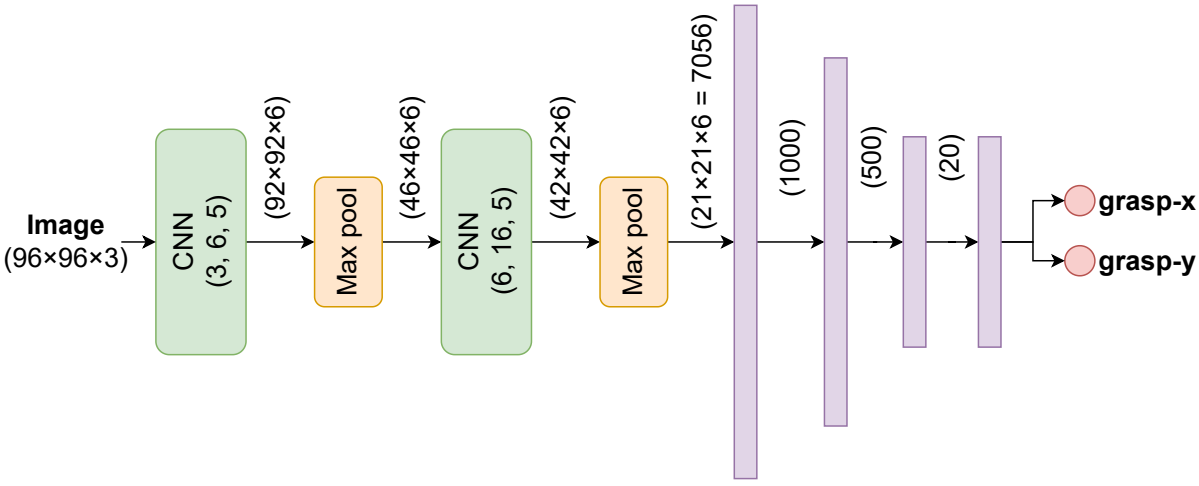


Figure 3.5: Shallow convolutional network for grasp detection.

3.3.5 Synthetic Data Generation

In this subsection, we describe how we collected our data. Collecting and labeling data for convolutional networks can be challenging because such a large amount of data is required to train the model effectively. Lack of labeled training data is often cited as a major problem when

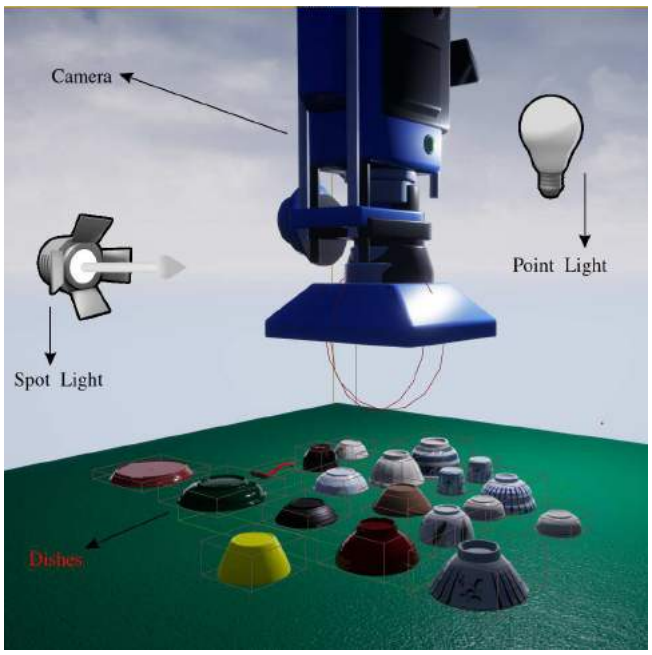


Figure 3.6: Unreal Engine 4 environment.

training convolutional networks [54]. When collecting data manually, several of the following problems can occur.

- **Time-consuming:** Collecting and labeling images is labor-intensive, time-consuming, and expensive.
- **Inaccurate:** Annotations can be mislabeled due to human errors.
- **Inconsistent:** There may be inconsistent results when shared between multiple workers.

To address these issues, we decided to generate our data synthetically. Over the past several years, there have been several approaches to creating synthetic data for computer vision tasks [55, 32, 56].

We use the Unreal Engine 4 gaming engine and NVIDIA Deep learning Dataset Synthesizer (NDDS) [32] plugin to simulate a realistic environment for our setup and generate our synthetic data. The NDDS tool enables us to move the dishes randomly in the scene. It also comes with a random background generator for domain randomization that helps in making the model robust to any background not seen during training. Domain randomization has previously been used to bridge the reality gap of models trained using only synthetic data [1] and to increase the accuracy of object detection for indoor objects [57].

Using the NDDS tool, we randomized object pose, object orientation in the x and y axes, scene lighting, light sources, and background. Our simulation setup in Unreal Engine is shown in Fig. 3.6. We made a couple of improvements to the tool to generate data for our objects.

1. We restricted the object's rotation in each axis to be controlled by the user as rotation around the z-axis is unfavorable for symmetric objects [58].
2. We also attached a "socket" to our dishes, which moves along with them. As this socket has a bounding box too, we use it to denote the grasp position. This socket is made invisible during data collection so that it is not seen in the RGB images.

- Optimized 3D model: 3D model with reduced number of polygons which can still represent the shape of the object adequately.

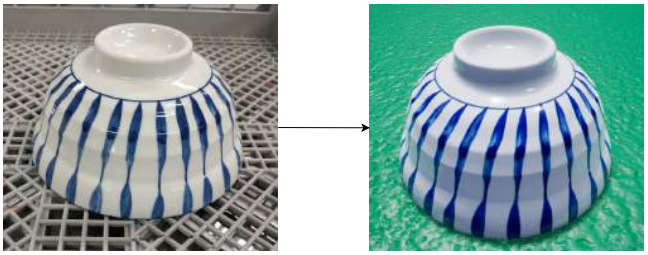
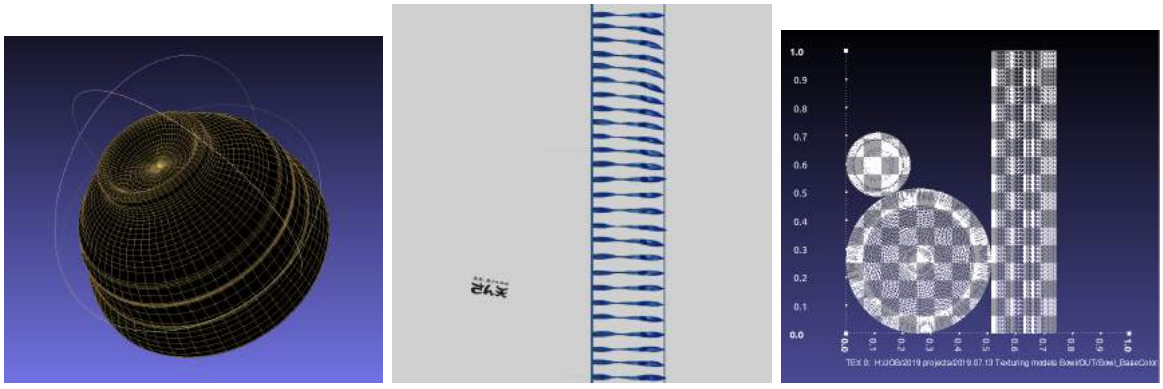


Figure 3.7: Real dish to 3D model.



(a) 3D model. (b) Base color. (c) UV map.

(d) Normal map. (e) Roughness map. (f) Metallic map.

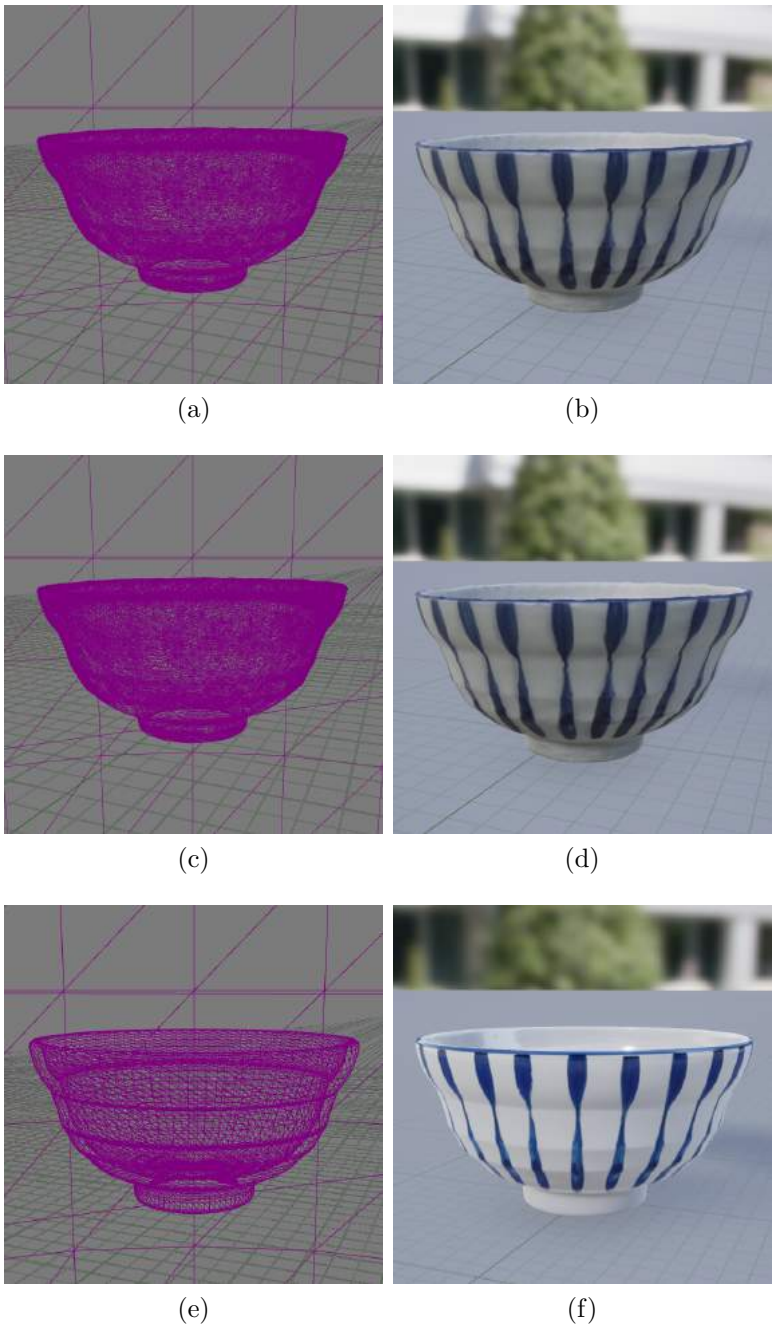
Figure 3.8: Real dish to 3D model process.

- UV map: Projection of 2D image to a 3D model surface for texture mapping.
- Base color: Texture for mapping on the 3D model surface.
- Normal map: Texture map for faking the lighting of bumps and dents.
- Roughness map: Defines how light scatters across the surface of the 3D model.
- Metallic map: Black and white texture that acts as a mask that defines areas on a texture set or material that behave like a metal and do not.

It is also possible to obtain the above data directly from a 3D scanner, but the results are not as good as a remodeled 3D model as seen in Fig. 3.9.

A sample of raw synthetic data is provided in Fig. 3.10. We add other kitchen objects (e.g., spoons) and noises to make the model robust against adverse changes in the environment.

Ortery 3D MFP scanner. Size: 14 MB Optimized model for UE4. : ¡ 1MB

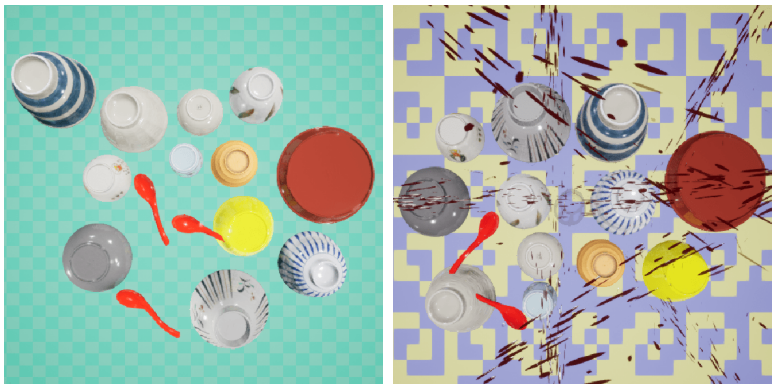


(a) (b) (c) (d) (e) (f)

Figure 3.9: Scanned vs designed 3D model.

Figure 3.11 shows the socket we attached to the top of the dish from where it is supposed to be picked up. This socket moves along with the dish as the dish itself moves randomly in the scene. As the socket's 3D position and 2D bounding box are also exported along with the dish, we can easily extract its location. Figure 3.12 shows how the 6D location of the dish and the grasp position are converted to the required bounding boxes for both. We then convert the bounding box of the grasp position to a single point which is shown in Fig. 3.14.

In total, we generated 10,000 images for each class. We further collected 100 real images for each class and used them as a test set. For testing the accuracy of our training, we made squares of fixed sizes with centers at the ground-truth grasp position and the predicted grasp position. The IoU metric: $IoU = \frac{AreaofOverlap}{AreaofUnion}$ was used for the evaluation.



(a) (b)

Figure 3.10: Noises in synthetic data.

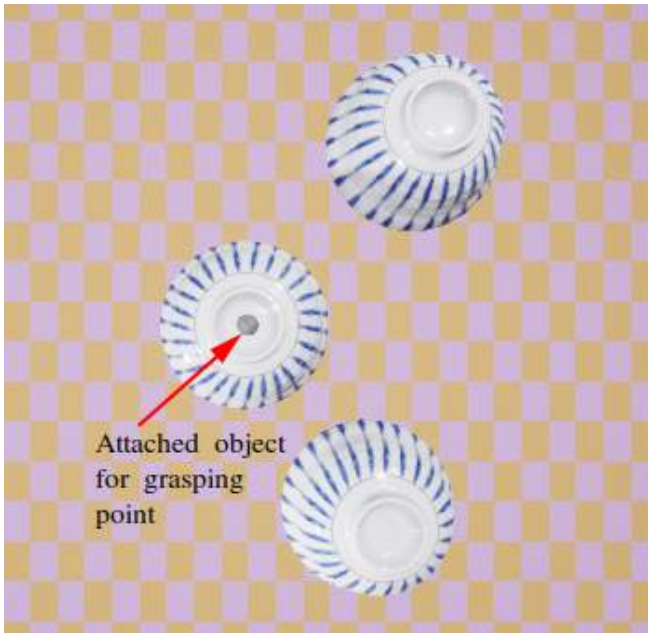


Figure 3.11: Socket attached to meshes in Unreal Engine 4.

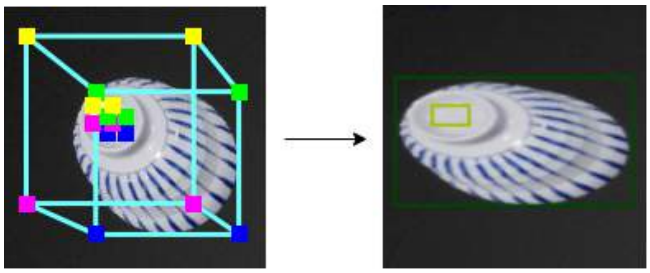


Figure 3.12: From NDDS format to YOLO format.

3.4 Experiments

In this section, we describe the setup and conditions for our experiments and then report the results.

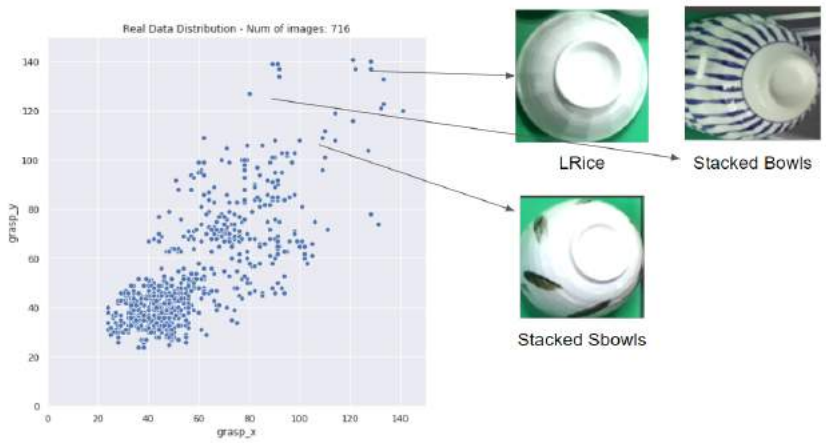


Figure 3.13: Real data distribution.

Figure 3.14: Synthetic data sample.

3.4.1 Experimental Setup

To test the above methods impartially, we used a robot to pick up the dishes based on the predicted grasp positions by each method. Pick-up pass or fail was determined as demonstrated in Fig. 3.15.

We conducted our test on 12 classes of dishes labeled (a)-(l) in Fig. 2.20. Dishes were selected randomly from the commercial restaurant dishes available to us. We further evaluated our methods on a new set of dishes shown in Fig. 3.16 without fine-tuning or retraining the model to test their ability to adapt to new dishes.

The above methods were selected because they had low computational cost and could be integrated into our existing processing system, which consists of an industrial PC with a Xeon E-2176G processor and GTX 1080 GPU.

3.4.2 Baselines

We compared our method against FGE [49] (described in section 3.2.2) and the linear shift grasp position predictor (described in section 3.4.3).

Figure 3.15: Pick-up pass vs. fail.

(a) (b) (c) (d) (e) (f)

Figure 3.16: New dish set.

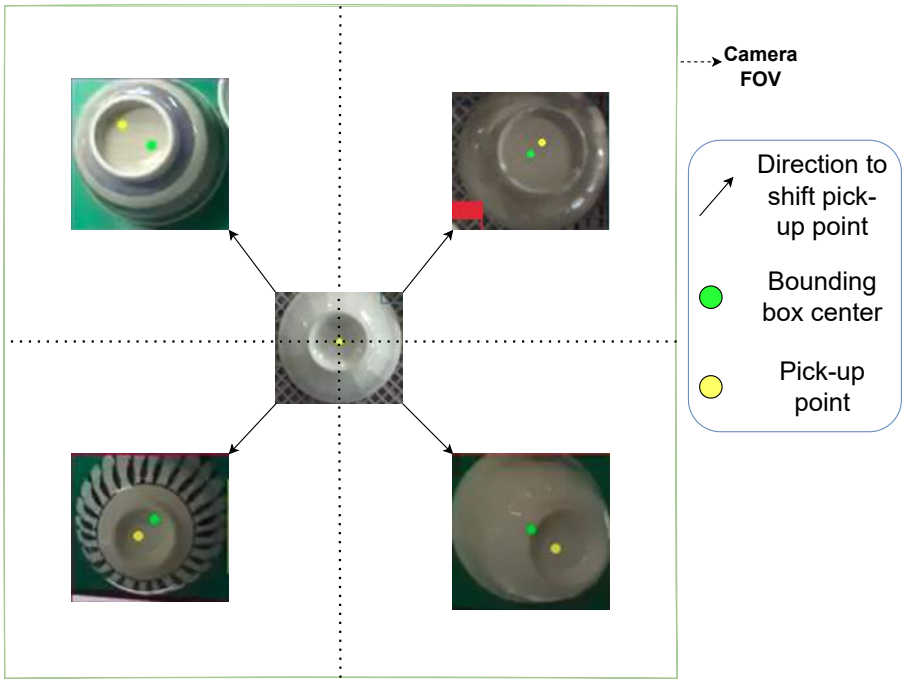


Figure 3.17: Linear shift grasp position predictor.

3.4.3 Linear Shift Grasp Position Predictor

In this method, we predict the grasp position based on the location of the object's bounding box in the image and the center of this bounding box. The algorithm utilized for this is shown in Fig. 3.17. The bounding box is predicted by the object detector explained in section 3.3.3. We first divide the image into four quadrants and make the following assumptions:

Table 3.2: Results of pick-up test.

	Total no. of dishes	**Pickup pass**	**Pickup fail**	**Pass percentage**
Linear Shift Algorithm	133	133	0	100%
FGE	68	28	40	41.17%
Grasp Detector (Ours)	133	129	4	96.99%

1. The objects are symmetric.

2. The bounding box of the object detector fits exactly around the dish.

3. The height of the dish does not vary too much.

Then, we determine the distance of the bounding box center with respect to the center of the image by

$$dist = |\frac{SizeofImage}{2} - bboxCenter|, \tag{3.3}$$

where *bboxCenter* denotes the center of the bounding box.

As shown in Fig. 3.17, we then shift the center of the bounding box linearly in the direction based on its quadrant to predict the grasp position,

$$GraspPoint = bboxCenter - (dist * const) \tag{3.4}$$

In Eq. 3.4, *const* denotes the constant for each dish, which needs to be fine-tuned manually since the size and shape of each dish varies considerably.

Although the manual fine-tuning for each dish is time-consuming, we can achieve high grasp accuracy with this method. However, if there is a change in the camera parameters, such as resolution or distance from the pick-up area, the fine-tuning process needs to be repeated. Automatic fine-tuning of this algorithm can be challenging due to the varying heights of the dishes or in cases where the dishes are stacked on top of each other.

3.4.4 Results

The results of the pick-up test are shown in Table 3.2. For the linear shift grasp detector and the shallow network grasp detector, we used 133 dishes, each of which included all the dishes separately for each class and a mixed set of dishes. For FGE, we had to reduce the number of dishes because the predicted grasp positions were highly inaccurate when the dishes were placed close to each other. The linear shift algorithm serves as a baseline for our tests, as it is highly accurate due to manual fine-tuning.

We found that FGE was not suitable for this task. Figure 3.18 (b)-(f) shows cases where FGE fails to predict the grasp position. As FGE uses only the cropped depth image, it is susceptible to any noise in the depth image and thus fails to predict the grasp position inside the pick-up area, as seen in Fig. **??** and **??**. Also, for dishes with sharp ridges, as seen in Fig. **??** and **??**, the pick-up fails because the predicted grasp position is too close to the edge.

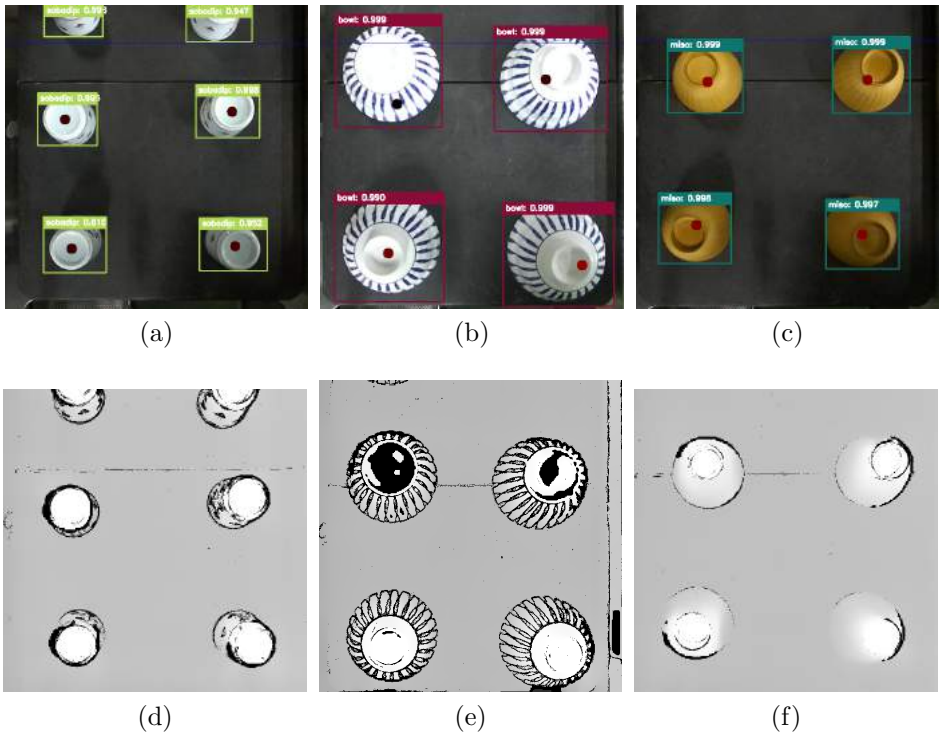


Figure 3.18: Fail and success cases for FGE.

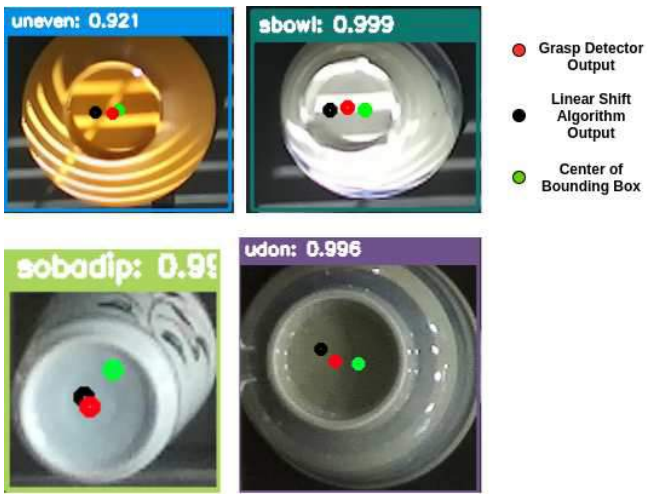


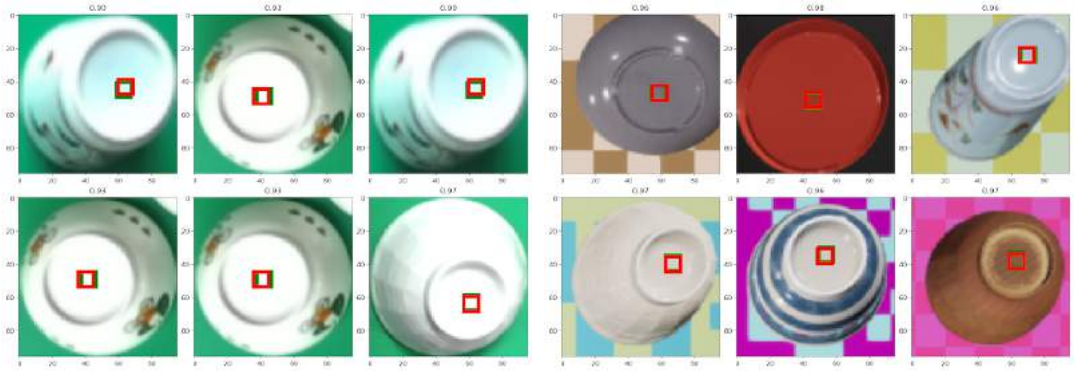
Figure 3.19: Grasp detection output samples.

Results on New Dishes

The grasp detector can also generalize to new dishes without having to retrain the model. To demonstrate this, we conducted pick-up tests on a new set of dishes (shown in Fig. 3.16) without fine-tuning the constant for the linear shift algorithm and without retraining the grasp detector model.

As we can see in Table 3.3, the pick-up accuracy of the linear shift algorithm drops, as the shapes of the dishes are slightly changed. In contrast, the pick-up accuracy of the grasp detector is maintained in fact, it exhibits a slight increase due to the presence of texture on the replacement dishes.

Figure 3.19 shows examples of grasp position prediction by the linear shift algorithm and the grasp detector. As we can see, the grasp detector performs remarkably well in challenging lighting conditions even though it has been trained only on synthetic images. Also, on the



(a) Real test data. (b) Synthetic test data.

Figure 3.20: Test result on real and synthetic images.

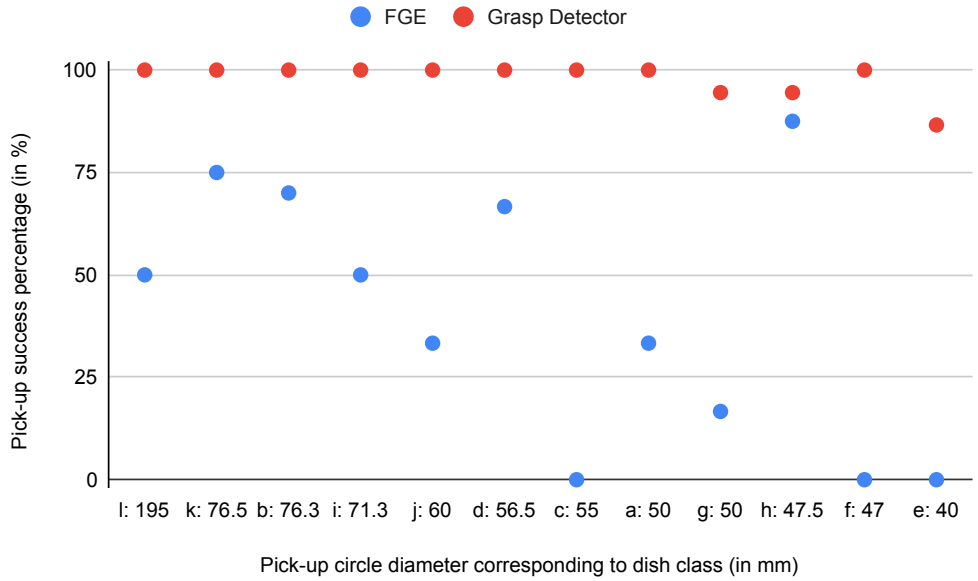


Figure 3.21: Pick-up success vs. pick-up area diameter.

Table 3.3: Results of pick-up test with new dishes.

	Total no. of dishes	**Pickup pass**	**Pickup fail**	**Pass percentage**
Linear Shift Algorithm	192	159	33	82.81
Grasp Detector (Ours)	192	190	2	98.95

original set of dishes, we can see the grasp detector's accuracy is slightly higher than that of the manually tuned method.

As we can see from the above results, the linear shift grasp detector can be highly accurate when its initial assumptions are unchanged. Since it is possible to fine-tune the constant for each dish, we can iteratively improve the accuracy. The shallow network grasp detector performs as well as the linear shift method without the need of manual fine-tuning for each dish. It also generalizes to new dishes unseen during training, making it suitable to adapt to new dishes quickly.

Also, if we need to change the position of where to grab the dish, we can easily adjust this by changing the socket's position during data collection. We can also adapt the network to a parallel-plate robotic gripper or hand-like grippers.

Further Evaluation of FGE

We further evaluated the performance of FGE by comparing its pick-up success rate to the diameter of the pick-up area at the bottom of each dish among our classes. As shown in Fig. 3.21, the accuracy of FGE decreases with the decrease in the diameter of the pick-up area. In contrast, the shallow network grasp detector is largely unaffected by changes in the diameter of the pick-up area.

For FGE, there are some exceptions, notably for class (h), which has a high success rate for pick-up despite a small diameter. We can see in Fig. **??** and **??** that this is due to the shape of the dish, as only the top surface of the (h) class dish is visible from the top-view, unlike other dishes whose shapes are more curved.

From the above findings, we conclude that FGE is susceptible to noise in the depth image, the shape of the dish, and the diameter of the pick-up area, which makes it unsuitable for the task of picking up dishes with a single suction gripper.

3.5 Conclusion

In this paper, we proposed a grasp detection method for the dishwasher in a commercial kitchen environment. The proposed method consists of a shallow network that makes it effective with limited computational resources while ensuring accuracy and low computational cost. We also developed a synthetic environment to collect training data more efficiently. The simulation environment is made in the UE4 gaming engine using accurate 3D models.

Our experimental results in a commercial kitchen environment demonstrate that the proposed shallow network grasp detector trained with only synthetic data can predict the grasp position accurately. It is easy to configure, easy to debug, and requires fewer resources compared to conventional methods.

As future work, we will improve our grasp detector by combining it with FGE to provide the 3D coordinates of grasp position only when there are no holes in the point cloud. We will also combine it with a linear shift grasp detector to provide a fail-safe when the predictions from the shallow network are inaccurate.

Chapter 4

Automatic Data Collection for Object Detection and Grasp-position Estimation with Mobile Robots and Invisible Markers

4.1 Introduction

Computer vision methods using deep convolutional networks have been influential in solving some of the major problems in robotic manipulation with object detection [59, 30, 60], grasp-position-estimation [61, 42, 54] and six-dimensional (6D) position estimation [62, 63, 64]. These supervised learning methods rely on reliable and high-quality data [54, 65].

Data collection is a crucial process for building accurate models that can perform robustly in real environments. However, manual data collection is a time-consuming and expensive process.

The data-collection process for object detection involves two steps. The first is image collection and the second is annotation. In the image-collection step, one needs to ensure that there is a large variation in the position and orientation of the object in the images. Objects placed in the same position in every image can make the model over-fit to the particular position and orientation and less robust when it is placed at a different location. In the annotation step, the annotator marks the bounding box of the object in the image to specify the location of the object in the image. This process can be highly laborious depending on the size and shape of the object, the number of objects in the image, and tolerance in annotation accuracy. The annotators need to have the domain knowledge of the objects they are annotating, in terms of how to distinguish between similar-looking objects. Similarly for grasp-position annotation, one needs to have the domain knowledge of the appropriate grasp locations for each object and knowledge of which end-effectors would be used for grasping.

Certain approaches use outsourcing to tackle this task in which people can be hired to annotate image datasets for a fee [66, 67, 68, 69]. This gives rise to another problem, which is quality check. When annotation is outsourced, several annotators of varying skill levels are employed. Because the hired annotators have varying levels of skill and can annotate the same objects in different ways, these annotated datasets must be quality checked. Another issue is that confidential data cannot be outsourced to such services for annotation [70]. For accurate object detection, it is also important that the training data be representative of the real-world data the model might encounter, which makes it necessary to add similar noise in the training data as well.

To solve these problems, we propose the automation of the data collection and annotation process using 1) mobile robots equipped with novel tilt platforms and 2) invisible markers.

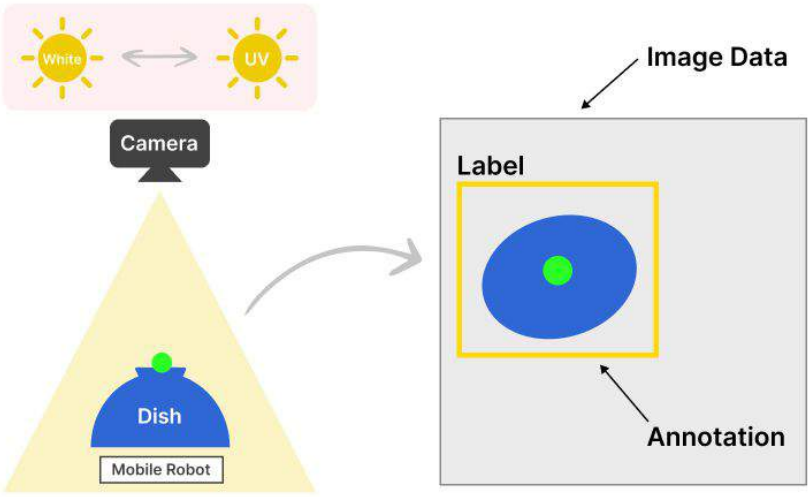


Figure 4.1: Automatic data collection for object detection and grasp-position estimation with mobile robots and invisible markers.

Mobile robots transport objects from one location to another. Invisible markers visible only under ultraviolet light are used to mark the grasp position of the objects. The concept of our data-collection method is shown in Fig. 4.1.

Object grasping is a crucial step in robotic manipulation in various applications. It entails 1) identifying an object in a given image and 2) determining the position from where the robot can dependably grab the object. Objects are of various shapes and sizes in a general grasp estimation problem. Additionally, the scene is typically cluttered and is shot from an arbitrary viewpoint. A successful grasp depends on correctly identifying the object and choosing the proper position to grab it from, depending on the type of end-effector. In an industrial setting, the camera is usually fixed in a top-down view above the objects of interest. The types of objects are limited, and an appropriate end-effector is chosen, which can grab the objects reliably. The scene can also be constrained to increase the rate of successful grasps. For example, to ensure better detection, one might regulate the lighting conditions and only permit objects with specific shapes, like circles. To ensure that the end-effector can reliably grasp the objects, one can also restrict the size and weight of the objects.

We focus on data collection for the object-detection and grasp-position-estimation tasks for a commercial dishwasher system [6]. In our vision system, the camera is fixed in a top-down view above the objects. This is a conventional setup in industrial applications.

Our setup of a commercial dishwasher system is shown in Fig. 4.2. This system has two main objectives: 1) to process numerous dishes in a short amount of time to prevent any bottlenecks in restaurant operation and 2) to pack the dishwasher rack efficiently to maximize the number of dishes washed in each cycle. Our scene is less cluttered than a regular bin picking problem as it is a commercial human-robot collaborative dishwasher system used to automate dishwashing in restaurants. Maintaining close to real-time constraints in a commercial system is essential to avoid ambiguity and process failure. To achieve the above goals, the dishes are placed upside down by human assistants and the dishes are picked up from the center so as to pack the rack efficiently. In addition, since we use a single suction gripper in our application, the bottom center of the dish is the only position where the suction can work without failing.

The symmetric circular shape of the dish allows for a simpler 2D bin-packing method which is out of the scope of this paper. The dishes need to be classified for sorting after dishwashing and to differentiate which dish needs to be washed with a brush and which one needs to be

(a) Dishwasher system robot setup.

(b) Pick and pre-wash dishes before placing them in the rack.

(c) Sort and place the washed dishes.

Figure 4.2: Commercial dishwasher system [6].

washed with water only. Although our target objects in this work have symmetric shape, we show how this method can be used for asymmetric objects as well.

In Section 4.2 we describe related work and briefly compare them with ours. In Section 4.3 we give details of the proposed method, data-collection setup along with hardware used, and invisible markers to annotate grasp position. In Section 4.4, we describe the experimental setup of comparing the data-collection methods and the results. Finally, we provide our conclusion based on the tests. To the best of our knowledge, our proposed method of using mobile robots and invisible markers has not been applied before to collect data for object detection and grasp-position estimation.

4.2 Related Work

Manual data collection and annotation have several problems, i.e., time-consuming, expensive, requires skill, and can contain human errors. To overcome these problems, there are several methods of automatically collecting and annotating image datasets. We briefly describe some of them in the following subsections.

4.2.1 Data Collection

Methods Using Robot Arms

Certain methods use manipulator robot arms to automate the process of data collection [71, 72, 73]. The manipulator robot arms are used to change the camera position and angle to acquire images of the objects from various perspectives. The objects are stationary throughout the data collection process.

Although the robot arms help in providing diverse perspectives and a consistent dataset compared with manual methods, they are limited by the reach of the robot arms. These methods also require human supervision and are not fully autonomous as the robot arm trajectories can encounter singularities and joint limits while moving in their environment which need to be resolved.

The high cost of a manipulator robot can also be detrimental when used for data collection. Fang et al. and De Gregorio et al. used a Universal Robots UR5 robot arm with an average cost of around 30,000 USD [73, 71]. Rennie et al. used a Motoman SDA10F robot, which can cost upto 100,000 USD.

Synthetic-data-generation Methods

There are methods that use accurately textured 3D models of the objects to generate a labeled synthetic dataset [74, 1, 75, 76]. These methods use physics simulation environments to change the object position and orientation. The background can also be easily modified and one can use domain randomization to make the model robust against different backgrounds during inference [1].

These methods can be especially helpful for annotating the 6D pose of an object, which is much more complicated compared with 2D annotation. For 6D annotation, one needs to 1) draw a 3D bounding box around the object and 2) indicate the angle of orientation with a normal vector with respect to the camera plane.

Synthetic-data-generation methods require highly accurate 3D models of the objects for high-fidelity data. The drawback of using synthetic data is that the models trained using only synthetic data do not perform well during inference when tested on real data [77]. Poorly constructed 3D models can lead to further increasing the Sim2Real gap. Synthetically generated data have been known to increase detection accuracy when combined with a sparse real dataset [78].

4.2.2 Annotation

Bounding-box Annotation

Deep learning methods and pre-trained object detectors have been used to reduce the annotation time of bounding boxes [79, 80, 81]. These methods use object detectors pre-trained on a broad variety of data and later fine-tuned with manually collected data. One major drawback

with such methods is that one needs to train an object-detector model first that would be able to accurately annotate the new data.

Certain methods annotate the first image manually then, using a well-calibrated camera use camera poses to project the bounding boxes on the rest of the images captured in that sequence [71, 73]. De Gregorio et al. used an augmented reality pen to manually outline virtual boxes around the target objects for the first image [71]. The objects in the scene need to be stationary and the camera moved using a robot arm with these methods.

Grasp-position Annotation

Annotating the grasp position has been tackled using synthetic-data-generation methods [82, 83]. However, as mentioned above, the visual domain gap can lead to low grasping accuracy when inferring on real images.

Other methods combine real-world images with computation in synthetic environments [73, 84]. Fang et al. used high-quality mesh models of objects and downsampled them to obtain a large number of grasp candidates. These candidates are then filtered to fit a parallel plate gripper so there would be no collision and no empty grasp.

Although this method generates a large number of possible grasp positions for an object, it does not account for planning the placement once it has been picked up. This is essential in an industrial setting, where packing as many objects as possible in a small space is crucial.

Other methods use invisible markers that are visible under ultraviolet (UV) light and invisible under regular (white) light [85, 86]. Takahashi et al. used invisible markers for annotating segmentation masks of deformable objects [86].

4.3 Automatic Mobile Robot Data Collection Setup

In this section we describe the system setup of our proposed data-collection method.

4.3.1 Proposed Method

In a fixed camera setting, the view of the object can change considerably as its position with respect to the camera changes. For example, a spherical object, semi-spherical object, and inverted semi-spherical object all appear as a circle when the objects' centers align with the principal axis of the camera. However, the appearances differ greatly when they are at the edge of the camera's field of view, i.e., away from the camera center. It can be enormously expensive to manually capture images of objects for every possible position under the camera's field of view.

Our method enables data collection and annotation for both object-detection and grasp-position-estimation tasks by using mobile robots and invisible markers. There have not been any methods that have explored the use of mobile robots for data collection.

Using small mobile robots enables us to automate moving the objects and capturing the object's image at every possible location inside the camera's field of view. Our approach allows greater control over the objects, which in turn allows greater control over the scene. This enables us to randomize the scene and fill in any data gaps in a specific configuration.

Invisible markers enable us to automate the annotation process. They are visible only under UV light and invisible under white light. This enables us to extract the invisible marker's position in UV-light-lit images and annotate the white-light-lit images accordingly. The invisible markers are necessary since we do not want the markers to affect the appearance of the dish. Using visible markers can change the appearance of the dishes, and we would need to use

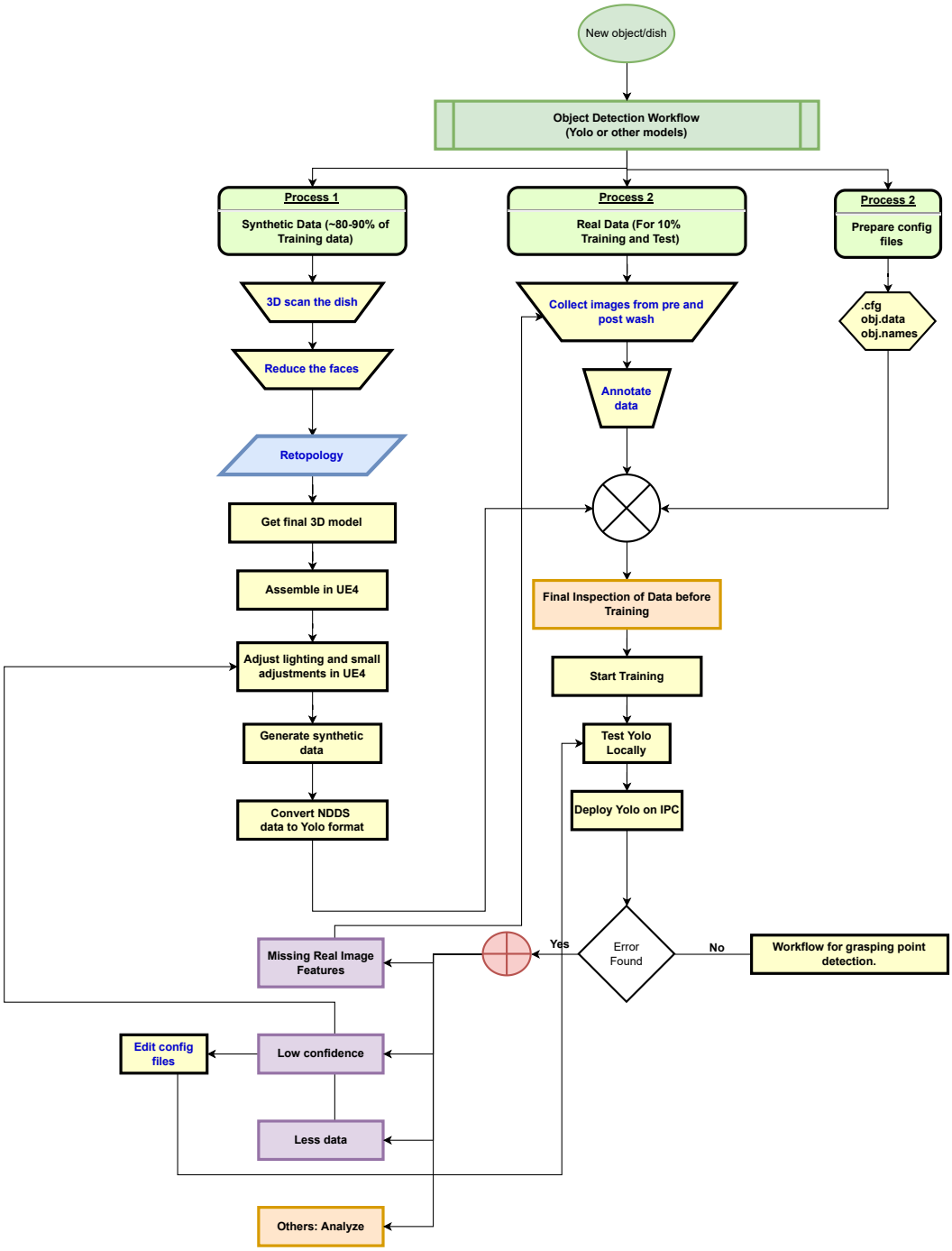


Figure 4.3: Original data collection workflow for a new object.

the markers during inference as well which would not be practical in a commercial dishwasher system.

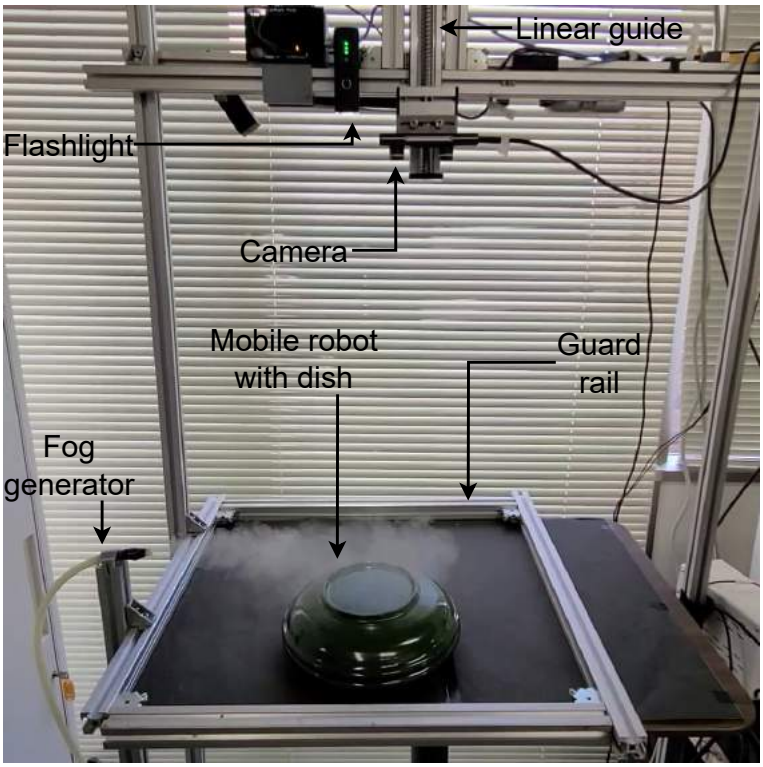
By adding noise during data collection with random lighting and a fog generator, we are able to generate data close to what robots might encounter in the real world. Since our application is a commercial dishwasher system, we set up the environmental noise as seen at the dishwashing station in a real restaurant.

4.3.2 Previous Data-Collection Workflow

In Fig. 4.3, we can see our data collection workflow when combining synthetic data and manually annotated real data. The above workflow represents the iterative process that is typically

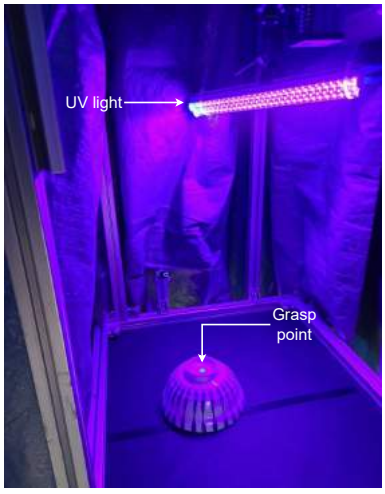
employed for addressing the data-drift and concept-drift problems introduced in Section **??**.

4.3.3 Data-collection Setup



(a) Data collection setup.

(b) White light environment.



(c) UV light environment.

Figure 4.4: Automatic data collection setup: Mobile robot setup with lighting environment for invisible markers.

Our data-collection setup is shown in Fig. 4.4.

Camera and Linear Guide Setup

We use a ZED-M 3D camera, but our method requires only RGB images. The camera has a high capture rate of 30 frames per second. This enables us to capture multiple images of an object under motion while seemingly at the same location.

(a) Khepera robot. (b) Khepera with dish side-view. (c) Khepera with dish top-view.

(d) E-puck2 robot. (e) E-puck2 with dish side-view. (f) E-puck2 with dish top-view.

Figure 4.5: Mobile robots with dishes.

The camera is placed on a linear guide, which enables us to move the camera vertically to change the height between the platform and camera. Moving the camera on the z-axis continuously during data collection enables us to simulate real conditions under which the camera height to the platform can vary.

Mobile Robots

The mobile robots are used to carry objects over a platform under the camera's field of view. We surveyed multiple mobile robots and selected E-puck2 and Khepera IV robots for their capabilities of accurate locomotion and control with Bluetooth or Wifi.

Khepera IV is able to bear a payload of up to 2 kg, which is sufficient to carry stacks of dishes. Although E-puck2 does not have a specific payload capacity, it is capable of carrying objects of weights up to 150 grams.

The size of the robot was also an important factor since we needed the robots to be hidden in the top-down view of the camera. Figure 4.5 shows the mobile robots we used along with the dishes. The robots were programmed to cover every position on the platform. We used guard rails to prevent the mobile robots from falling off the platform.

Invisible Markers and Lighting Setup

We used the same invisible ink mentioned in a previous study [86] to mark the grasp position of the object in our application.

Our lighting environment consisted of USB-powered RGB light-emitting diodes (LEDs) for the white-light environment, and UV LEDs for the UV-light environment. The data-collection platform was covered with tarp to block any external light. Since we used inexpensive UV LEDs, there was a visible violet hue to the UV-light-lit images. However only the invisible-ink-marked point stood out in the images.

We controlled the lighting using a programmable USB hub. This enabled us to switch between white light and UV light at high speeds programmatically for every frame captured with the camera.

(a) Three-axis platform. (b) Three-axis platform with dish. (c) Khepera with three-axis platform.

Figure 4.6: Three-axis platform.

Table 4.1: Three-axis platform specifications.

Elements	**Technical Information**
Microprocessor	Raspberry Pi ZeroW
Language	Python
Communication	802.11 b/g/n wireless LAN
Sensors	MPU9250
Motors	RDS3115MG x3
Tilt angle limit	-15° to + 15°
Power supply	5V

Three-axis Platform

Using only mobile robots would restrict the object's orientation to the XY plane. To overcome this, we designed and built a three-axis platform that can be placed over the mobile robots, as shown in Fig. 4.6. The platform enables us to tilt the objects placed on it in any direction with a maximum tilt of up to 15 degrees.

One caveat of our three-axis platform is that due to its size, it can be placed on top of the Khepera IV robot only. Thus, it restricted our ability to collect data using it for objects which are smaller than this robot. Table 4.1 provides the technical specifications of our three-axis platform.

Environmental Noises

Flashlights were used to add random light noise to simulate reflections and harsh lighting conditions of a real-world environment. We used a fog generator to add smoke noise to the images. This is representative of a real-world dishwasher system, where the steam from the dishwasher can partially occlude the objects under the camera.

The data-collection setup was modular. The components could be replaced or new components added to vary the environmental conditions. Adding the above mentioned noise enabled us to train an object-detection model with realistic data exceptionally robust against real-world conditions. Figure 4.7 shows a sample of our data with environmental noise.

4.3.4 Data-collection and Annotation Workflow

Figure 4.8 shows the workflow of our data collection and annotation process.

(a) (b) (c)

Figure 4.7: Environmental noise.

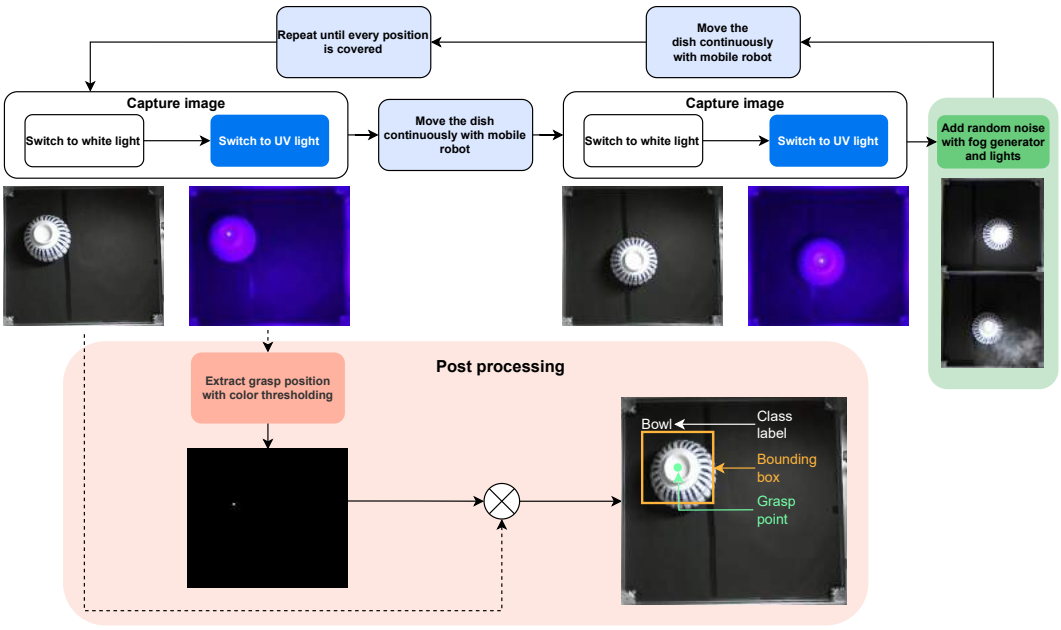


Figure 4.8: Automatic data-collection and annotation workflow.

Pre-setup

We first painted the object's grasp position using an invisible marker. We marked the bottom center of the dish with this marker. The dish was then placed on a mobile robot depending on the size of the dish. We move the robot across the platform inside the camera's field of view.

Image Capture

At every location, we captured two images of the object, one under white light and the other under UV light. This process was very rapid with the help of a high-frame-rate camera capture and lighting environment that changed alternately every frame. This enabled us to capture a total of 2,300 images in around 140 seconds. The 2,300 images consisted of 1,150 pairs of white-light-lit and UV-light-lit images.

Since the speed of the mobile robots was kept low to avoid any collision or damage, and the capture rate of the camera was high, we captured many image-pairs when the object was in the same location. Thus, after deleting any duplicate image-pairs, we obtained around 300 image-pairs in 140 seconds. We choose to filter out the duplicate image-pairs in the post-processing step rather than delaying the image capture because carrying out post-processing is much faster, i.e., it did not increase our cycle-time for data collection, and the time required for the process was negligible.

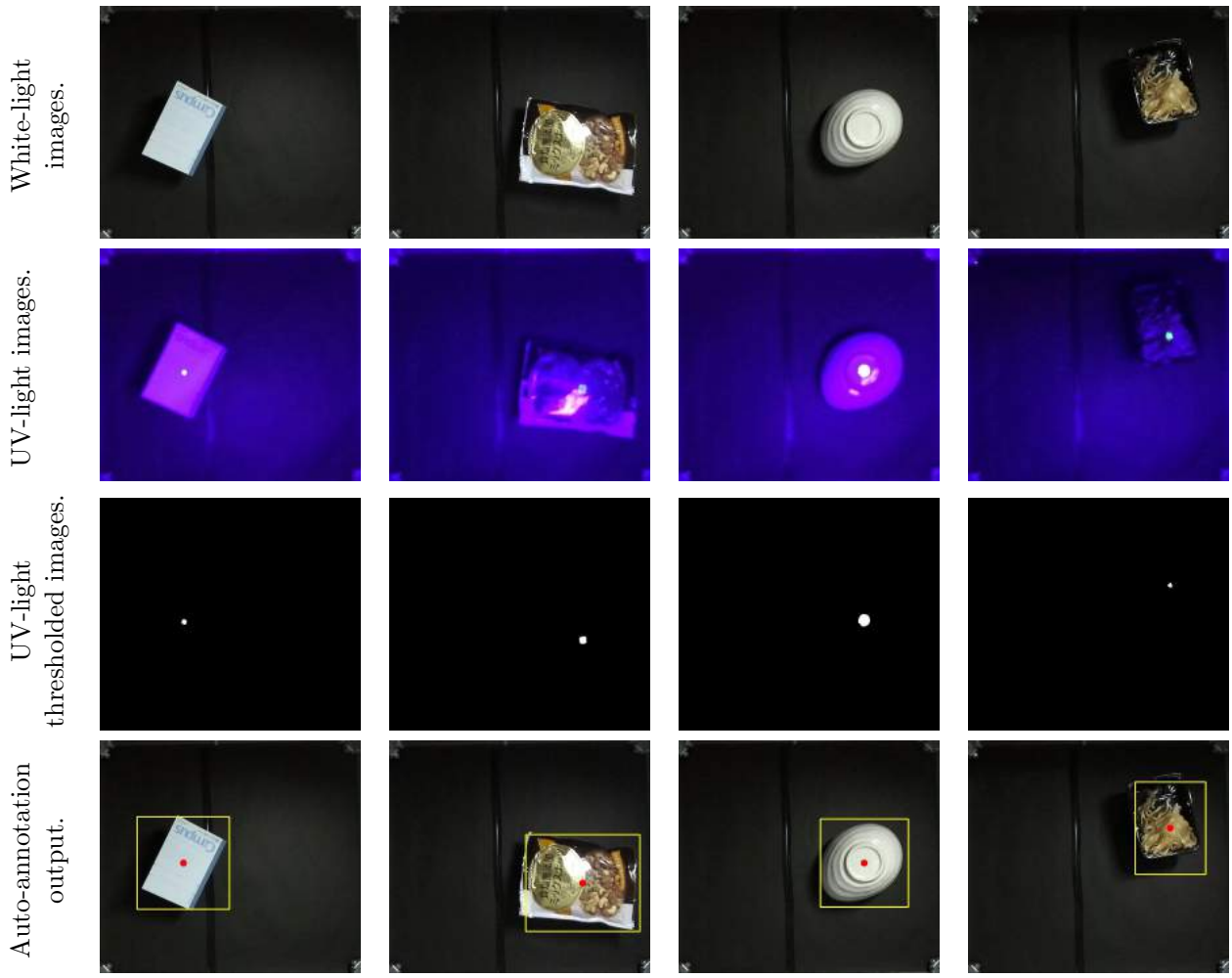


Figure 4.9: Data collection of objects of various shapes.

Grasp-position Annotation and Bounding-box Annotation

The white-light-lit image is the primary image without annotation. The invisible marker clearly stands out under the UV light. Its pixel coordinate is easily obtained by thresholding the RGB value of the color of the marker. This coordinate was used for the grasp-position annotation in our use case.

We then drew a square bounding box with the grasp-position coordinate as the center, by using the following equation:

$$\begin{aligned}(x_{TL},\ y_{TL}) &= (x_c - \mathrm{w}/2),\ (y_c - \mathrm{h}/2) \\ (x_{BR},\ y_{BR}) &= (x_c + \mathrm{w}/2),\ (y_c + \mathrm{h}/2)\end{aligned} \tag{4.1}$$

where $(x_{TL},\ y_{TL})$ and $(x_{BR},\ y_{BR})$ denote the top-left and bottom-right coordinates of the bounding box, respectively, and $(x_c,\ y_c)$ is the center of the bounding box. The *width* and *height* of the bounding box are denoted by $\mathbf{w}$ and $\mathbf{h}$ in the equation, which can be adjusted according to the size of the object. Although the bounding box drawn using the above equation is imprecise and does not fit the object tightly, it performed well to classify the dishes.

4.3.5 Other Applications

Although we considered only symmetric bowls due to the constraints of our target application, our data-collection setup can be used to collect data for asymmetric objects as well, as shown in Fig. 4.9.

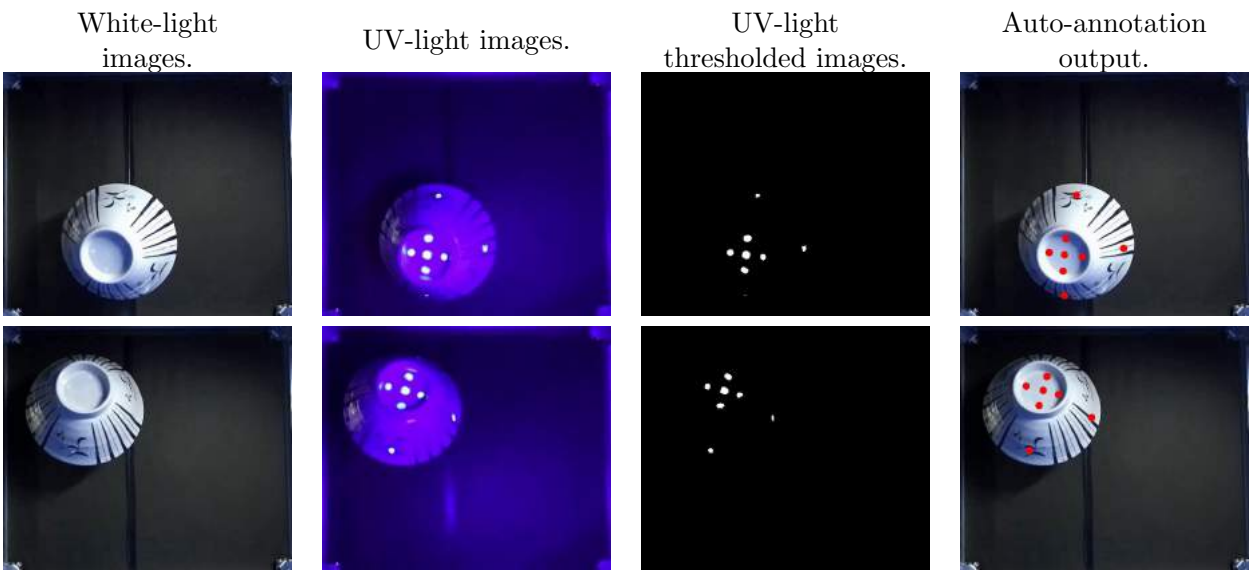


Figure 4.10: Key-point data collection with three-axis platform.

Table 4.2: Comparison of data-collection methods in terms of robot cost and annotation time.

	Robot	Cost (USD)	Annotation	Annotation Method	Time for data collection and annotation of one image
De Gregorio et al. [71]	Universal Robots UR5	30,000	2D/6D	AR pen + Camera projection (Semi-automatic)	Data collection: 7.2 seconds. Annotation: ∼300 seconds for first image in sequence. Negligible time for following images.
Rennie et al. [72]	Motoman SDA10F	100,000	6D	Human annotation (Manual)	Data collection: 5∼10 seconds. Annotation: ∼50 seconds.
Fang et al. [73]	Universal Robots UR5	30,000	6D	Human annotation + Camera projection (Semi-automatic)	Data collection: 5∼10 seconds. Annotation: ∼300 seconds for first image in sequence. Negligible time for following images.
Mobile robot and invisble marker data collection (Proposed)	Khepera IV and E-puck2	5,000	2D	Invisible markers (Automatic)	Data collection + Annotation : 0.4 seconds.

Invisible markers can also be used to annotate key points of an object, as shown in Fig. 4.10. These key points can be used to estimate the orientation of the dish with respect to the camera, which we did not explore in this study.

Table 4.2 provides a comparison of our method against the methods presented in Section 4.2 in terms of cost, annotation and time required for data collection. De Gregorio et al. took around 7.2 seconds to capture an image [71]. We estimated the time required for data-collection by the other two methods [72], and [73] by simulating robot motion and average time for the robot to calculate inverse kinematics and move to a new position for image capture. For estimating the time required for annotating objects with Rennie et al.'s method [72], we used the data provided by Su et al. [69], i.e., an average of 50 seconds for drawing one bounding box per image. Similarly, we extrapolated the time required for 6D annotation with De Gregorio et al.'s method [71] to estimate the time required to annotate images with Fang et al.'s method [73].

4.4 Experiments

To check the efficacy of our data-collection method, we conducted two comparison tests.

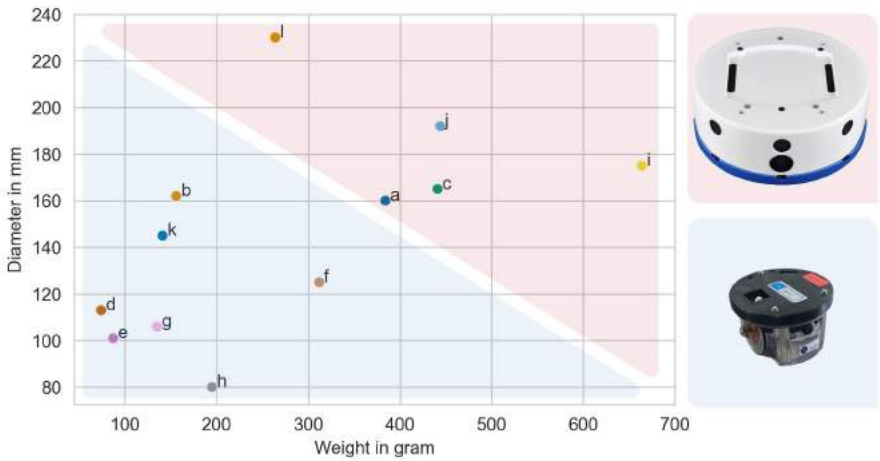


Figure 4.11: Mobile robots assigned to dish based on its dimensions and weight.

We first trained an object-detection model with the data collected with our method. We trained another instance of the same model with synthetically generated data. The test data in both cases were hand-annotated data. We further expanded this comparison by training an instance of the model with both the data collected with our method and synthetically generated data. We compared the mean Intersection over Union (IoU) for all dish classes between the three instances of the model.

We then compared the following three data-collection methods: 1) manual data collection, 2) synthetic data collection, and 3) automated data collection with mobile robots and invisible markers (proposed). The metrics of comparison were procedure of data collection and annotation, cost, and time required.

4.4.1 Object Detection Performance Comparison

The objects for detection were dishes of various classes shown in Fig. 2.20. We randomly selected twelve dishes from the commercial restaurant dishes available to us. Figure 4.11 describes which robot was assigned to each dish according to its dimension and weight.

Data Collected using Mobile Robots and Invisible Markers

For automatic data collection with our method, we used three backgrounds: green, black and dishwasher rack. Figure 4.12 shows a sample of data collected with our method.

We collected around 1,500 images for each dish type with the images distributed over the three backgrounds mentioned above. We were able to collect more than 18,000 images. The total time required to collect and annotate the images was around 2 hours, leaving out the time required to setup the objects and changing the backgrounds, which was only a few minutes.

Synthetically Generated Data

To create synthetically generated data, we first created accurate 3D models with realistic textures of our target dishes using a 3D scanner and hiring a professional digital artist. Figure 4.13 shows one of the 3D models.

We used Unreal Engine 4 (UE4) [37] and NVIDIA Deep learning Dataset Synthesizer (NDDS) [32] tool to generate our synthetic data. Our simulation setup in UE4 is shown in Fig. 4.14.

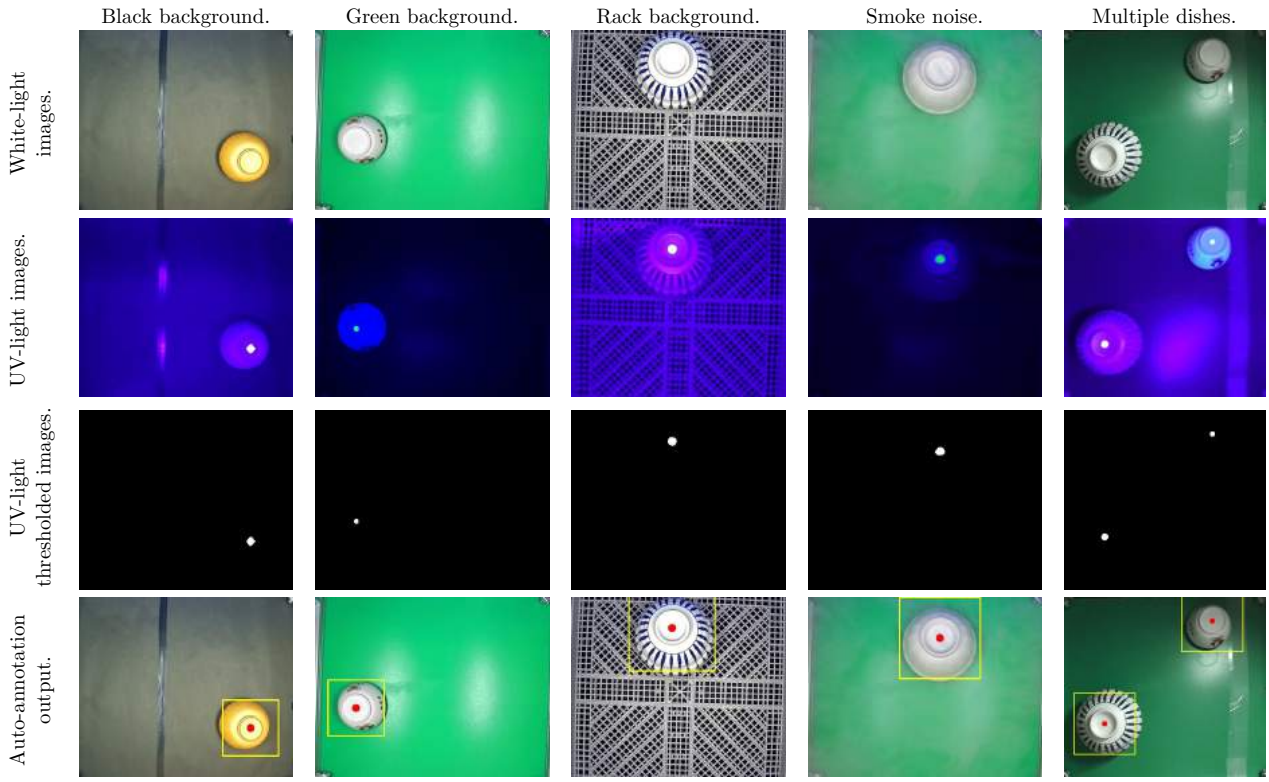


Figure 4.12: Data sample from proposed method.

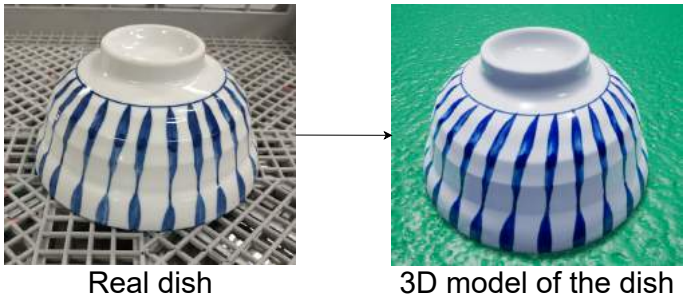


Figure 4.13: Real dish converted to 3D model.

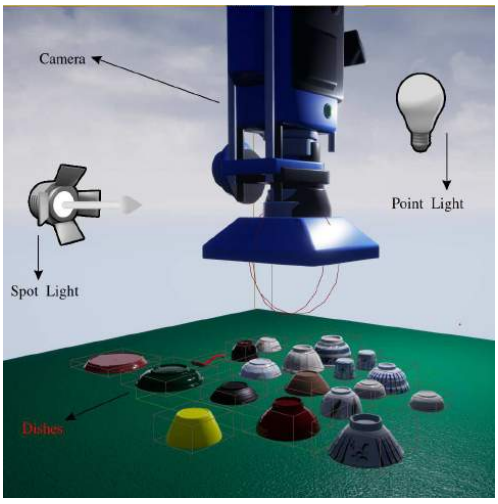


Figure 4.14: Unreal Engine 4 environment.

Figure 4.15: Synthetic data sample.

(a) (b) (c) (d)

Figure 4.16: Test images.

The NDDS tool enabled us to move the dishes randomly in the scene. It also provided a random background generator for domain randomization that helped make the model robust against any background not seen during training. Domain randomization had previously been used to bridge the reality gap of models trained using only synthetic data [1] and increase the accuracy of object detection for indoor objects [57].

We made a few changes to the NDDS tool to restrict the rotation of our symmetric objects in each axis to be controlled by the user as rotation around the z-axis is unfavorable for symmetric objects [58].

We generated synthetic data that had a mixture of images where each dish is present individually and images where all the dishes are mixed together. The individual-dish images contained three dishes of the same type. Five thousand images for each of the 12 dishes were generated amounting to 60,000 images. A further 5,000 mixed-dish images brought the total to 65,000 synthetic images.

The 5,000 mixed-dish images also contained 1,000 images to which we added synthetic noise such as water droplets, smoke, and dirt. Some of these images are shown in Fig. 4.15. A sample of synthetically generated data is shown in Fig. 4.15.

Test Data

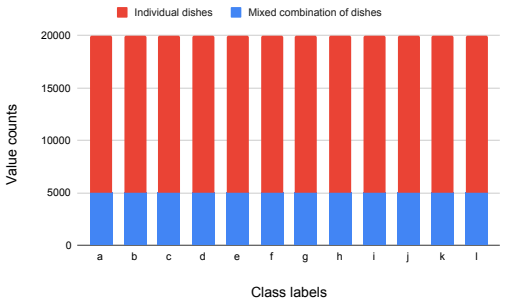
The test data used for all three model instances was a hand-labeled dataset of multiple dishes in various real-world backgrounds. We collected and hand-annotated around 1,500 images containing different combinations of our dish classes. A sample of our test data is shown in Fig. 4.16. The distributions of the training and test data for both models are shown in Fig. 4.17.

Experiment Results

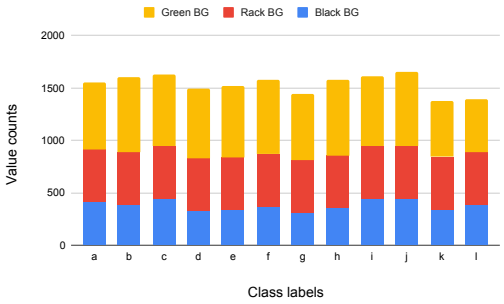
We used the mean IoU metric for comparing the performances of each model instance. The model instance trained with synthetic data only achieved a mean IoU for all dishes of 73.45 %. The model instance trained with the data collected with our method instead achieved a mean IoU for all dishes of 94.33 %. We also ran an ablation test to see how much of an impact the extra noise had, and the mean IoU dropped to only 93.62 %. This was due to a lack of a significant number of edge case images with noise in the test data.

We further combined the two datasets, synthetically generated and the data collected with our method with added noise, to train a third instance of the model. It achieved a mean IoU for all dishes of 97.21 %.

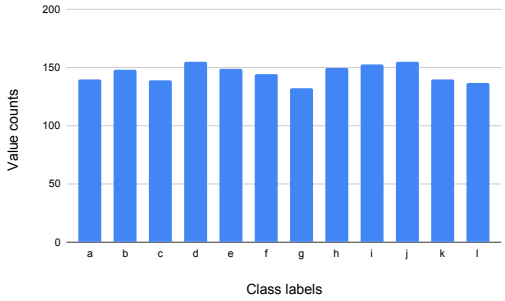
Table 3 summarizes our test results and Fig. 4.18 shows the IoU score for each dish-class by each model instance. As we can see, synthetic data by itself does not make the model robust to inference on real images. However when combined with data collected by our method, it helps



(a) Distribution of synthetic-training data.



(b) Distribution of mobile-robot and invisible-marker data.



(c) Distribution of test data.

Figure 4.17: Distributions of training and test set data.

Table 4.3: Comparison of mean IoU scores on test data.

	Mean IoU on test data (Threshold >0.5)
Synthetic data	73.45 %
Automatically collected data - with noise (Proposed)	94.33 %
Automatically collected data - without noise	93.62 %
Combined dataset	97.21 %

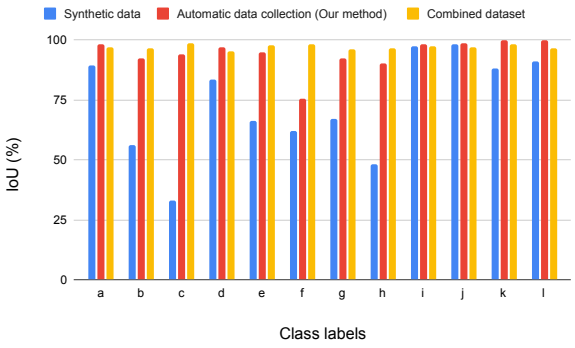


Figure 4.18: Comparison of IoU scores for all dishes.

to increase the robustness of object-detection model by contributing with data of edge cases which would have been difficult to collect manually.

4.4.2 Comparison among methods of data collection and annotation

Time taken to collect images and annotate

Manual data collection is highly laborious and time consuming. It took us around 30 hours for manual data collection and annotation of around 1500 images, i.e., around 70–80 seconds per image. The time required increased as the size of the dataset increased. One could reduce the time by distributing the work among more workers, but the cost would also increase.

Synthetic data generation is a much faster process compared with manual data collection. However, creating accurate 3D assets of objects is an arduous task and requires technical expertise. Setting up the simulation environment also requires domain knowledge and expertise

Table 4.4: Comparison of data-collection methods.

	Manual data collection	Synthetic data collection	Automated data collection with mobile robot (Proposed)
Data collection	- Images are captured manually. - Variation in object position, background, lighting, and other environment variables depend on worker's intuition and knowledge.	- Images of objects in synthetic environment are captured automatically. - Object position, background, lighting, and other environment variables can be randomized programmatically.	- Images are captured automatically. - Object position, lighting, and other environment variables can be randomized programmatically. Scene background can be changed manually or modified in post-processing.
Annotation	- Objects in image and their grasp-position are annotated manually. - Quality and consistency of bounding-box and grasp-position annotation depends on worker's skill level.	- Objects in image and their grasp-position are automatically annotated during data generation. - Quality and consistency of bounding-box and grasp-position annotation is high.	- Objects in image and their grasp positions are automatically annotated using corresponding invisible-marker image during operation. - Quality of grasp-position annotation is consistent, but the bounding-box annotations are not precise.
Cost for data collection + annotation	- Man hour cost for data collection + annotation: 1,500-2,000 USD. - Repeat costs for every new dataset.	- Average cost of 3D model: 250 USD. - Negligible cost for any data generated for same objects.	- Cost for mobile robots and other equipment: 5000 USD. - Negligible cost for any data collected for any object.
Time required for data collection and annotation of one image (seconds).	∼70-80.	∼0.03.	∼0.4.

of the tool. Once the 3D models are available and environment is setup, however, the process is very fast.

It took us less than 5 minutes to generate 10,000 images and a little bit more than an hour to generate the whole training dataset of 65,000 images. This equals around 0.03 seconds per image. The time required for data generation can be further reduced by using better computing resources. We used an NVIDIA GTX 1050 GPU and an Intel i7 processor for the synthetic-data-generation process.

Data collection using mobile robots and invisible markers is a much less laborious task compared with manual data collection. We were able to collect and annotate around 300 images in a little bit over 2 minutes and our whole dataset of 18,000 training images in around 2 hours.

Cost

As manual data collection was done by in-house engineers, we estimated the monetary cost for collecting the images to be around $ 0.45. The cost of manual data annotation was estimated using the current fee listed in Amazon Mechanical Turk, which is around $ 0.80 per image, thus a total of around $ 1.4 per image. This cost could be reduced by outsourcing both the data collection and annotation tasks, but one should consider the cost of quality and training how to collect the data in that scenario.

In synthetic data generation, although the cost of generating a new dataset is negligible, we need to create an accurate 3D model of the objects beforehand. We used an external service to construct accurate 3D models of our objects, which cost us up to $ 250 for each object.

Data collection and annotation with our method has an upfront cost of around $ 5,000 for the mobile robots, three-axis platform, invisible markers and other equipment. Once the setup is complete, the cost of collecting data and annotating an image is very small. Table 4.4 provides a comparison among the various data-collection methods used in our experiments.

4.5 Conclusion

We proposed a method for data collection and annotation using mobile robots equipped with a tilt platform and invisible markers. Such mobile robots enable us to capture images of objects at every possible location inside the camera's field of view. A high-frame-rate image capture combined with switching between white light and UV light enables us to obtain an image-pair of white-light-lit and UV-light-lit images. The invisible marker in the UV-light-lit image helps obtain annotation data during data collection.

The data collected with our method are much more useful in making deep learning models robust against real-world conditions compared with synthetically generated data. We conclude that our proposed method can create large datasets for deep learning models that are consistent, realistic, less time-consuming and inexpensive. Other than object detection, our method can also be used to collect data for grasp-position-estimation and key-point-estimation tasks.

Chapter 5

6D Pose Estimation Using Key-point Estimation

5.1 Problem Formulation and Solution Concept

5.1.1 Problem Formulation

(a) Tilted dish on stack. (b) Flat dish on bottom.

Figure 5.1: Tilted dish vs flat dish.

6D pose estimation is the process of determining the position and orientation (also known as pose) of an object in 3D space. It is an important task in robotics, as it involves determining the position and orientation of an object in 3D space. This information is crucial for a wide range of robotic applications, such as grasping, manipulation, and navigation. However, 6D pose estimation is not a straightforward task, and it is challenging due to several factors [87], [88].

One of the main difficulties in 6D pose estimation is the presence of occlusions and partial views of the object. This can make it challenging to extract accurate features from the image or point cloud data and to match them with a model of the object. Additionally, the presence of noise, clutter, and varying lighting conditions can also make the task more challenging.

Another major issue with 6D pose estimation is the processing time required for current methods. Many of the existing algorithms are computationally intensive and can take several seconds to process a single image or point cloud. This makes it difficult to deploy them in real-world scenarios, where real-time performance is critical. Moreover, the accuracy of current methods is also not sufficient for real-world deployment. This combination of slow processing

time and inaccuracy makes it challenging for 6D pose estimation to be used in real-world robotic applications.

We pose the problem of estimating the 6D pose of a dish in the pre-wash area. This dish would be placed upside down on a stack of other dishes as shown in Fig. 5.1. Here, we mention a stack of dishes as the bottom most dish would be placed on a flat surface.

Knowing the position of dishes would make our dishwashing system more reliable. It would also make it easier for workers since they wouldn't have to align the dishes when they stack them.

In our dishwashing system, we initially assumed that the dishes would be placed relatively flat on the surface. This was to make sure that we only needed to know the 3D position of the dish in relation to the robot in order to pick it up. We made this assumption to make the dishwashing system simpler and able to be used in real restaurants.

The biggest challenge for using a robot system like this is getting the algorithm to predict the 6D position of the dish very accurately. Any differences between the predicted and actual position of the dish can cause the robot to miss it, which will slow down the cleaning process. And, in the worst case, it could break the dish and cost the business money.

5.1.2 Solution Concept

In this chapter, we will discuss our efforts to estimate the 6D position of dishes. We will tackle the problem while keeping strict requirements in mind. These requirements include:

- A low processing time to avoid slowing down the dishwashing process.
- For a deep-learning model, a model that can be trained quickly for fast iteration and debugging
- Low consumption of resources
- High accuracy.

5.1.3 Chapter Organization

This chapter is organized as follows. In Section 5.2, we will first describe related work in this field and analyze its advantages and disadvantages, keeping in mind the feasibility of using it in a real-world setting. Next, in Sections 5.2.1 and 5.2.2, we will describe the methods we tested and evaluate the results.

In Section 5.3, we will present the method we proposed to address the shortcomings of earlier methods. Finally, in Section 5.6, we will examine the drawbacks of our proposed method and discuss potential future work.

5.2 Related Work

In this section, we will describe some of the methods found in literature. Estimating the 6D pose of an object has been a popular theme in robotics as its applications are highly impactful [1], [89], [64], [88], [90].

Among the deep learning methods, Kehl et al. estimate the 6D poses from RGB data in a single shot extending the popular single-shot detection (SSD) method [64]. They infer the 6D pose from the 2D bounding boxes and then refine their pose.

Tremblay et al. train a convolutional network capable of detecting an object and estimating its 6D position using only synthetic data [1]. We will describe this approach in detail in Section 5.2.2 and evaluate its strength and weaknesses for practical use.

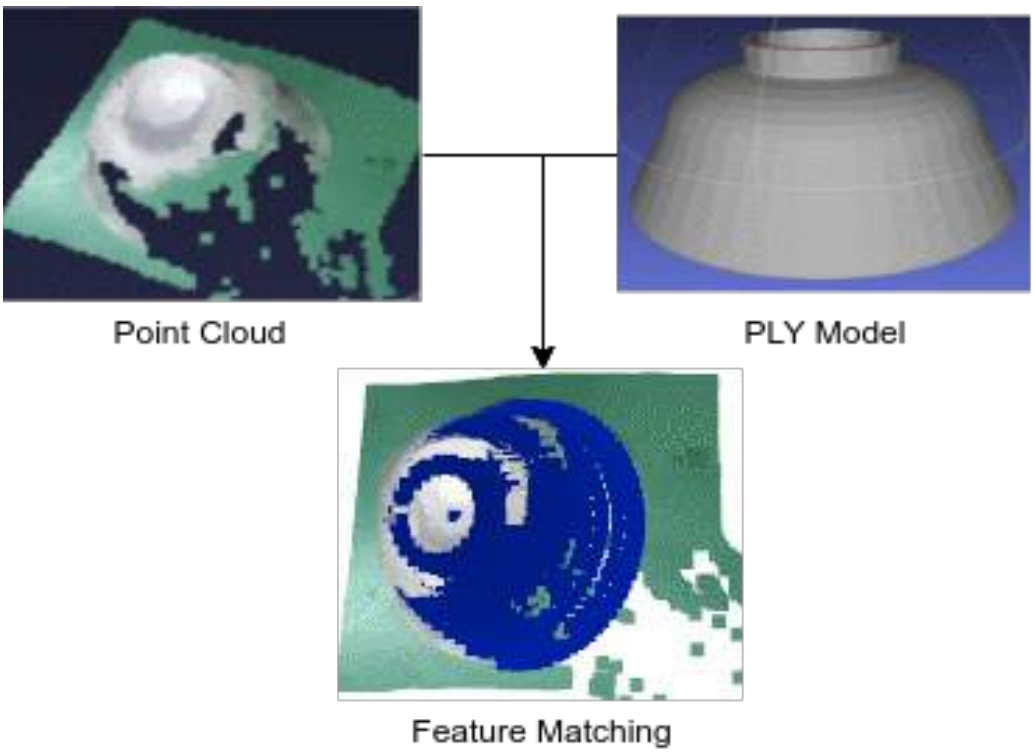


Figure 5.2: Global registration.

5.2.1 Global Registration

Global registration is a method that doesn't need an initial alignment. It usually gives less precise results compared to local registration methods like Iterative Closest Point (ICP) and colored point cloud registration. But it can be used to start local registration methods.

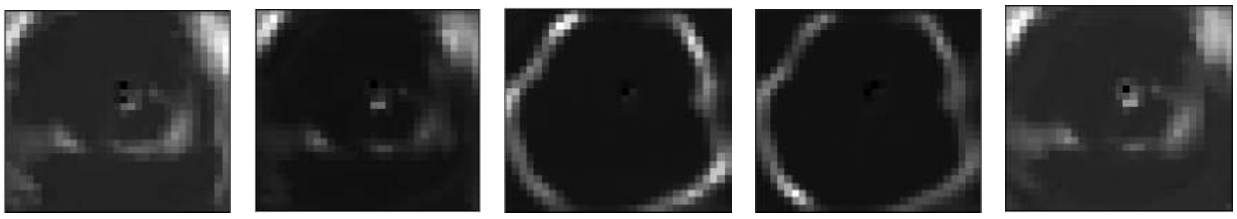
We used Open3D's global registration algorithm. First, we made an accurate 3D model of the object using 3D scanning or a 3D modeling software. We then made the model smaller for easier processing. This model was used as the target for the global registration algorithm.

Next, we used a 3D camera to get an RGB image and the point cloud of the scene. We used an object detection framework like YOLOv3 to find the object and crop the point cloud that corresponds to the object. This point cloud was then used as the input for global registration.

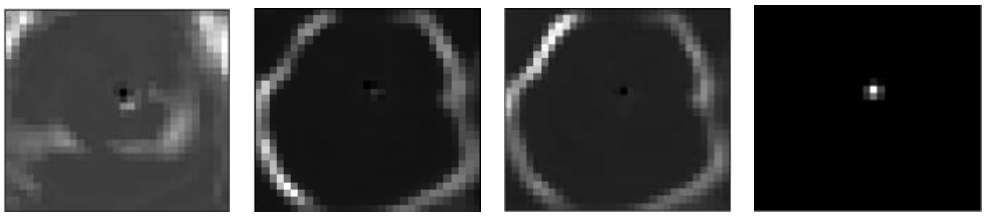
We reduced the size of the point cloud, found the normals, and computed a 33-dimensional Fast Point Feature Histogram (FPFH) feature for each point. The FPFH feature describes the local geometric properties of a point. The theoretical computational complexity of the FPFH descriptors for a point cloud with n points is $O(nk)$, where k is the number of neighbors for each point in the point cloud [91].

The results of our implementation on the target object are shown in Fig. 5.2. As we can see, the algorithm can match even a partial point cloud with the input 3D model. This result was recorded when the object was directly under the camera. However, one major drawback of this method is that it requires an accurate 3D model of the object. This can add extra cost and time for preparing a 3D model for each new object, and also affect memory usage as the models need to be stored during the process.

In practice, we've noticed other limitations of this method. For example, the algorithm's accuracy in predicting the 6D pose of the dish is greatly reduced when the dish is at the edge of the camera's view. This makes it difficult to use in real-world scenarios where the object may not always be directly under the camera. Additionally, the processing time for an unoptimized version of this algorithm is around 1-2 seconds for each object in the scene, which would result in 10-12 seconds per frame for a scene containing 5-6 objects.



(a) Vertex 1. (b) Vertex 2. (c) Vertex 3. (d) Vertex 4. (e) Vertex 5.



(f) Vertex 6. (g) Vertex 7. (h) Vertex 8. (i) Centroid

Figure 5.3: Belief maps of the vertices and centroid.

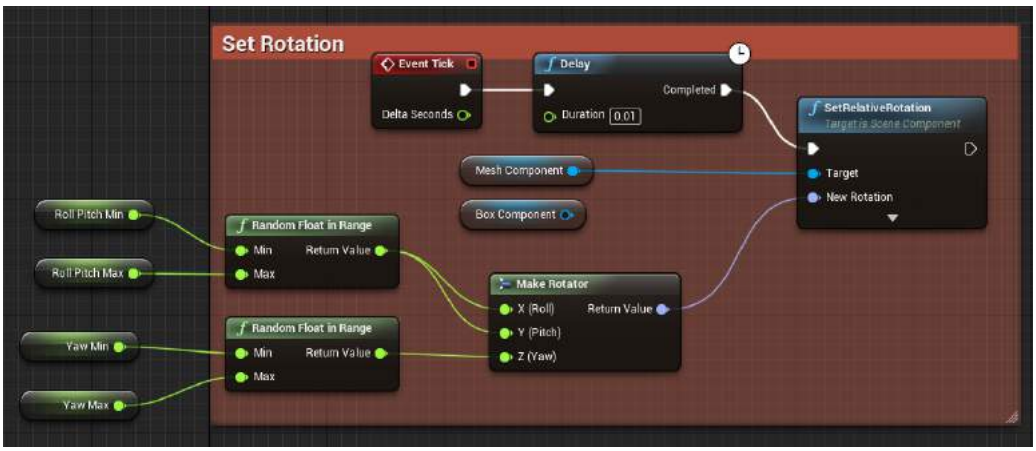


Figure 5.4: UE4 Blueprint to restrict rotation.

5.2.2 Deep Object Pose Estimation [1]

In this subsection, we describe the second method we attempted to estimate the 6D position of our objects using the deep object pose estimation (DOPE) network [1].

The input to the model is an image containing one or more instances of a single object. The labels are 3D bounding boxes, which include eight vertices and one centroid. Since the labels are 3D bounding boxes, it is very difficult to manually annotate such data. To do this, we used a tool provided by the authors of [1]. This tool is described in Section 2.5.4.

We encountered a problem in our first iteration, where our trained model was unable to detect any objects or estimate their position. Upon visualizing the belief maps for the eight vertices and the centroid shown in Fig. 5.3, we discovered that although the centroid was being detected, the vertices were not.

This happened because our objects are symmetric in shape [58]. To solve this issue, we restricted the rotation of our objects on the z-axis. Fig. 5.4 shows the blueprint we made in UE4 to restrict the object rotation in different axes.

After retraining the model, we were able to detect the dish and estimate its 6D position. Figure 5.5 shows a sample result from the DOPE network. As we can see, the network is able to estimate the 6D pose of the two dishes in the image accurately. However, in practice, we found that the estimated 6D position is not very consistent. This can affect the robot's motion

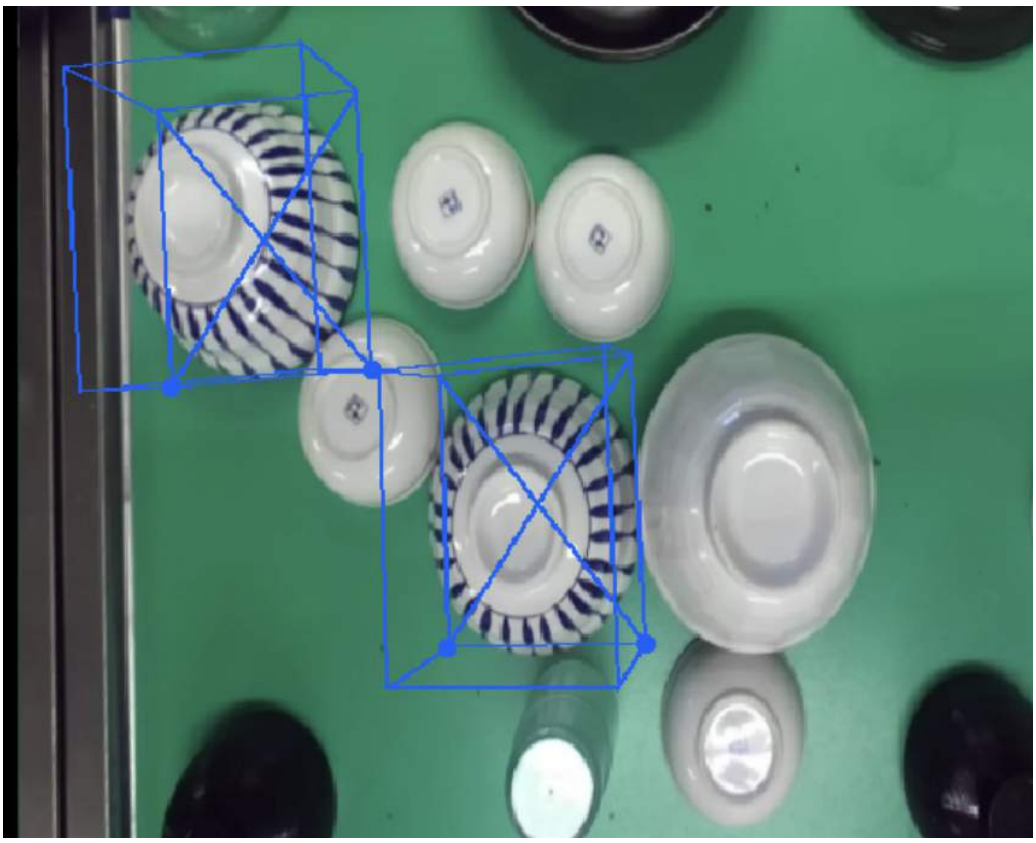


Figure 5.5: Pose estimation result from DOPE network.

planning as it receives different positions for grasping the object.

Another problem with this method is that it cannot detect the type of dish. This means that a separate model needs to be trained for each individual object. This can be a huge burden on both memory and processing, as multiple models are needed for different objects.

Additionally, training a single model requires around 50,000 - 100,000 synthetic images, and takes around 2-3 days on a NVIDIA Tesla K10 GPU. This results in long training times for any new iteration.

In conclusion, the DOPE network [1] is useful for detecting objects and estimating their 6D position using only synthetic images. However, since it was mainly designed for research purposes, it is challenging to deploy it without significant optimization and changes to the model.

5.3 Proposed Method

In our method for estimating the 6D position of an object, we build upon our previous work in Chapter 3 and Chapter 4. We start by making some assumptions, such as assuming that even if the dishes are tilted, they will still be upside down, allowing the robot to grasp them. For dishes that are right side up, they will be ignored by the pose estimation network and reported in the user interface.

We also limit the range of tilting angles to zero, thirty, forty-five, and sixty degrees. This is because in our application, we do not need a very precise estimation of the angle of tilt since the suction gripper can adapt to small differences. Additionally, the maximum value of tilt is limited to fifteen degrees, since dishes can easily topple over when the angle exceeds around fifteen degrees. Furthermore, since most of our objects are symmetric, we keep the rotation around the z-axis at zero at all times.

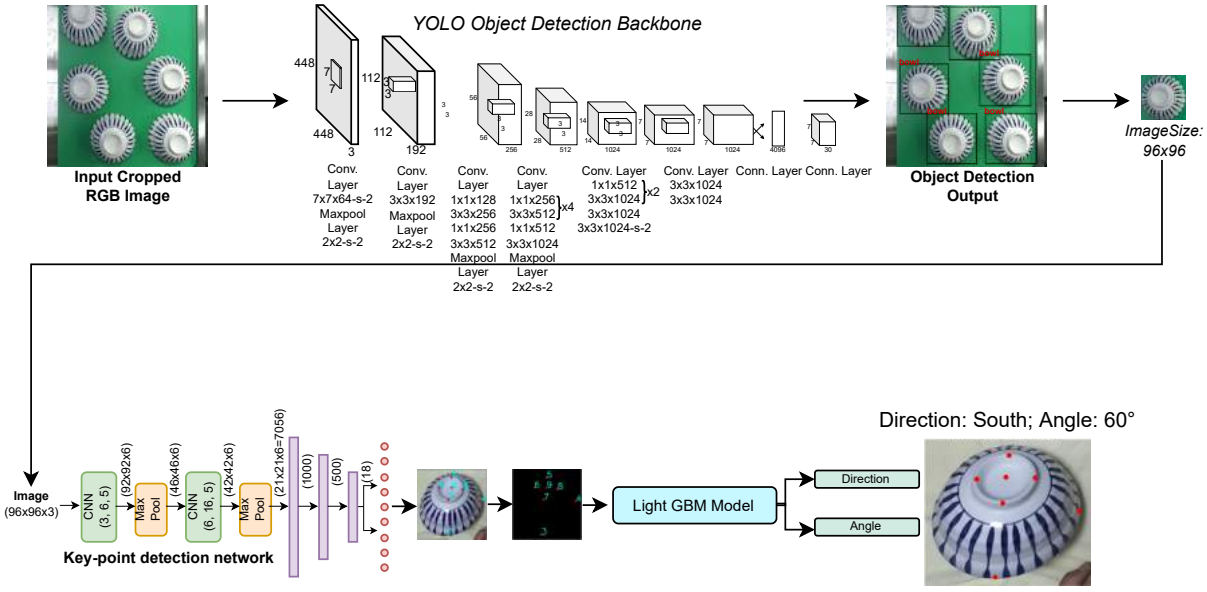


Figure 5.6: 6D pose estimation network.

Additionally, in the beginning, we use discrete values for the direction of tilt, such as north, south, east, and west. These assumptions were made to simplify prototyping and ease data collection in the early stages. However, in practice, it would be ideal to have continuous values for better accuracy, particularly in the case of the direction of tilt.

Our network is illustrated in Fig. 5.6. We use a modular architecture on top of our object detection framework, similar to our previous work. The first model detects key-points in the cropped image. Each dish has nine key-points. Based on the angle of tilt, some key-points are visible and some are not. During training, the key-point estimation network takes as input the cropped image and the annotation of the nine key-points. In inference, it predicts the nine key-points, with non-visible key-points output as negative values.

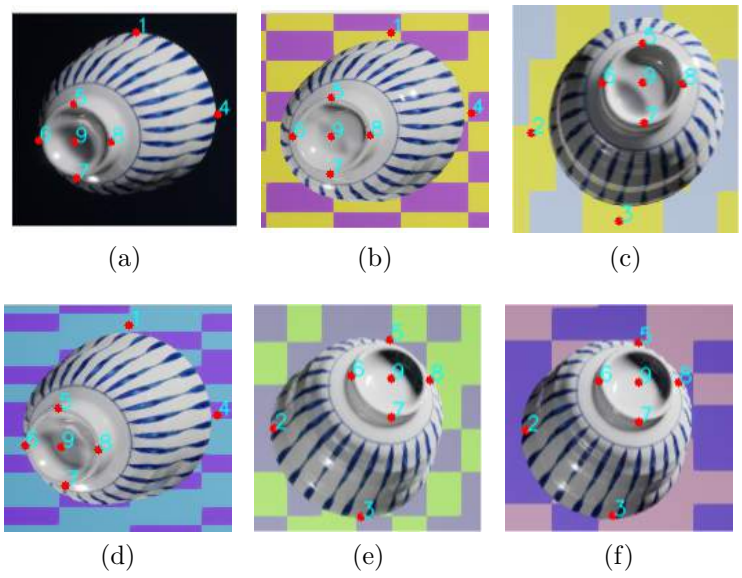
The results of the key-point detection are then passed to a LightGBM (LGB) model, which estimates the direction and angle of tilt based on the position of key-points. The LightGBM gradient boosting framework is designed to be efficient and scalable, and it is a popular choice for working with large datasets. LGB "booster" refers to the underlying algorithm used to train the model, which is based on gradient boosting. Gradient boosting is a method for fitting an ensemble of weak models to minimize a loss function, such as mean squared error for regression or cross-entropy for classification. LGB booster specifically uses a tree-based algorithm for fitting the weak models, which allows for efficient handling of large datasets and categorical features [92].

During training, this model takes as input the tabular data of the nine key-points along with the annotation of the angle and direction of tilt. In inference, it predicts the angle and direction of tilt given the position of the nine key-points.

5.3.1 Data Collection

To collect data for the key-point estimation network, we extend the method described in Section 3.3.5. For this network, we require nine sockets for each key-point. The first four keypoints are placed at the edge of the dish in counter-clockwise mode. The next four keypoints are placed at the edge of the picking area. And the final keypoint is the same as the grasping position of the dish. A sample of our synthetic data is shown in Fig. 5.7.

For the ground truth images, we tilt the dishes manually and perform manual annotation.



(a) (b) (c)

(d) (e) (f)

Figure 5.7: Key-point synthetic data sample.

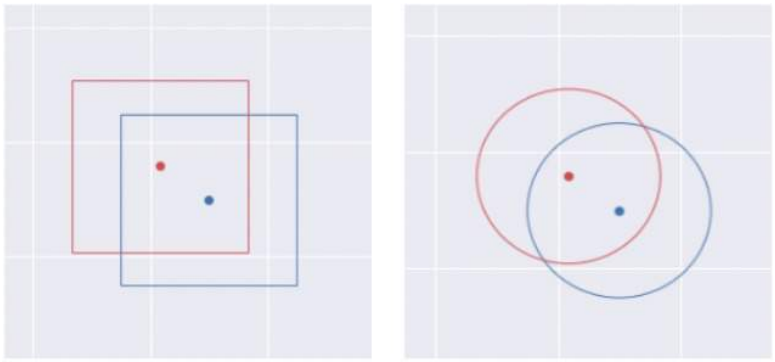


Figure 5.8: Square IoU vs Circular IoU.

Table 5.1: Feature importance.

Feature Name	Feature Importance
Keypoint 5 - Y	227
Keypoint 4 - Y	213
Keypoint 4 - X	67

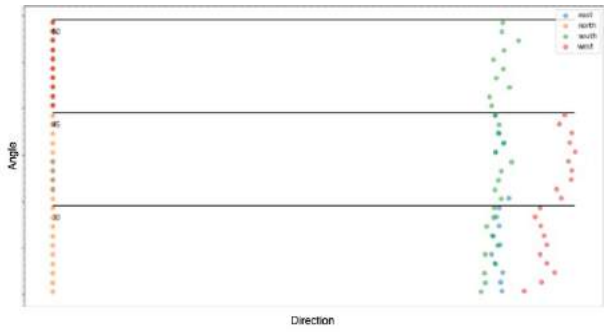
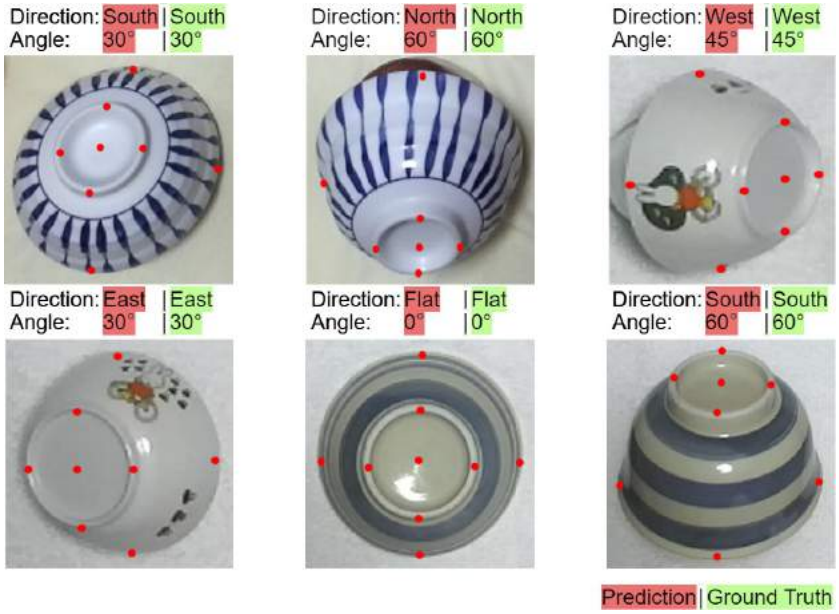


Figure 5.18: Feature analysis on sample feature.

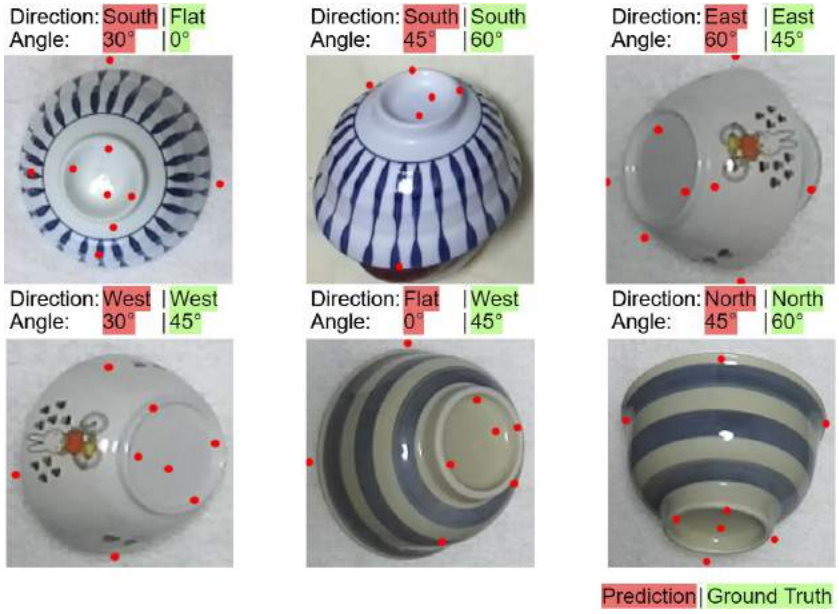
Further, we also use mobile robots and 3-axis tilt platform for real data collection mention in Sections 4.3.3 and Section 4.3.5.

5.4 Experiments

For evaluating our results of key-point estimation, we use the intersection over union (IoU) metric. IoU is a commonly used metric for evaluating the performance of object detection algorithms. It is defined as the ratio of the area of intersection between the predicted bounding



(a) Accurate predictions.



(b) In-accurate predictions.

Figure 5.19: Key-point estimation result on real-data.

box and the ground truth bounding box to the area of union between the two boxes. In other words, IoU is a measure of how well the predicted bounding box overlaps with the ground truth bounding box.

But instead of using square regions for calculating the IoU, we use circular regions as seen in Fig. 5.8. Using circular regions, also known as keypoint heatmaps, as a metric for predicting keypoints is considered to be a better approach than using bounding boxes. It better represents the shape of the keypoint, is more robust to small changes in the position of the keypoint, allows for the use of continuous values to describe the confidence of a keypoint prediction, and allows to separate different keypoints in crowded regions. Overall, it provides a more accurate and robust evaluation of a model's performance, making it a preferred method in many cases.

Once the key points are detected, we analyze the features and engineer new features for the LGB model to better predict the direction and angle. Feature analysis and feature importance are techniques used to understand and interpret the behavior of machine learning models. The

goal of feature analysis is to understand how the model is using the input features to make predictions. Feature importance, on the other hand, is a measure of the relative contribution of each feature to the model's predictions. Both of these techniques can be used to identify patterns in the data and understand how the model is making its decisions.

Figure 5.18 shows a sample visualization of feature analysis on one of the key points. In Table 5.9, we can see that keypoint 5 and keypoint 4 are highly influential in predicting the correct angle and direction.

5.5 Results

We can see the results of our pose estimation network in Fig. 5.19. For our test set sample, the direction and angle prediction is highly accurate when the keypoints are accurately predicted, as seen in Fig. 5.19 (a). But, in Fig. 5.19 (b), we can see that the results of angle prediction are affected due to inaccurate prediction of the key point position.

5.6 Conclusion

In this chapter, we proposed using key-point detection to detect the 6D pose of an object, specifically circular dishes kept upside down in a stacked configuration. We first attempted to use a classical method called global registration and a deep learning method called deep object pose estimation, but found them to be too slow for our use case. To solve this, we made some assumptions to reduce the problem setting. We limited the direction of tilt to north, south, east, and west and constrained the angle of tilt to 15, 30, 45, and 60 degrees. However, this approach has a major drawback, which is the limitation of the angle of tilt to discrete values, which would be less useful in real-world scenarios where continuous values would be more useful.

To make the detection faster, we used a shallow network for detecting keypoints and then predicted the direction and angle of tilt based on the keypoints' positions. We borrowed the idea of using a shallow network from Chapter 3 and used the idea of data collection from Chapter 4, where we used an additional three-axis platform on top of the mobile robot.

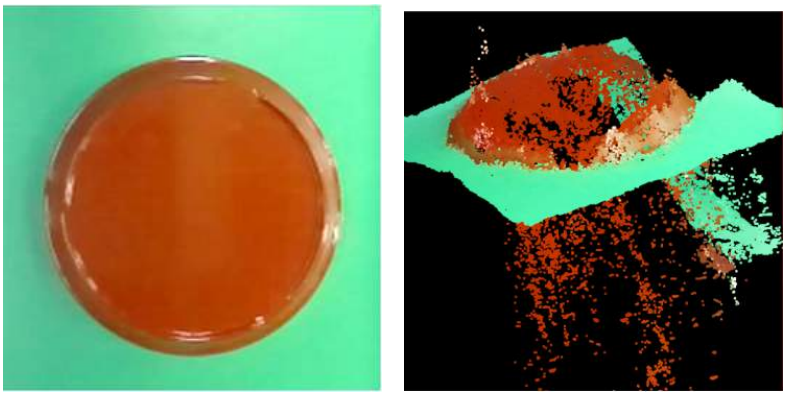
Overall, our proposed method is a valuable approach to quickly detecting the 6D pose of an object on limited resources, but it has some limitations that need to be addressed in future research. The limitation of the angle of tilt to discrete values is a major drawback and future work should investigate ways to detect 6D pose in continuous values. Additionally, expanding the approach to other objects besides circular dishes and improve the accuracy of the detection will be an interesting area to explore.

Chapter 6

Addressing the lighting issue in unstructured environments for obtaining reliable point cloud data

6.1 Problem Formulation and Solution Concept

6.1.1 Problem Formulation



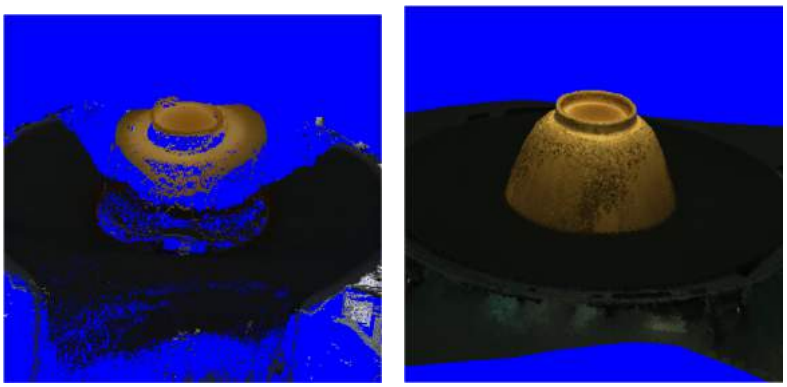
(a) RGB image. (b) Point-cloud measurement.

Figure 6.1: Point-cloud result with bad lighting environment.

As discussed in Chapter 1, dealing with harsh lighting conditions is a significant issue for the vision system of service robots operating in unstructured environments [93], [94], [95], [96].

Harsh lighting conditions pose a significant challenge for robots, particularly in depth-estimation and object detection. One major problem is that the intense glare or shadows created by harsh lighting can distort the visual information available to the robot, making it difficult to accurately estimate the distance to objects or surfaces. This can be particularly problematic for tasks such as pick and place. It can be difficult for the robot to accurately estimate the depth of the objects, as the intensity of the lighting can affect the way the objects are perceived.

Object detection is another area where harsh lighting can cause problems for robots [97], [98]. The high contrast and strong shadows created by harsh lighting can make it difficult for traditional object detection algorithms to accurately identify and classify objects. However, recent advances in deep learning have led to the development of more robust object detection algorithms that are able to handle a broader range of lighting conditions. These algorithms can learn from large amounts of annotated training data [1], [99]. They can be fine-tuned to perform



(a) Harsh-lighting point-cloud measurement.

(b) Ideal point-cloud measurement.

Figure 6.2: Ideal and non-ideal point-cloud measurement.

well in specific environments [100]. Unfortunately, while these deep learning-based approaches can be effective at addressing the challenges posed by harsh lighting for object detection, they are not as effective at addressing the problems with depth estimation.

In our use case, due to harsh lighting conditions, the point-cloud data is missing a lot of information as seen in Figure 6.1. This can lead to incorrect or missing 3D position of the grasp location. However, for pick and place applications, it is ideal to have complete point-cloud data without any missing information, as shown in Figure 6.2.

6.1.2 Solution Concept

This chapter addresses the problems with harsh lighting conditions using system design methods which 1) control the camera position, 2) control the position of lighting, 3) using multiple cameras to obtain views from different positions; and deep-learning approaches where we use an auto-encoder to rectify the poor point-cloud data.

In the system design methods, we first proceed with measuring the variation in lighting at different times of the day. We then briefly explain the difficulties in controlling these lighting conditions. We then proceed to explain how

6.1.3 Chapter Organization

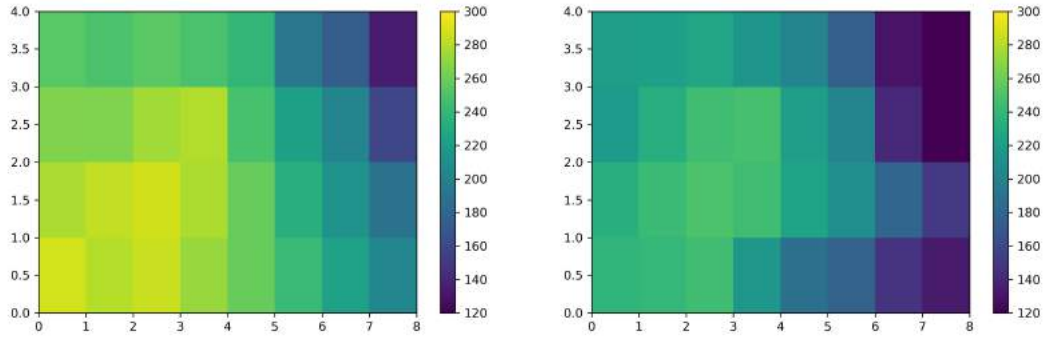
This chapter is organized as follows: in Section 6.2, we briefly describe the theory behind the stereo-vision and the process of obtaining the depth image and the point-cloud data from a pair of stereo images. We also describe the importance of ideal lighting conditions to obtain high-quality depth images and point cloud data. Further, in Section 6.3, we present some auxiliary methods that attempted to solve the lighting issue and provide the advantages and disadvantages of each method. Section 6.4 describes the deep-learning approach to fill the holes in point-cloud data using auto-encoders. Finally, section 6.5 draws a conclusion on the main findings and summarizes the major limitations of the presented methods.

6.2 Stereo Vision

Stereo vision is a technique used by robots and computer systems to extract three-dimensional (3D) information from two-dimensional (2D) images. It involves the use of two cameras or a single camera with two lenses, which allows for the creation of a 3D model of the environment

Time: 17:00							
221	221.4	225.6	214.6	201.8	174	129.2	110
218	233	244.6	246.9	220.5	201.8	140.5	112.8
234	243	249.3	244.5	225.4	209.2	179	150.3
238	240	244.9	215	185.4	175.8	147.2	133
Time: 14:00							
253.2	249	254	248.2	239.4	192.4	173.4	134.5
266	265.2	275.2	279.1	247.4	222	202.1	158.4
277.2	283.7	287	277.8	257.9	234.5	212.2	188.7
282	280.1	284.7	272.5	258.4	242	224	202.5

(a) Raw lux readings.



(b) Afternoon.

(c) Evening.

Figure 6.3: Light intensity (in lux) map for pre-wash setup in morning vs afternoon.

by triangulating the position of objects based on the difference in perspective between the two cameras [101].

One of the key challenges in stereo vision is achieving accurate depth perception, which relies on the ability of the system to identify matching points between the two images [101]. This process, known as correspondence matching, can be influenced by factors such as image resolution, noise, and the presence of textureless or reflective surfaces [102].

Ideal lighting conditions are important for successful stereo vision as they can improve the visibility of features and reduce the effects of noise and shadow [94], [93], [96], [95]. In particular, it is important to ensure that the lighting is evenly distributed and that there are no strong shadows or specular reflections, as these can interfere with the correspondence-matching process.

6.3 Auxiliary Methods

In this section, we shall describe some of the auxiliary methods we attempted in order to solve the problems described in Section 6.1.1.

Before implementing the following methods, we first measured the light intensity on the working area using a lux meter. We divided the working area evenly in small grids and measured the light intensity in each grid at different times of the day. Figure 6.3 shows the results of our light intensity readings. As we can see, under natural conditions, the light intensity varies greatly and unevenly over the working area.

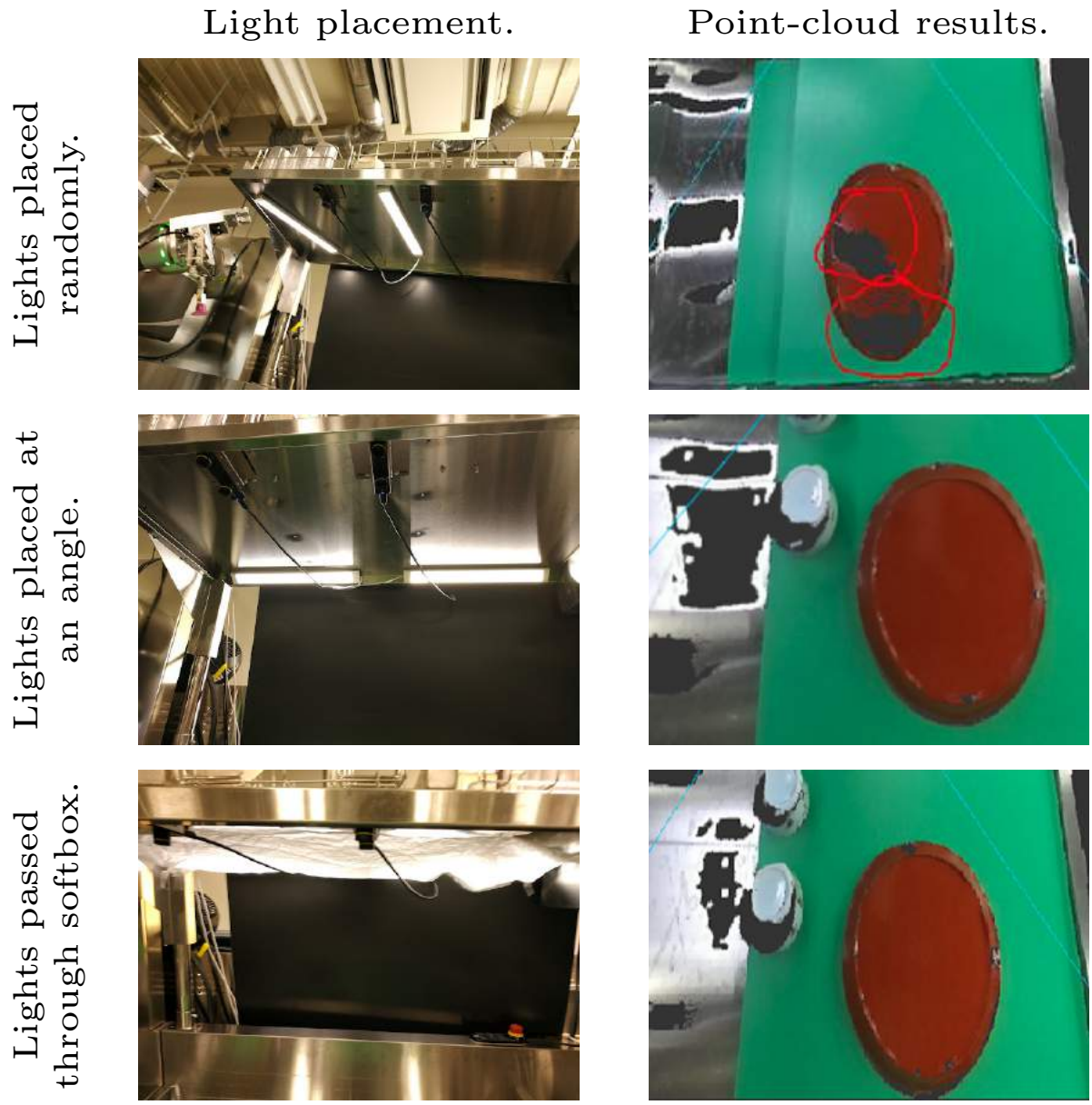


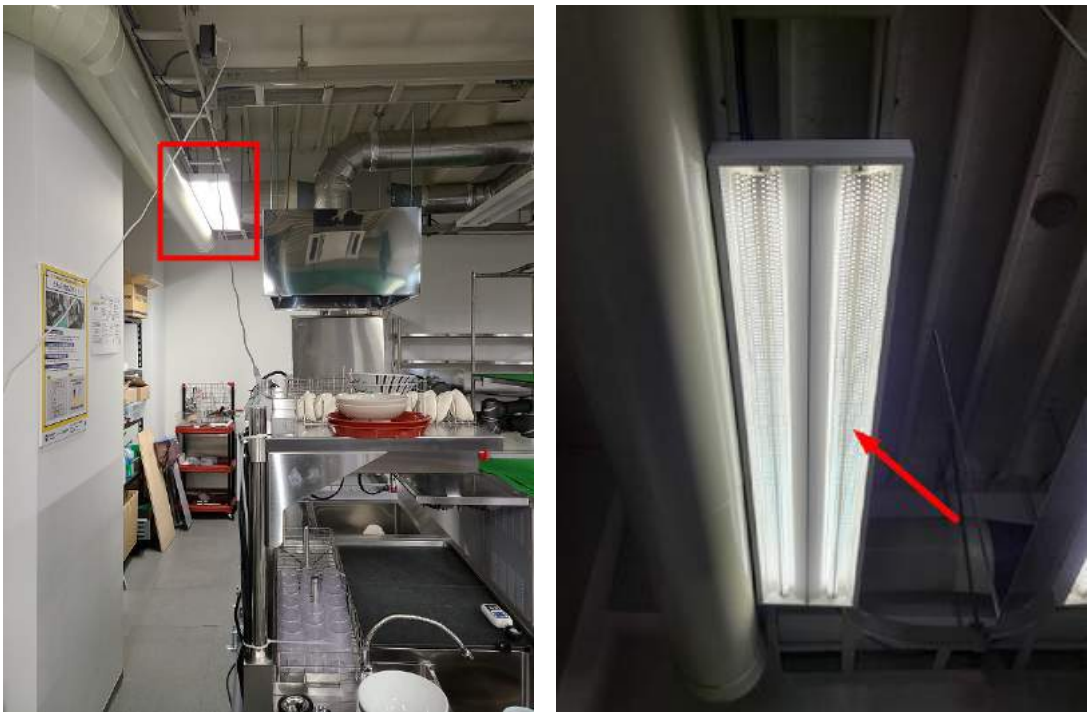
Figure 6.4: Experiments with light placement.

6.3.1 Using a Soft-Box

A softbox is a lighting accessory that is commonly used in photography and videography to reduce harsh lighting and create a more even, diffused light source. It consists of a box-shaped frame with a translucent fabric or diffuser stretched over one end, and it is typically used to surround a light source such as a flash or continuous light.

Using a softbox can help to reduce harsh lighting in several ways. First, it diffuses the light, meaning that it spreads the light out over a wider area rather than focusing it into a single beam. This can help to eliminate harsh shadows and create a more even, natural-looking light. Second, it can reduce the intensity of the light by absorbing and scattering some of the light particles, making it less harsh and more pleasant to look at. By using a softbox to diffuse the light and eliminate harsh shadows, it can help to improve the accuracy of the depth estimation and allow the robot to perform its tasks more effectively.

We used off-the-shelf light-emitting diodes (LEDs) and a translucent piece of fabric to create a makeshift softbox, as seen in Fig. 6.4. We also used a diffuser on the overhead lighting, as



(a) Before. (b) After.

Figure 6.5: Modifications to lighting environment.

seen in Fig. 6.5. As we can see in Fig. 6.4, row (1), when the lights are placed randomly, and the light intensity is not uniform, the point cloud data has many holes.

Upon improving the placement of the LEDs, we can see in Fig. 6.4, row (2) that the point cloud has lesser holes. This is due to the uniform distribution of the light intensity. Since we don't have too high or too low intensity focused on one particular spot, it is easier for the stereo vision cameras to detect the features and form the depth map.

In Fig. 6.4, row (3), we see that upon passing the light from the LEDs through a white cloth acting as a softbox, the light intensity is less harsh and more uniformly distributed. This prevents any harsh light from falling on the pre-wash area. We can see that controlling the lighting environment in the manner described above allows us to create perfect conditions for our vision module.

While using a softbox can be an effective way to reduce harsh lighting and improve depth estimation in robotics, it can also be difficult to implement in real-world applications. This is because the lighting conditions in a real-world environment are often much more complex and dynamic than in a controlled laboratory setting. For example, in the restaurant environment, the lighting conditions can be affected by different sources, including natural light from windows, fluorescent lighting from overhead fixtures, and reflections from shiny surfaces.

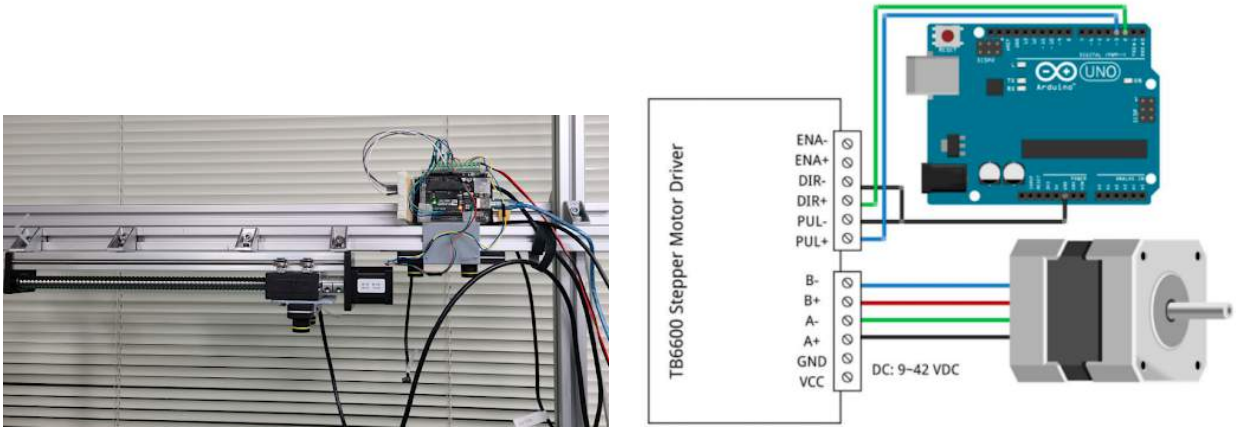
In such situations, using a softbox may not be sufficient to completely eliminate harsh lighting, as it can only affect the light source that is directly within its frame. Other lighting sources, such as reflections or ambient light, may still interfere with the robot's vision and cause problems with depth estimation.

6.3.2 Using Lens Cover with Polarizer

A polarizer lens cover works by blocking certain wavelengths of light polarized in a specific direction, allowing only light with a perpendicular polarization to pass through. This can help to reduce glare and reflections. Figure 6.6 shows our setup of a 3D printed lens cover equipped

(a) ZED lens cover. (b) Polarizer (c) ZED with lens cover.

Figure 6.6: ZED camera with lens cover.



(a) Hardware setup. (b) Wiring setup.

Figure 6.7: Sliding camera setup.

with a set of polarizers. It helps reduce the effect of glare after fine-tuning the polarizers.

However, implementing a polarizer lens cover in a real-world robotic application can be difficult for several reasons:

- The polarizer lens cover must be carefully aligned with the polarization of the light source to be effective. This can be challenging in a dynamic environment, where the direction of the light may change over time.
- The polarizer lens cover may reduce the overall brightness of the image, which can make it more difficult for the robot to see in low-light conditions.
- The polarizer lens cover may be prone to wear and damage over time, which can affect its effectiveness.

6.3.3 Change Camera Position Dynamically

In another experiment, we changed the position of the camera dynamically whenever it encountered harsh reflections. Allowing the camera to change its position lets it avoid positions where it can encounter harsh lighting.

Figure 6.7 shows our experimental setup to move the camera using a linear guide. The linear guide is actuated with a stepper motor. We attached the ZED-M camera to a linear

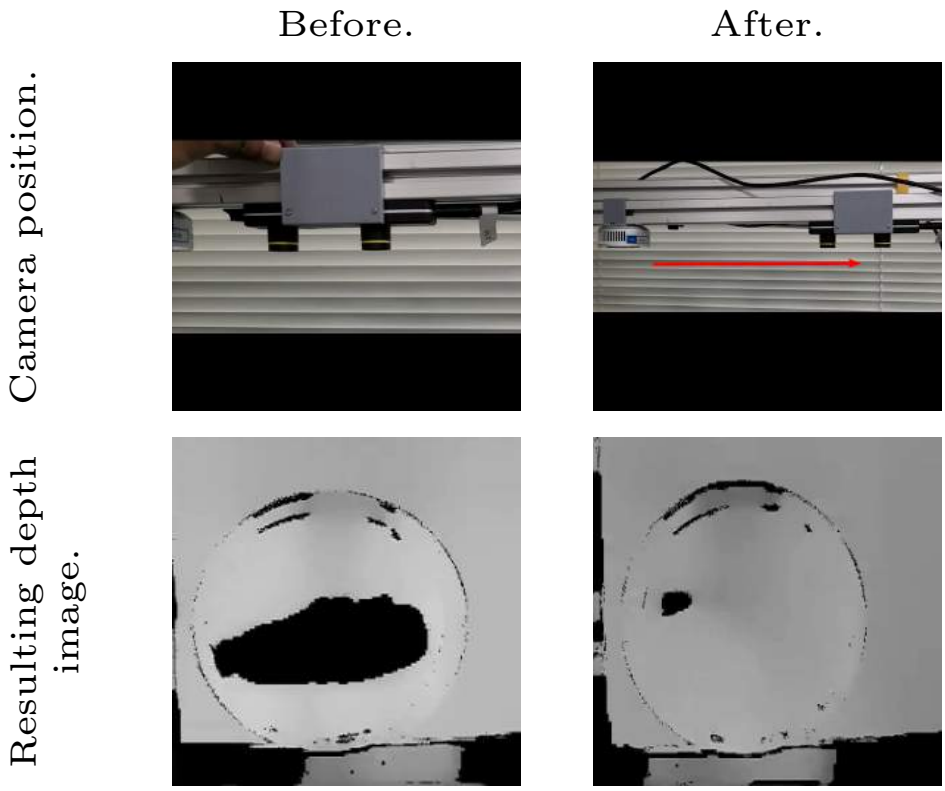


Figure 6.8: Sliding camera to avoid holes in point cloud.

Table 6.1: Sliding camera speed experiments.

Microstep	**Delay (Frequency) in microseconds**	**Voltage (V)**	**Current (A)**	**Time to reach 300 mm (s)**
16	50	27.5	0.12	13.2
8	110	27.5	0.12	12.8
4	230	27.5	0.12	13.07

guide. We used Arduino UNO and TB6600 Stepper Motor Driver to control the stepper motor of the linear guide. The linear guide has a total length of 300 mm.

There are a few ways of achieving linear motion, namely, a) using a timing belt drive actuated with stepper motors, b) using a ball-screw linear guide actuated with stepper motors, and c) using a linear motor. A ball-screw linear guide actuated with stepper motors is a cheaper alternative to linear motors, which are far more expensive and take up much more space. The advantage of using a ball-screw linear guide over a timing belt drive is the accuracy of motion. Since the stepper motor receives signals in multiples of step-rate controlled by the driver, we can precisely predict the position of the camera with respect to its starting position.

One drawback of the ball-screw linear guide is the low acceleration and velocity. Table 6.1 shows the time taken to move the camera at different micro-steps.

In Fig. 6.8, we can see the results of our experiments. In row (1), at the start of the camera position, the stereo vision module is not able to estimate depth due to harsh reflections. But upon moving the camera position along one of the horizontal axes, we see that the harsh reflections do not fall directly on the camera sensor. This allows it to detect the features and create a valid depth map.

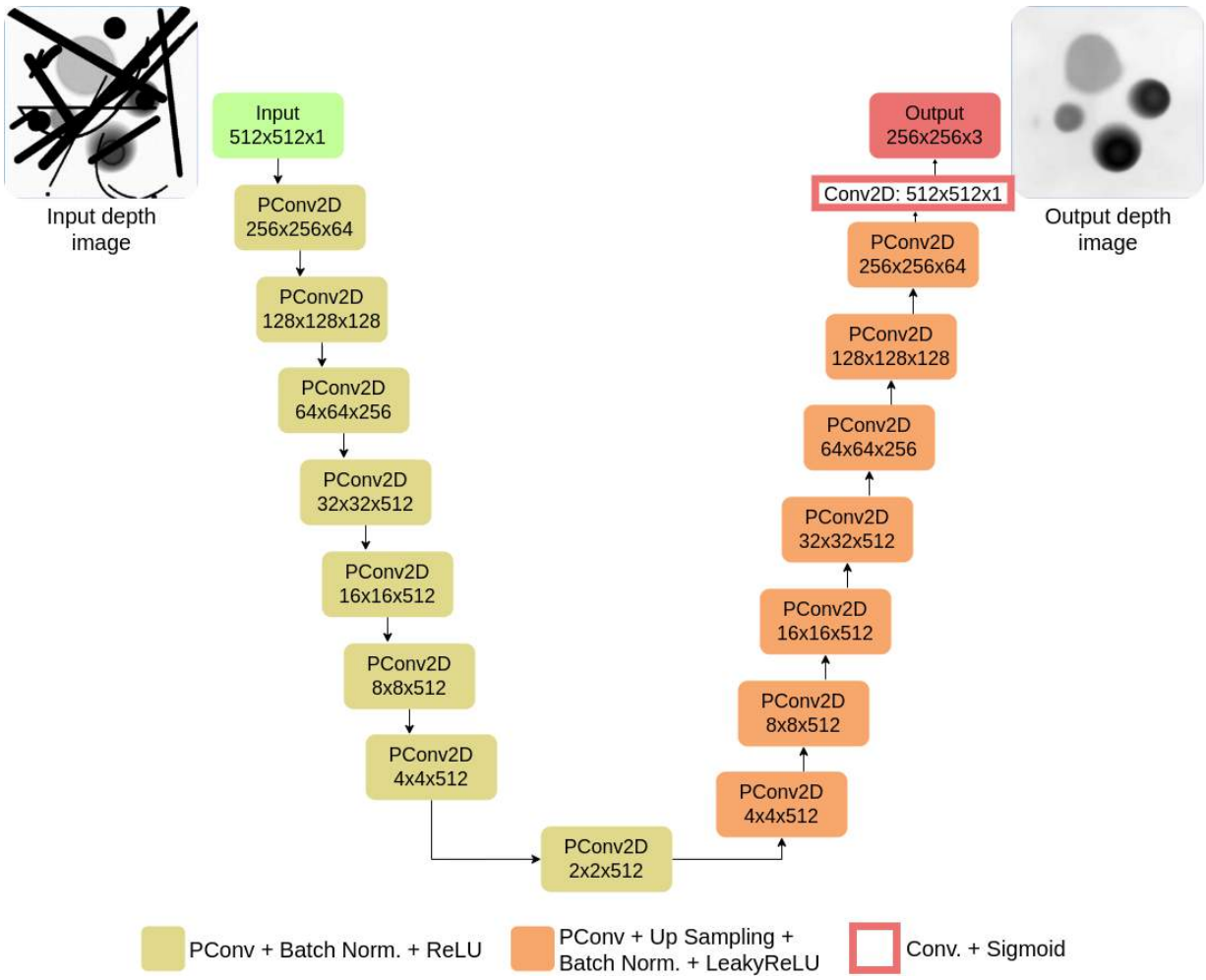


Figure 6.9: U-Net structure.

6.4 Auto-Encoders to In-paint Depth-Image

[1]

6.4.1 Introduction

Auto-encoders are neural network architectures that are used to learn compact, efficient representations of data. They do this by training a network to reconstruct its input data from a lower-dimensional encoding. This process is useful for a number of tasks, including dimensionality reduction, data compression, and missing data imputation [103], [104], [105].

One common application of auto-encoders is to use them to fill in missing information in images. This can be achieved by training an auto-encoder on a dataset of images with some missing pixels, and then using the trained model to generate the missing pixels when presented with a new image with missing information. This process can be applied to any type of image, including depth images from stereo-vision cameras.

To fill in missing information in depth images, the auto-encoder can be trained on a dataset of depth images with some missing pixels or regions. The trained model can then be used to generate the missing depth values when given a new depth image with missing information. This can be useful for improving the accuracy and completeness of depth maps, which are used

[1]This work was done in collaboration with National Institute of Advanced Industrial Science and Technology (AIST).

in a variety of applications such as robotics, augmented reality, and 3D modeling.

One variant of the auto-encoder, known as the partial convolution (pconv) auto-encoder, has proven particularly effective for image in-painting tasks. Pconv auto-encoders are able to handle irregularly shaped missing regions, as they do not require a fixed mask size. This allows them to more accurately predict the missing information, resulting in higher quality in-painted images.

Partial convolution is defined as,

$$\begin{cases} W^T(X \bigodot M)\frac{sum(1)}{sum(M)} + b, & if sum(M) > 0 \\ 0, & otherwise \end{cases} \tag{6.1}$$

, where W is the weight of the convolution filter, b is bias, X is pixel value for window, M is the binary mask.

According to the formula for Partial Convolution, the convolved value is output if there is even one unmasked pixel in the window. This means that the mask will gradually shrink and eventually disappear when a sufficient number of Partial Convolution layers are applied.

It's worth noting that the original implementation of Partial Convolution uses a pre-trained network for the loss function, which was trained on general 2D color images. Using this loss function for depth image inpainting can lead to poorer results. To improve performance, the loss function was modified to use the L2 norm.

6.4.2 Proposed Solution

We used U-Net to develop and train a network for our purposes. The configuration of the network used in the experiment is shown in Figure 6.9. The input image is on the top left and the output image is on the top right. Both images have a resolution of 512x512 pixels and the pixel values are unsigned 16-bit integer grayscale.

When this trained network is given a depth image with defects as input, it produces an output depth image in which the depth values for that region have been restored. We have observed that the pixel values can change due to image conversion by the CNN, and in the case of depth images, the height of the object may also increase or decrease. We found that these changes are almost linear, so we applied a straight line to the depth values before and after conversion and corrected the depth values after conversion to match the values before conversion.

We used depth images of real dishes to train the network. We used two ZED minis as the cameras for our system, one looking directly down on the dishes and the other at a slight angle. By using two cameras, we were able to increase the number of images and also collect data on tilted dishes. In addition, we used a Zivid One+ camera, which has higher accuracy than the ZED mini, to acquire depth images as approximations of the true values. The Zivid camera is larger, heavier, and more expensive than the ZED, so it was only used for creating training data. We also calibrated the axis of rotation of the stage, the direction of movement, and the coordinate conversion between cameras using markers, which allowed us to synthesize point groups measured from multiple viewpoints. Finally, to create data that was closer to the true value, we measured the dishes from four directions using the Zivid and integrated the results to reduce defects.

We automated the process of collecting training data to save time. The dishes was placed on the programmable XY-rotation stage shown in Figure 6.10 and measured. The stage moved and rotated while the dishes was placed on it, and the measurement results were recorded. The camera measurements and stage movement were automated by a program, saving the labor of manually collecting training data. We collected about 14,000 data using this method. We also created approximately 6,000 data with true values, which were used as the final training data.

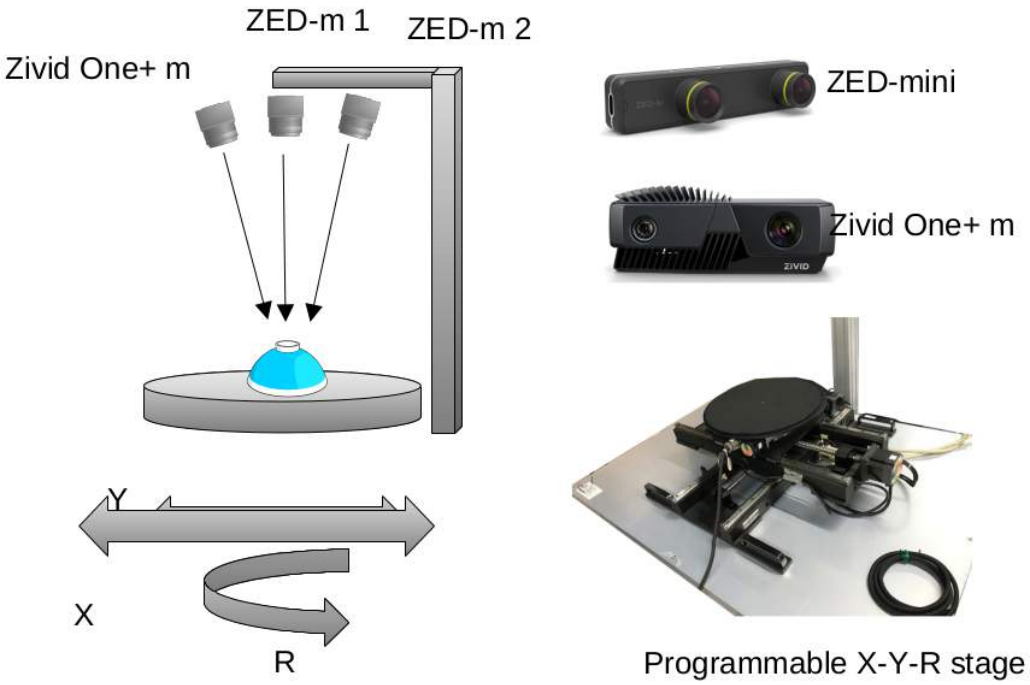
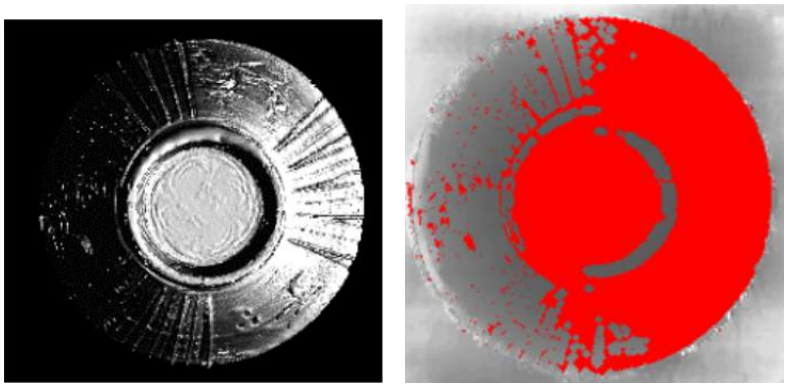


Figure 6.10: Data collection setup.



(a) Synthetic image with specular reflection.

(b) Masked depth image.

Figure 6.11: Convert specular reflection intensity to depth image mask.

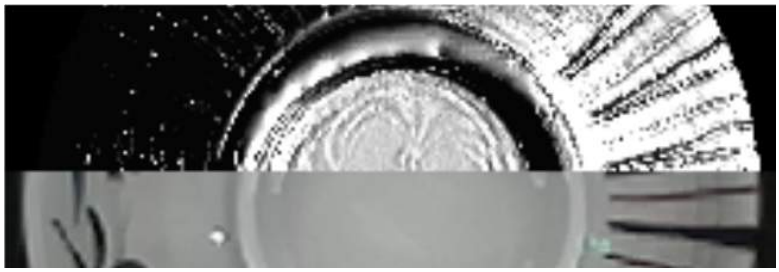
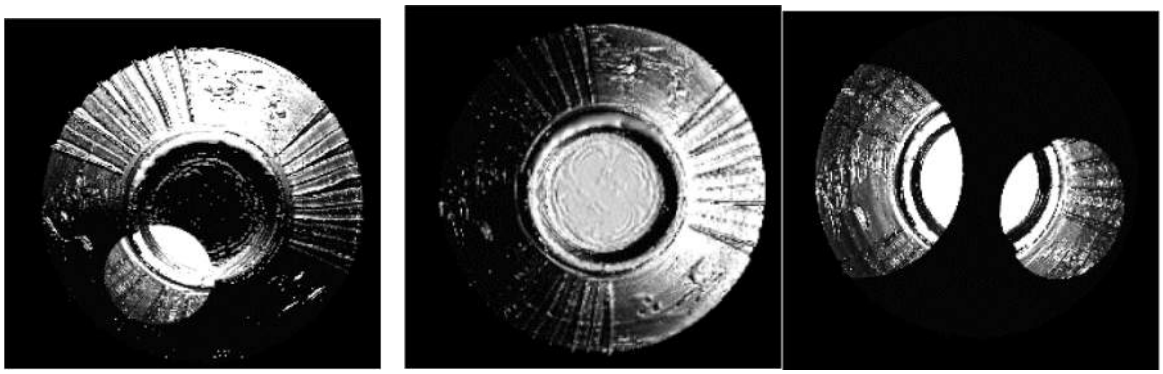


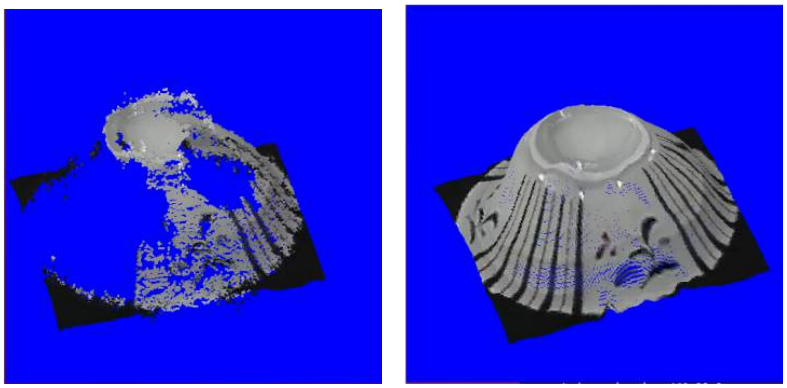
Figure 6.12: Synthetic(upper) and camera color image(lower).

The main PC used for program development and machine learning training had the following specifications: PC1 had a Xeon W-2135 CPU, 64GB of memory, and an Nvidia TITAN RTX 24GB GPU; PC2 had a Core i9 10900X CPU, 16GB of memory, and an NVIDIA GeForce RTX 2080Ti 11GB GPU. We also used the ABCI (AI bridging cloud) service for about 630 hours of machine learning training.



(a) Light 1. (b) Light 2. (c) Light 3.

Figure 6.13: Convert specular reflection intensity to depth image mask.



(a) (b)

Figure 6.14: Example of 3D point-cloud reconstruction.

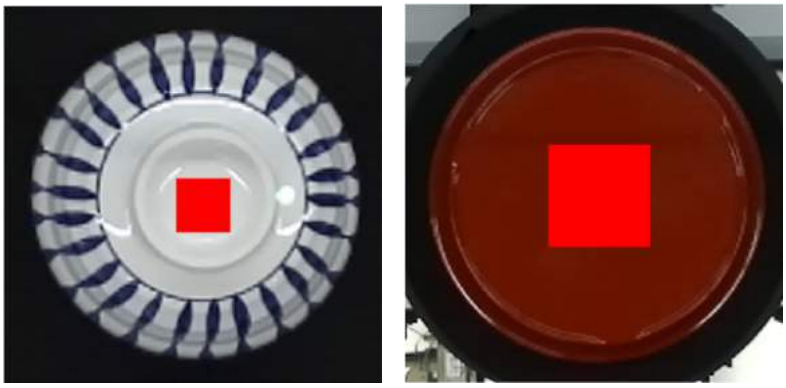
6.4.3 Experiments

To evaluate the performance of our system, we compared the result of reconstructing the depth image with defects to the true value of the depth image. We simulated specular reflection to reproduce the missing data by synthetic images of the dishes using only specularly reflected light as seen in Figure 6.11 (a). Then, we replaced the corresponding depth values in the depth image with invalid values for bright areas of luminance as seen in Figure 6.11 (a). At this time, the synthetic image and the image captured by the camera match pixel by pixel as seen in Figure 6.12 because the synthetic image was generated based on the camera parameters of the actual ZED camera. To simplify the problem, we assumed that the noise in the depth image occurred in the highly glossy portion of the dishware.

The program is expected to restore the defects at the adsorption point. Therefore, we used three types of lighting for the evaluation, illuminating the bottom of the dishes as seen in Figure 6.13. The first is a combination of parallel light that illuminates the sides of the dish and a spot light that illuminates the bottom. The second is parallel light that obliquely looks down on the dishes. The third is two spotlights that illuminate the bottom of the dish.

6.4.4 Results

We evaluated the performance of our implemented program by using twelve different types of dishes and measuring each dish in four different postures by changing the angle of the dish. Defective depth images were created from the evaluation data and restored using our program.



(a) (b)

Figure 6.15: Example of areas to assess.

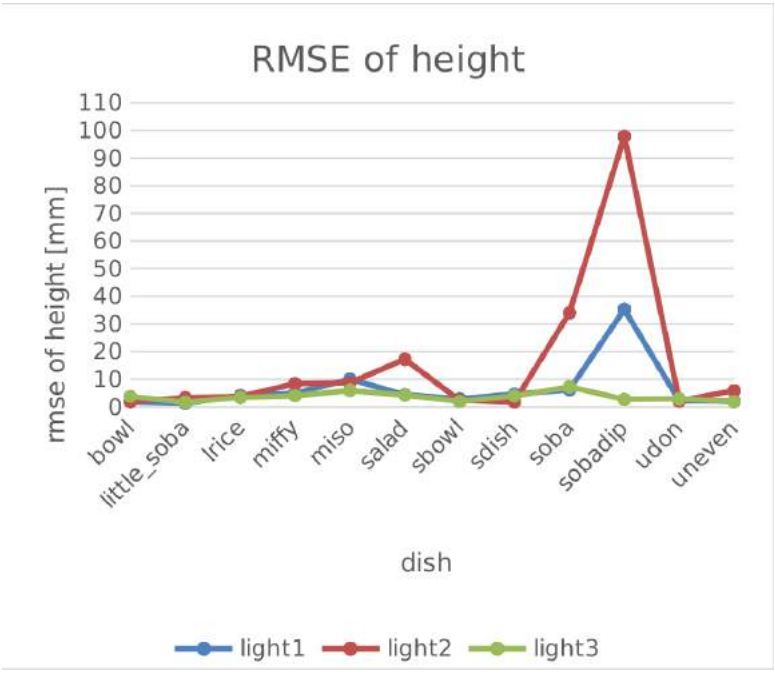


Figure 6.16: Height error.

The data used for evaluation is the same training data we used to train the network.

Figure 6.14 shows examples of 3D point clouds before and after restoration processing. Other examples of point-cloud restoration can be found in Figure 1 in the appendix.

As seen in Figure 6.14 (a), we can see that the surface of the 3D point cloud before restoration is missing because if the depth values in the original depth image are replaced with invalid values, the corresponding points are lost in the 3D point cloud. As seen in Figure 6.14 (b), after restoration the missing part is restored.

During the evaluation, we only focused on the dishes and not the entire image, and the background was removed. We also manually defined labels for evaluation, which were used to identify the parts of the dish that the robot arm could lift if it stuck to it. These areas were defined as a square ROI (Region of Interest) for easy labeling. Figure 6.15 shows an example of an ROI, with the red area being the evaluation target. The total number of 3D points to be evaluated for the twelve types of dishes and four postures is 286,572.

The error of the restored depth image compared to the true value depth image is shown in a graph, with the vertical axis being RMSE (root mean square error in millimeters) as seen in Figure 6.16. Most of the errors are less than 10mm, but some combinations of dishes and

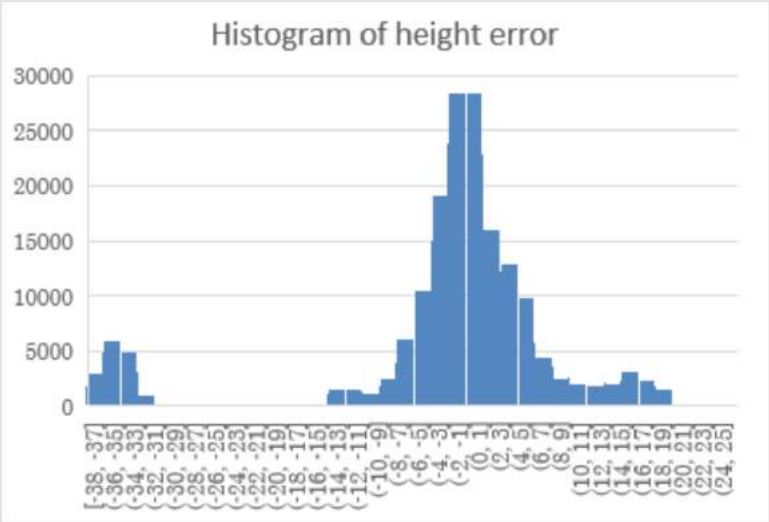


Figure 6.17: Height difference.

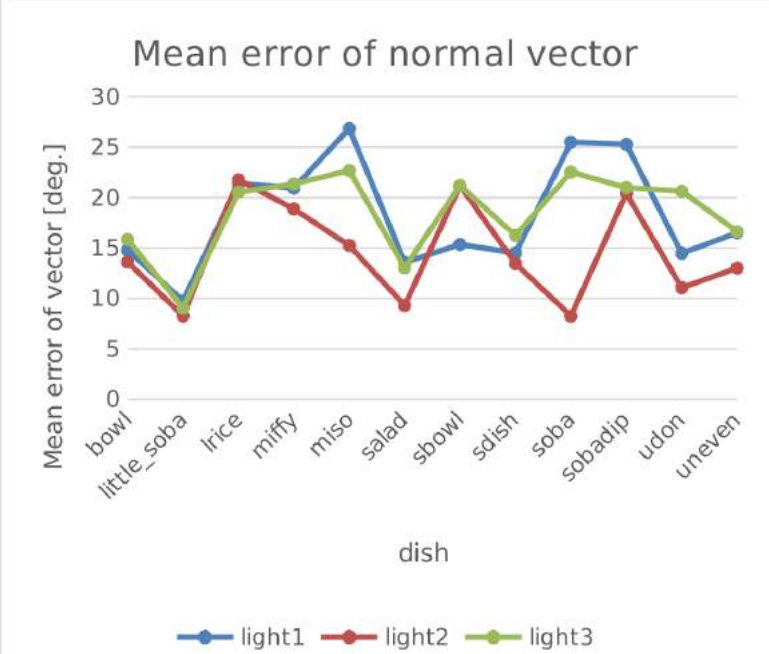


Figure 6.18: Average error of normal vector.

lighting conditions have larger errors.

A histogram of the height differences is shown in Fig. 6.17, which has clusters of outliers in the negative bins. This is due to the wrong dish height being reconstructed and is different from the errors around the true value. Outliers were isolated by using the 9th percentile of the overall error as the lower bound for the difference and analyzing the remaining points.

The Kolmogorov-Smirnov test showed that the height difference could not be considered a normal distribution, even with the outliers excluded. The Wilcoxon signed-rank test also showed that the null hypothesis (that the difference is symmetrical about 0) can be rejected, and the height difference distribution is not symmetrical about 0.

The RMSE, excluding outliers, was 6.1mm with a median of 1mm. The average height difference (height after reconstruction minus true height) was 1.7mm with a standard deviation of 5.9mm. The average processing time for restoration, not including program initialization and file input/output, was 0.95 seconds on PC1.

In the next evaluation, we focused on the error of the normal vector. The graph in Fig.

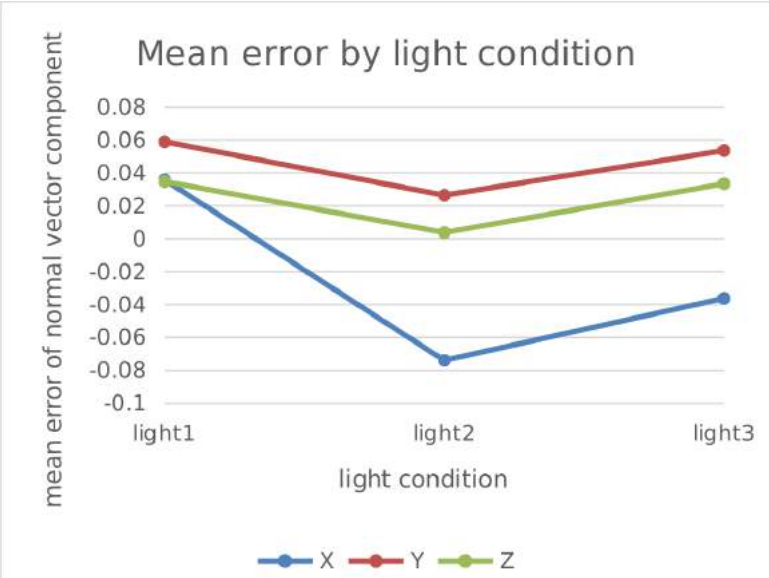


Figure 6.19: Average error of normal vector by x, y, z components.

6.18 shows the average value of the angle between the normal vector of the true value and the normal vector after restoration. This normal vector is obtained from the local surface, with a local plane size of 7x7 pixels centered on the point of interest in the depth image.

The average angle between the restored and true vectors ranges from 5 to 25 degrees. There is no significant difference in the combination of dishes and lighting conditions other than the difference in height.

To further analyze the normal vector error, we calculated the average difference for each of the X, Y, and Z components of the normal vector. The graph in Fig. 6.19 shows that the difference in the X component is negative under lighting conditions light2 and light3, meaning that the surface after restoration is tilted in the x-axis direction compared to the true value.

We also tried to detect the sticking point based on the normal vector using an algorithm that compares performance. This algorithm assumes that the less variation there is in the normal vector, the higher the possibility of adsorption. To use this algorithm, we first calculated the average value of the standard deviation of the normal vector for each three-dimensional point in the local region and ranked them in ascending order.

We used the L2 norm to calculate the difference between the vectors when calculating the standard deviation. We also narrowed down to 3D points within 30% of the highest, as the original algorithm does not consider height. Using this method, we determined that adsorption is possible within the top 1% of points with the smallest evaluation value and impossible for the others.

We labeled the points that can be adsorbed as correct and the points that cannot be adsorbed as incorrect. To judge whether the points were correct or incorrect, we used circular labels instead of square ROIs. We took the center of the ROI as the center of the circle and the diameter as the diagonal length of the ROI.

If a point predicted to be able to be adsorbed was correct, it was considered a true positive. If it was incorrect, it was considered a false positive. The accuracy rate for the dishes as a whole was 91.7% and the precision rate was 54.7%. When the same process was applied to the true value 3D point cloud, the accuracy rate was 92% and the precision rate was 83.9

In this experiment, the height error of restored dishes was found to be asymmetrical around 0, with median values ranging from 0 to 1 mm. This may be due to the inpainting and correction processes used, which can introduce systematic errors and flatten the height of the restored dishes. The error in the normal vector components varied depending on lighting conditions and

the direction of the vector.

The accuracy rate for predicting which dishes could and could not be adhered was 90%, but the precision rate was only 50%. Some individual dishes had compliance rates close to 80%. This algorithm tends to select points near the plane as adhesion points, and the restoration may be more unstable for dishes with larger damage on the bottom surface.

Possible causes for the errors in the height and normal vector values were identified as side effects of the depth value restoration and correction processes, as well as defects in the depth images used for restoration. Possible countermeasures include limiting the 3D points used to those close to a plane, and adding information specifying the desired shape to the restoration process.

6.5 Conclusion

In this chapter, we addressed the issue of harsh lighting conditions for service robots in a commercial kitchen environment. Specifically, we aimed to improve the detection of dishes and the estimation of their grasp position for an automated dishwasher. Harsh lighting conditions can cause missing data in the point cloud, making it difficult to accurately detect and grasp the dishes.

We tried several methods to address this issue, such as using a softbox to control lighting conditions, a lens cover with a polarizer, and changing the camera position. However, these methods were difficult to implement in a real-world environment or were slow. Our main proposed solution was using auto-encoders with partial convolutions to inpaint the missing depth information. However, this solution had the drawback of distorting the resulting image and producing inaccurate point cloud data. Additionally, it required a significant amount of GPU power for inference.

Overall, addressing the issue of harsh lighting conditions in a commercial kitchen environment is a challenging task. Our proposed solutions had drawbacks and limitations. Further research is needed to develop more effective methods to improve the detection and grasping of objects in harsh lighting conditions.

Chapter 7

Conclusions, Limitations and Future Work

This chapter summarizes, discusses and concludes the research findings and contributions of this thesis, which are grounded by the objectives and research questions defined in Chapter 1.

7.1 Discussion Summary

In Chapter 1, we introduced the problem statement of implementing robotics and AI in the real-world environment, particularly in the context of dishwashing in a restaurant kitchen using cobots and computer vision. We discussed the challenges of deploying service robots in real-world environments compared to industrial robots, and highlighted the importance of addressing issues such as robustness, data collection, and non-ideal lighting conditions. We also introduced our research questions, which aimed to explore ways to simplify existing deep-learning models for object detection and grasp position estimation, as well as ways to reduce the time and resources required for data collection. Through this chapter, we established the background and context for our research and underscored the importance of addressing these challenges in order to improve the deployment of service robots in real-world environments.

In Chapter 2, we introduced the system architecture and design concept of our dishwasher system for a commercial kitchen environment. We discussed the problem setting and the hardware components, including two cobots with additional rotational joints, a 3D camera, and a 2D camera. We also outlined the software components, such as Yolov3, the Robot Operating System, ZeroMQ, and the Nvidia Deep Learning Dataset Synthesizer (NDDS), which were used for object detection and grasp position estimation of dishes in a restaurant. Through this chapter, we presented a comprehensive overview of the system's design and its functionalities. We believe that this system will be able to provide an efficient and accurate solution for dishwashing in commercial kitchens and pave the way for further research and development in this field.

In Chapter 3, we focused on the research questions one, two and three. We presented a method for detecting the grasp position of a dishwasher in a commercial kitchen environment. Our approach uses a shallow network, which is efficient in terms of computational resources while still maintaining accuracy. To facilitate training, we also created a synthetic environment using the UE4 gaming engine and accurate 3D models. Our experimental results in a commercial kitchen environment showed that the proposed shallow network grasp detector, trained with only synthetic data, can accurately predict the grasp position. It is easy to use and requires fewer resources compared to conventional methods. In future work, we plan to improve the grasp detector by incorporating FGE to provide 3D coordinates for the grasp position and combining it with a linear shift grasp detector for fail-safe predictions.

In Chapter 4, we focused on research question four. We presented a method for data collection and annotation using mobile robots equipped with a tilt platform and invisible markers. Our approach allows us to capture images of objects from every possible location and angle using a combination of high-frame-rate image capture and switching between white light and UV light. The invisible markers in the UV-light-lit images provide annotation data during data collection. Our method is able to create large datasets that are consistent, realistic, and less time-consuming and expensive compared to synthetically generated data. As a result, these datasets are more useful in making deep learning models robust against real-world conditions. Additionally, our method can be used for other tasks such as grasp-position-estimation and key-point-estimation. Overall, our proposed method is a powerful and efficient approach for data collection and annotation in deep learning models.

In Chapter 5, we proposed using key-point detection to detect the 6D pose of circular dishes kept upside down in a stacked configuration. To overcome the problem of slow detection using classical methods, we made assumptions to reduce the problem setting and used a shallow network for quickly detecting keypoints. We borrowed the idea of using a shallow network from Chapter 3 and the idea of data collection from Chapter 4. However, our approach has the limitation of angle of tilt to discrete values which is less useful in real-world scenarios. This method is a valuable approach to quickly detect 6D pose of an object on limited resources but it has limitations that should be addressed in future research.

In Chapter 6, we focused on research question five. We attempted to address the issue of harsh lighting conditions for service robots in a commercial kitchen environment. We attempted several methods, such as using a softbox, a lens cover with polarizer and changing the camera position, but they were either hard to implement or slow. Our main proposed solution was using auto-encoders with partial convolutions but it had the drawback of distorting the resulting image and producing inaccurate point cloud data. It also required a significant amount of GPU power for inference. The harsh lighting condition is a challenging task and our proposed solutions had drawbacks and limitations, therefore, further research is needed to develop more effective methods to improve the detection and grasping of objects in harsh lighting conditions.

7.2 Limitations and Future Work

This section presents the current limitations of the presented research as well as how this limitation can be addressed in future research projects.

In conclusion of Chapter 3, we proposed a grasp detection method for the dishwasher in a commercial kitchen environment that relies on a shallow network and synthetic environment for training. While our experimental results show that the proposed method can predict grasp position accurately and requires fewer resources compared to conventional methods, it does have a few drawbacks. The approach heavily relies on synthetic data which may not fully capture the complexity and variability of real-world conditions. Additionally, it is limited in its ability to detect the grasp position when there are holes in the point cloud and has the limitation of angle of tilt to discrete values which is less useful in real-world scenarios. To address these limitations, possible future work could include incorporating real-world data in the training process, developing methods to detect grasp position in the presence of holes in the point cloud, and exploring other types of objects.

In conclusion of Chapter 4, we proposed a method for data collection and annotation using mobile robots equipped with a tilt platform and invisible markers. While our approach enables us to capture a large amount of real data and annotate it quickly, it is limited in its ability to capture images of objects from different perspectives or orientations, with multiple cameras. This could affect the generalization of the trained models other than the top-down view. To

address these limitations, possible future work could include developing methods to incorporate more diverse perspectives and orientations from multiple cameras in the data collection process.

In Chapter 5, we showed several approaches to solve the 6D pose estimation problem and their measure of deployability. As we presented in our own method of using keypoint detection to estimate the 6D pose, there are still several difficulties. Firstly our current method is only able to predict the angle and direction discretely. In practical use this would be insufficient.

In Chapter 5, we proposed using key-point detection to detect the 6D pose of an object. While this method could be efficient in terms of computational resources and able to quickly detect 6D pose of an object, it has the major drawback of limitation of the angle of tilt to discrete values which is not useful in real-world scenarios where continuous values would be more useful. As a future work, it is necessary to investigate ways to detect 6D pose in continuous values.

In Chapter 6, we proposed several methods to address the issue of harsh lighting conditions for service robots in a commercial kitchen environment. However, these methods had drawbacks such as being difficult to implement in a real-world environment or slow. Our main proposed solution of using auto-encoders with partial convolutions had the drawback of distorting the resulting image and producing inaccurate point cloud data and required significant amount of GPU power for inference. To address these limitations, possible future work could include research on improving the efficiency and accuracy of the in-painting method.

7.3 Conclusion

This thesis presented two published works aimed at improving the deployability of robotic systems with deep-learning technology in the real world. Grasp position estimation for a single suction gripper mentioned in Chapter 3 presented the problem of deploying an accurate and robust deep-learning system with limited resources. We showed how we can overcome this with shallow networks that are modular in nature and trained entirely on synthetic data.

In Chapter 4, we presented a novel approach using mobile robots and invisible markers to collect and annotate high quality real data. We showed how this approach not only increases the robustness of the model but also decreases the time to collect and annotate data.

In other chapters, we also discussed other problems facing effective use of deep-learning in robotic systems for practical deployment. The main contributions of this thesis are described below:

- A systematic identification of the gaps in the deep-learning research in academia and practical application.
- An in-depth understanding of the specifications required for a successful deployment of robotic systems using artificial intelligence.
- Development of a grasp position estimation network that is low compute intensive, modular and trained with only synthetic data.
- A unique approach to solve the data collection problem using mobile robots and invisible markers.
- A preliminary attempt on solving the 6D pose estimation problem for symmetric dishes using key-points.
- An in-depth study on addressing the issue of harsh lighting conditions using hardware and auto-encoders for in-painting.

Publications

The publications listed below were produced and accepted during this doctoral work.

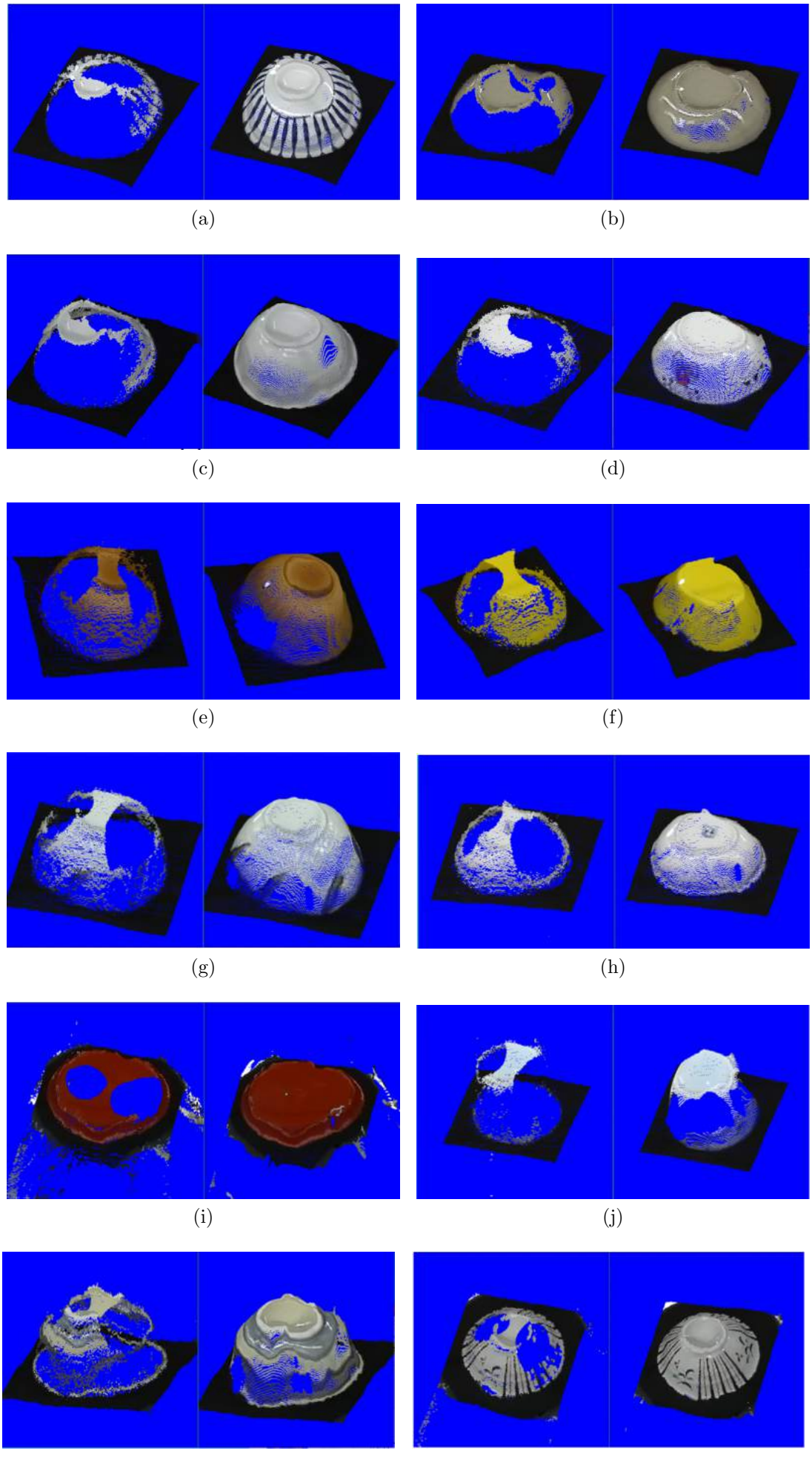
- **S. P. PATTAR**, T. HIRAKAWA, T. YAMASHITA, T. SAWANOBORI, and H. FUJIYOSHI. *Single Suction Grasp Detection for Symmetric Objects Using Shallow Networks Trained with Synthetic Data.* IEICE TRANSACTIONS on Information and Systems, Vol.E105-D, No.9, pp.1600-1609, 2022.

- [Accepted] **S. P. PATTAR**, T. KILLUS, T. HIRAKAWA, T. YAMASHITA, T. SAWANOBORI, and H. FUJIYOSHI. *Automatic Data Collection for Object Detection and Grasp-position Estimation with Mobile Robots and Invisible Markers.* Advanced Robotics, 2022.

Other publications:

- W. S. Lo, C. Yamamoto, **S. P. Pattar**, K. Tsukamoto, S. Takahashi, T. Sawanobori, and I. Mizuuchi. *Developing a Collaborative Robotic Dishwasher Cell System for Restaurants.* in International Conference on Intelligent Autonomous Systems. Springer, 2022, pp. 261–275.

Appendices



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

Figure 1: Examples of point-cloud restorations.

Bibliography

[1] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 969–977.

[2] International Federation of Robotics (IFR), "World robotics industrial robots 2022." [Online]. Available: https://ifr.org/

[3] K. Kosuge, Y. Hirata, J. Lee, A. Kawamura, K. Hashimoto, S. Kagami, Y. Hayashi, N. Yokoshima, H. Miyazawa, R. Teranaka, Y. Natsuizaka, K. Sakai, M. Koizumi, J. Koyama, N. Kanayama, S. Tezuka, and H. Torimitsu, "Development of an automatic dishwashing robot system," in *2009 International Conference on Mechatronics and Automation*, 2009, pp. 43–48.

[4] W. S. Lo, C. Yamamoto, S. P. Pattar, K. Tsukamoto, S. Takahashi, T. Sawanobori, and I. Mizuuchi, "Developing a collaborative robotic dishwasher cell system for restaurants," in *International Conference on Intelligent Autonomous Systems*. Springer, 2022, pp. 261–275.

[5] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[6] Connected Robotics Inc., "Dishwasher robot." [Online]. Available: https://connected-robotics.com/products/dishwashing-bowl/

[7] S. S. H. Hajjaj and K. S. M. Sahari, "Review of agriculture robotics: Practicality and feasibility," in *2016 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, 2016, pp. 194–198.

[8] IEEE, "Technical committee for robotics research for practicality." [Online]. Available: https://www.ieee-ras.org/robotics-research-for-practicality

[9] R. Young, "Robotics in the real world: the perceptual control theory approach," in *The Interdisciplinary Handbook of Perceptual Control Theory*. Elsevier, 2020, pp. 517–556.

[10] S. W. Bennett and G. F. DeJong, "Real-world robotics: Learning to plan for robust execution," *Machine Learning*, vol. 23, no. 2, pp. 121–161, 1996.

[11] M. Schneier, M. Schneier, and R. Bostelman, *Literature review of mobile robots for manufacturing*. US Department of Commerce, National Institute of Standards and Technology, 2015.

[12] R. Colbaugh and M. Jamshidi, "Robot manipulator control for hazardous waste-handling applications," *Journal of Robotic Systems*, vol. 9, no. 2, pp. 215–250, 1992.

[13] L. Pedersen, D. Kortenkamp, D. Wettergreen, I. Nourbakhsh, and D. Korsmeyer, "A survey of space robotics," in *ISAIRAS*, 2003.

[14] G. O'Regan, "Unimation," in *Pillars of Computing*. Springer, 2015, pp. 219–223.

[15] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *ACM Computing Surveys (CSUR)*, vol. 18, no. 1, pp. 67–108, 1986.

[16] A. Pugh, *Robot vision*. Springer Science & Business Media, 2013.

[17] Z. Liu, B. Zhao, and H. Zhu, "Research of sorting technology based on industrial robot of machine vision," in *2012 Fifth International Symposium on Computational Intelligence and Design*, vol. 1. IEEE, 2012, pp. 57–61.

[18] K. G. Harding, "Machine vision—lighting," *Encyclopedia of Optical Engineering*, vol. 2, pp. 1227–1239, 2003.

[19] C. Li, Y. Ma, S. Wang, and F. Qiao, "Novel industrial robot sorting technology based on machine vision," in *2017 9th International Conference on Modelling, Identification and Control (ICMIC)*, 2017, pp. 902–907.

[20] J.-K. Oh and C.-H. Lee, "Development of a stereo vision system for industrial robots," in *2007 International Conference on Control, Automation and Systems*. IEEE, 2007, pp. 659–663.

[21] J. H. Jung and D.-G. Lim, "Industrial robots, employment growth, and labor cost: A simultaneous equation analysis," *Technological Forecasting and Social Change*, vol. 159, p. 120202, 2020.

[22] International Organization for Standardization (ISO), "ISO 8373:2021," 2021.

[23] Y. Fukuzawa, Z. Wang, Y. Mori, and S. Kawamura, "A robotic system capable of recognition, grasping, and suction for dishwashing automation," in *2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, 2021, pp. 369–374.

[24] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1957–1964.

[25] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[26] M. Mazumder, C. Banbury, X. Yao, B. Karlaš, W. G. Rojas, S. Diamos, G. Diamos, L. He, D. Kiela, D. Jurado *et al.*, "Dataperf: Benchmarks for data-centric ai development," *arXiv preprint arXiv:2207.10062*, 2022.

[27] A. Mayr, D. Kißkalt, M. Meiners, B. Lutz, F. Schäfer, R. Seidel, A. Selmaier, J. Fuchs, M. Metzner, A. Blank *et al.*, "Machine learning in production–potentials, challenges and exemplary applications," *Procedia CIRP*, vol. 86, pp. 49–54, 2019.

[28] S. E. Whang, Y. Roh, H. Song, and J.-G. Lee, "Data Collection and Quality Challenges in Deep Learning: A Data-Centric AI Perspective," *arXiv preprint arXiv:2112.06409*, 2021.

[29] Tokyo University of Agriculture and Technology and Connected Robotics Inc., "Container robotic system and robotic hand JP2022124023A." [Online]. Available: https://patents.google.com/patent/JP2022124023A/en

[30] A. Farhadi and J. Redmon, "Yolov3: An incremental improvement," *Computer Vision and Pattern Recognition, cite as*, 2018.

[31] P. Hintjens, *ØMQ - The Guide*. ZeroMQ Org, 2012.

[32] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, W. Hodge, and S. Birchfield, "NDDS: NVIDIA deep learning dataset synthesizer," 2018, https://github.com/NVIDIA/Dataset_Synthesizer.

[33] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.

[34] P. Bezak, P. Bozek, and Y. Nikitin, "Advanced robotic grasping system using deep learning," *Procedia Engineering*, vol. 96, pp. 10–20, 2014.

[35] J. Watson, J. Hughes, and F. Iida, "Real-world, real-time robotic grasping with convolutional neural networks," in *Annual Conference Towards Autonomous Robotic Systems*. Springer, 2017, pp. 617–626.

[36] A. Hentout, M. Aouache, A. Maoudj *et al.*, "Key challenges and open issues of industrial collaborative robotics," in *2018 The 27th IEEE International Symposium on Workshop on Human-Robot Interaction: from Service to Industry (HRI-SI2018) at Robot and Human Interactive Communication. Proceedings. IEEE*, 2018.

[37] Epic Games, "Unreal engine." [Online]. Available: https://www.unrealengine.com

[38] S. A. Stansfield, "Robotic grasping of unknown objects: A knowledge-based approach," *The International journal of robotics research*, vol. 10, no. 4, pp. 314–326, 1991.

[39] R. Pelossof, A. Miller, P. Allen, and T. Jebara, "An svm learning approach to robotic grasping," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 4. IEEE, 2004, pp. 3512–3518.

[40] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1316–1322.

[41] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[42] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 769–776.

[43] X. Zhou, X. Lan, H. Zhang, Z. Tian, Y. Zhang, and N. Zheng, "Fully convolutional grasp detection network with oriented anchor box," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7223–7230.

[44] P. Jiang, Y. Ishihara, N. Sugiyama, J. Oaki, S. Tokura, A. Sugahara, and A. Ogawa, "Depth image–based deep learning of grasp planning for textureless planar-faced objects in vision-guided robotic bin-picking," *Sensors*, vol. 20, no. 3, p. 706, 2020.

[45] M. Schwarz, C. Lenz, G. M. García, S. Koo, A. S. Periyasamy, M. Schreiber, and S. Behnke, "Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3347–3354.

[46] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Analysis and observations from the first amazon picking challenge," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 172–188, 2016.

[47] C. Hernandez, M. Bharatheesha, W. Ko, H. Gaiser, J. Tan, K. van Deurzen, M. de Vries, B. Van Mil, J. van Egmond, R. Burger *et al.*, "Team delft's robot winner of the amazon picking challenge 2016," in *Robot World Cup*. Springer, 2016, pp. 613–624.

[48] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning," in *2018 IEEE International Conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 5620–5627.

[49] K. Mano, T. Hasegawa, T. Yamashita, H. Fujiyoshi, and Y. Domae, "Fast and precise detection of object grasping positions with eigenvalue templates," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4403–4409.

[50] Y. Xu, L. Wang, A. Yang, and L. Chen, "Graspcnn: Real-time grasp detection using a new oriented diameter circle representation," *IEEE Access*, vol. 7, pp. 159 322–159 331, 2019.

[51] N. Mellado, D. Aiger, and N. J. Mitra, "Super 4pcs fast global pointcloud registration via smart indexing," in *Computer graphics forum*, vol. 33, no. 5. Wiley Online Library, 2014, pp. 205–215.

[52] Z. Wang, Z. Li, B. Wang, and H. Liu, "Robot grasp detection using multimodal deep convolutional neural networks," *Advances in Mechanical Engineering*, vol. 8, no. 9, p. 1687814016668077, 2016.

[53] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[54] S. Caldera, A. Rassau, and D. Chai, "Review of deep learning methods in robotic grasp detection," *Multimodal Technologies and Interaction*, vol. 2, no. 3, p. 57, 2018.

[55] M. Denninger, M. Sundermeyer, D. Winkelbauer, D. Olefir, T. Hodan, Y. Zidan, M. Elbadrawy, M. Knauer, H. Katam, and A. Lodhi, "Blenderproc: Reducing the reality gap with photorealistic rendering," in *International Conference on Robotics: Sciene and Systems, RSS 2020*, 2020.

[56] S. I. Nikolenko, "Synthetic data for deep learning," *arXiv preprint arXiv:1909.11512*, 2019.

[57] A. Dehban, J. Borrego, R. Figueiredo, P. Moreno, A. Bernardino, and J. Santos-Victor, "The impact of domain randomization on object detection: A case study on parametric shapes and synthetic textures," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2593–2600.

[58] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3828–3836.

[59] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[60] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[61] J. Yu, K. Weng, G. Liang, and G. Xie, "A vision-based robotic grasping system using deep learning for 3d object recognition and pose estimation," in *2013 IEEE international conference on robotics and biomimetics (ROBIO)*. IEEE, 2013, pp. 1175–1180.

[62] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 1386–1383.

[63] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," 2018.

[64] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1521–1529.

[65] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[66] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[67] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis *et al.*, "A large-scale benchmark dataset for event recognition in surveillance video," in *CVPR 2011*. IEEE, 2011, pp. 3153–3160.

[68] A. Sorokin and D. Forsyth, "Utility data annotation with amazon mechanical turk," in *2008 IEEE computer society conference on computer vision and pattern recognition workshops*. IEEE, 2008, pp. 1–8.

[69] H. Su, J. Deng, and L. Fei-Fei, "Crowdsourcing annotations for visual object detection," in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[70] A. Yao, J. Gall, C. Leistner, and L. Van Gool, "Interactive object detection," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3242–3249.

[71] D. De Gregorio, A. Tonioni, G. Palli, and L. Di Stefano, "Semiautomatic labeling for deep learning in robotics," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 611–620, 2019.

[72] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1179–1185, 2016.

[73] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: a large-scale benchmark for general object grasping," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 444–11 453.

[74] C. Mitash, K. E. Bekris, and A. Boularias, "A self-supervised learning system for object detection using physics simulation and multi-view pose estimation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 545–551.

[75] B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, O. Lehmann, T. Chen, A. Hutter, S. Zakharov, H. Kosch *et al.*, "Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition," in *2017 International Conference on 3D Vision (3DV)*. IEEE, 2017, pp. 1–10.

[76] T. Hodaň, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. N. Sinha, and B. Guenter, "Photorealistic image synthesis for object instance detection," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 66–70.

[77] M. Sela, P. Xu, J. He, V. Navalpakkam, and D. Lagun, "Gazegan - unpaired adversarial image generation for gaze estimation," *CoRR*, vol. abs/1711.09767, 2017. [Online]. Available: http://arxiv.org/abs/1711.09767

[78] J. Borrego, A. Dehban, R. Figueiredo, P. Moreno, A. Bernardino, and J. Santos-Victor, "Applying domain randomization to synthetic data for object category detection," *CoRR*, vol. abs/1807.09834, 2018. [Online]. Available: http://arxiv.org/abs/1807.09834

[79] B. Adhikari, J. Peltomaki, J. Puura, and H. Huttunen, "Faster bounding box annotation for object detection in indoor scenes," in *2018 7th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 2018, pp. 1–6.

[80] B. Adhikari and H. Huttunen, "Iterative bounding box annotation for object detection," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 4040–4046.

[81] Y. Chen, L. Liu, J. Tao, X. Chen, R. Xia, Q. Zhang, J. Xiong, K. Yang, and J. Xie, "The image annotation algorithm using convolutional features from intermediate layer of deep learning," *Multimedia Tools and Applications*, vol. 80, no. 3, pp. 4237–4261, 2021.

[82] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3511–3516.

[83] X. Yan, J. Hsu, M. Khansari, Y. Bai, A. Pathak, A. Gupta, J. Davidson, and H. Lee, "Learning 6-dof grasping interaction via deep geometry-aware 3d representations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3766–3773.

[84] H. Cao, H.-S. Fang, W. Liu, and C. Lu, "Suctionnet-1billion: A large-scale benchmark for suction grasping," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8718–8725, 2021.

[85] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International journal of computer vision*, vol. 92, no. 1, pp. 1–31, 2011.

[86] K. Takahashi and K. Yonekura, "Invisible marker: Automatic annotation of segmentation masks for object manipulation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 8431–8438.

[87] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, 2021.

[88] Z. He, W. Feng, X. Zhao, and Y. Lv, "6d pose estimation of objects: Recent technologies and challenges," *Applied Sciences*, vol. 11, no. 1, p. 228, 2020.

[89] W. Zhao, S. Zhang, Z. Guan, W. Zhao, J. Peng, and J. Fan, "Learning deep network for detecting 3d object keypoints and 6d poses," in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2020, pp. 14 134–14 142.

[90] S. Zakharov, I. Shugurov, and S. Ilic, "Dpod: 6d pose object detector and refiner," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1941–1950.

[91] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 3212–3217.

[92] Microsoft, "LightGBM: Light gradient-based machine," 2016, https://github.com/microsoft/LightGBM.

[93] J. Skinner, S. Garg, N. Sünderhauf, P. Corke, B. Upcroft, and M. Milford, "High-fidelity simulation for evaluating robotic vision performance," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2737–2744.

[94] L. Nalpantidis and A. Gasteratos, "Stereo vision for robotic applications in the presence of non-ideal lighting conditions," *Image and Vision Computing*, vol. 28, no. 6, pp. 940–951, 2010.

[95] G. Klančar, M. Kristan, and R. Karba, "Wide-angle camera distortions and non-uniform illumination in mobile robot tracking," *Robotics and Autonomous Systems*, vol. 46, no. 2, pp. 125–133, 2004.

[96] A. Hogue, A. German, and M. Jenkin, "Underwater environment reconstruction using stereo and inertial data," in *2007 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2007, pp. 2372–2377.

[97] M. Ahmed, K. A. Hashmi, A. Pagani, M. Liwicki, D. Stricker, and M. Z. Afzal, "Survey and performance analysis of deep learning based object detection in challenging environments," *Sensors*, vol. 21, no. 15, p. 5116, 2021.

[98] M. B. Jensen, K. Nasrollahi, and T. B. Moeslund, "Evaluating state-of-the-art object detector on challenging traffic light data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 9–15.

[99] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2. IEEE, 2004, pp. II–104.

[100] Y. Zhou, H. Nejati, T.-T. Do, N.-M. Cheung, and L. Cheah, "Image-based vehicle analysis using deep neural network: A systematic study," in *2016 IEEE international conference on digital signal processing (DSP)*. IEEE, 2016, pp. 276–280.

[101] R. A. Hamzah and H. Ibrahim, "Literature survey on stereo vision disparity map algorithms," *Journal of Sensors*, vol. 2016, 2016.

[102] A. O'Riordan, T. Newe, G. Dooly, and D. Toal, "Stereo vision sensing: Review of existing systems," in *2018 12th International Conference on Sensing Technology (ICST)*. IEEE, 2018, pp. 178–184.

[103] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[104] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: A neural network framework for dimensionality reduction," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 490–497.

[105] S. Wang, H. Chen, L. Wu, and J. Wang, "A novel smart meter data compression method via stacked convolutional sparse auto-encoder," *International Journal of Electrical Power & Energy Systems*, vol. 118, p. 105761, 2020.