Tampere University

Mikael Petäjä

NATURAL LANGUAGE PATHFINDING FOR INDUSTRIAL APPLICATIONS ON A COLLABORATIVE ROBOT

Application of CLIPort for OpenDR

Master's Thesis
Faculty of Engineering and Natural Sciences
Examiners: Roel Pieters
Alexandre Angleraud
November 2023

ABSTRACT

Mikael Petäjä: Natural Language Pathfinding for Industrial Applications on a Collaborative Robot
Master's Thesis
Tampere University
Master of Science (Technology), Degree Programme in Engineering Sciences, Master's Programme in Automation Engineering
November 2023

Robotics and automation are common in modern industry but is often limited to pre-known workspaces and rigid tasks. With machine learning robotics can be made to derive task-relevant context from the workspace and act without explicit directions. This could benefit task robustness as well as allow for greater co-operation with humans in collaborative tasks.

Machine learning applications are particularly interesting in human-robot collaboration because of the difficulty of predicting the impact a human actor has on the workspace using traditional algorithms. The framework presented in this thesis attempts to react to changes in the workspace but does not directly detect human behaviour or attempt to avoid human body parts by identifying them in the workspace.

This thesis implements a CLIPort based natural language instruction tool for controlling a Franka Panda robotic arm. A final dataset-model -pair introduced as well as two prototype dataset-model -pairs for possible future improvements and development process explanation. A literature review was done to briefly discuss similar systems and other applications of machine learning on robotics. Backend systems, such as CLIP and CLIPort are briefly introduced as well along with other relevant works.

Results show that the presented model can achieve a location accuracy of 90.42% in examined industrial tasks. For certain object-task-environment configurations, this accuracy was observed to be up to 100.00%, but the overall the framework was found to successfully execute a complete pick & place task 75.07% of the time. The datasets with which the models were trained are examined and future improvements are considered with suggestions based on scope.

The most important contribution of the thesis is the demonstration that the implemented framework is suitable to industrial task execution. Other notable contributions include the identification of error-producing situations, and format and quantity recommendations for demonstrations in datasets.

Keywords: robotics, language instruction, industry, language, collaborative robotics, Franka Panda, real-robot, engine assembly, machine learning, CLIPort, object manipulation, visual servoing, language recognition, pose estimation, RGB-D, depth imaging, RGB-D object detection

TIIVISTELMÄ

Mikael Petäjä: Luonnollisen kielen avulla tapahtuva polunmääritys teollisia sovelluksia varten yhteistoiminnallisella robotilla
Diplomityö
Tampereen yliopisto
Automaatiotekniikan DI-ohjelma
Marraskuu 2023

Robotiikka ja automaatio on tullut yleiseksi nykypäiväisessä teollisuudessa, mutta rajoittuu vielä ennalta tunnettuihin työtiloihin ja suppeisiin tehtäviin. Koneoppimisen avulla robotin voi saada päättelemään tehtävän kannalta oleellisen kontekstin työtilasta ja toimimaan ilman eksplisiittisiä ohjeita. Tämä voi auttaa tehtävien toistettavuudessa tai suorituskestävyydessä ja mahdollistaa paremman yhteistoiminnan tehtävissä ihmisten kanssa.

Koneoppimisen sovellukset ovat erityisen mielenkiintoisia ihmisen ja robotin välisessä yhteistyössä siksi, että ihmistoimijan vaikutusta työympäristöön on erittäin vaikeaa ennustaa perinteisillä algoritmeilla. Tässä työssä esitelty järjestelmä pyrkii reagoimaan muutoksiin työympäristössä, mutta ei suoraan tunnista ihmisen käyttäytymistä tai pyri välttämään ihmisen ruumiinosia tunnistamalla niitä työympäristöstä.

Tässä työssä toteutetaan CLIPort-pohjainen puhutun kielen opastustyökalu Franka Panda -robottikäsivarren ohjaamiseen. Tässä työssä esitellään toteutettu tietokanta-mallipari, sekä kehitysprosessin selvennystä varten kaksi tutkintaan käytettyä tietokanta-malliparia. Työn kirjallisessa katsauksessa tarkastellaan samankaltaisia järjestelmiä ja muita nykyaikaisia koneoppimisen sovelluksia robotiikassa. Jotkin työn kannalta oleelliset taustajärjestelmät, kuten CLIP ja CLIPort esitellään muiden oleellisten töiden ohella.

Tuloksista nähdään, että esitelty malli voi saavuttaa 90.42% paikkatarkkuuden valituissa teollisuustehtävissä. Joillain esine-tehtävä-ympäristökonfiguraatioilla, tämä tarkkuus oli jopa 100%, mutta yleisesti ottaen toteutettu järjestelmä suorittaa kokonaisen poimimis- ja sijoitustehtävän 75.07% onnistumissuhteella. Mallien koulutukseen käytetyt tietokokonaisuudet esitellään ja tulevia parannuksia tarkastellaan toteutuslaajuuteen perustuvin ehdotuksin.

Työn tärkein tuotos on toteutetun järjestelmän soveltuvuuden osoittaminen teollisissa tehtävissä. Muita oleellisia löydöksiä ovat esimerkiksi virheitä tuottavien tilanteiden tunnistaminen sekä tietokokonaisuuksissa esiintyvien esimerkkien muoto- ja määräsuositukset.

Avainsanat: robotiikka, kielenopetus, teollisuus, kieli, yhteistoiminnallinen robotiikka, Franka Panda, oikea robotti, moottorin kokoonpano, koneoppiminen, CLIPort, objektien manipulointi, visuaalinen servoilu, kielen tunnistaminen, asennon arviointi, RGB-D, syvyyskuvaus, RGB-D-objektien tunnistus

PREFACE

This subject was given to me as a first assignment and orientation at the university which contributed to its long writing time. Significant complications were faced before real work ever began and I want to extend my thanks to my direct employer and supervisor, Roel Pieters for being both patient and supportive even when results still seemed distant.

I learned quite a few new things in the process of writing this thesis and in the end I think I achieved the results I hoped for even though the whole process took much longer than anticipated. This was also my first time writing such a document in LaTeX. I'd also like to extend my thanks to Akif Ekrekli, Alexandre Angleraud, Eetu Airaksinen and Gaurang Sharma for assisting me with setup related tasks where I had uncertainties. I also want to thank Omar Hassan for his previous work and assistance without which this thesis would not have been possible.

Tampere, 14th November 2023

Mikael Petäjä

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

ASR	Automatic Speech Recognition
GPU	Graphics Processing Unit
GUI	Graphical User Interface
h	Hour
HRC	Human-Robot Collaboration
IL	Imitation Learning
PC	Personal Computer
RGB	Red-Green-Blue
RGB-D	RGB with depth
RGBD	RGB with depth
TAMP	Task and Motion Planning
USB	Universal Serial Bus
VAD	Voice Activity Detection
VLM	Vision-and-Language Manipulation
VQA	Visual-Question-Answering

1. INTRODUCTION

Robotics have been a component of modern manufacturing ever since the 1980s, but it has proven to be dangerous and injurious when in close contact with humans. To better protect the workers in a factory with a robotic component in the assembly line, the robots are often separated from human workers. This has proven to be a successful solution to the safety concerns, but research has shown that factories with robots have an increased injury rate compared to non-robotic factories.

In modern robotics there exist potential solutions, such as using cobots. A cobot (compound for "collaborative" and "robot") is a robot specifically designed to work in tasks where human collaboration is necessary or desired, which can still include most manufacturing tasks due to the limited autonomy and decision-making capacity of modern computers. Cobots are designed to interact with the world in such a way that they pose a decreased chance of injury for humans. This safety comes with some limitations to the robot which can be detrimental to manufacturing, such as decreased movement velocity, which is why their implementation in manufacturing is still not common. Hybrid tasks that require both human problem solving, and the strength and repeatability of robotic assembly are also not a common paradigm compared to the more traditional "dumb" robotics assembly with separated human and robotic actors.

This thesis extends on a previous work done by Hassan O. [1], which introduced a CLIPort [2] based solution for executing tasks on a cobot based on language instructions. The main goal of this thesis is to show that this framework can be used for executing industrial tasks with sufficient success rates. The industrial tasks considered in the thesis come from engine assembly but are somewhat limited due to a focus on proving suitability instead of demonstrating actual engine assembly. Future works with the framework could be conducted on real engine assembly tasks with steps where human and robot actors would be truly beneficial, such as in precision insertion of pistons.

1.1 Background

Robotics is not the only means by which digital instructions can manipulate items in physical space. Other actuators such as heating elements or other radiators can also cause changes in the physical space, but these are often inaccurate or non-repeatable. Re-

peatability, strength, accuracy, and speed can be considered the most relevant traits to industry in robotics. These traits have already made robotics commonplace in modern industry, but mobile robots are also making appearances outside of manufacturing, such as mobile TUG delivery robots in hospitals [3] and in the streets of cities in Starship food delivery robots [4]. Traditional industrial robots usually only work with tasks that have a very low variance between repetitions, which can limit their applications. Typical industrial solutions usually can't react to changes in the workspace, such as a displaced object. This is why most robots in manufacturing utilize some sort of component feeding mechanism, which places the task objects in the same position for every repeated action sequence.

With the advent of machine learning and image space object classification, robots can be made to react to variances in their workspace. These solutions are still not commonplace in manufacturing compared to the cheaper and more proven industrial robots and are better suited for some tasks than others. Machine learning can also slow the process since predictions are usually not instant, so the value proposition of perception has to save time or money in some other form, such as ease of use, ease of installation or cost of operation. In a closed workspace, the variance between tasks can be minimal since the robots are the only actors and do not make mistakes except in accidents. When collaborating with a human on a manufacturing task, the workspace can't be closed off and the task variance is higher since the human is an unpredictable actor. This makes machine learning based solutions very suitable for collaborative robotics.

Speech is another problem with robots collaborating with humans. Robots can be programmed to collaborate with each other in manufacturing tasks, and they share data with modern communication protocols. Humans communicate mainly using speech, expressions, and gestures, which can't be directly transformed into machine-understood language without specialized tools, and even then understanding the messages usually relies heavily on environmental context and implied details. Natural language processing is a machine learning paradigm focusing on transforming human language into machine-understood language. Human reactions can also be relatively quick when compared with the speed of language processing, which often makes language-integrated robotics often seem slow to react. One of the aims of this thesis is to improve the natural language processing capacity of the previously implemented framework.

1.2 Problem Statement

Given the prior pipeline for generating what & where pathways, assess suitability for industrial tasks in engine assembly and compare the results to other models.

1.3 Research scope and objectives

The primary scope of the thesis is to test, assess, and possibly improve a previously developed solution (CLIPort nlihrc) and compare it to other solutions. This scope is then further split into the following research goals:

- Assess the accuracy of the speech model in CLIPort nlihrc.
- Assess the prediction accuracy of CLIPort nlihrc using multiple models with different samples and amount of training epochs.
- Generate dataset(s) and demonstrate suitability for industrial objects.
- potentially useful tasks for industrial applications using an existing solution back end.
- Research similar systems
- Compare CLIPort nlihrc results with results from similar systems

1.4 Thesis contribution

This project is mostly about testing the previously developed solution, and thus won't consider the development of completely new solutions. To facilitate model testing, the thesis work does include the extension of existing tools and possible development of tertiary tools. In the planning phase, several new requirements were found, and the thesis implements the following:

- A comparison pipeline for multiple trained models.
- General fixes to the existing solution without changing the prediction pipeline.
- Functional models that can be used to execute relevant tasks on industrial objects.
- Datasets that can be used to train the aforementioned functional models.
- Integration of existing solution with modern TensorFlow and CUDA (SM 8.6 compute compatibility).

The last point listed above (integration) isn't of interest to the thesis, regardless of its necessity, and won't be discussed further on the grounds of implementational irrelevance.

1.5 Thesis organization

This thesis is organized into 6 main chapters, the first of which being this introduction. The chapters to follow are:

- **Literature review:** This chapter discusses similar solutions and necessary theory. Because this thesis is based on the work in the thesis by O. Hassan [1], most

implementation-related theory will not be re-discussed, such as robotic manipulation, machine learning, camera calibration, or neural networks.

- **Experiment setup and changes to previous work:** This chapter discusses improvements and changes compared to the previous implementation presented in the thesis by O. Hassan [1]. This chapter also includes some theory verification, such as motion mapping.
- **Dataset format and purposes:** This chapter discusses the parameters used to collect the data sets. These data sets were used in training the models on which the success of industrial tasks was measured. This chapter also discusses why certain objects are present and their relative configurations.
- **Results:** This chapter discusses the success rates of the framework in relevant tasks that were considered industrial. Future works and improvement possibilities are also discussed.
- **Conclusions:** This chapter concludes the thesis and shows how research objectives were fulfilled. A summary is given in conclusion.

2. LITERATURE REVIEW

This chapter discusses similar works and potential future options to CLIPort for backend tests. The chapter also gives an overview of CLIP and CLIPort.

2.1 Collaborative robots in industry

Human-robot collaboration (HRC) in industry refers to an environment, where humans can work with robots in the in close proximity according to Wang et. al. [5]. Some benefits of HRC are increased quality of the product and increased manufacturing speed. Collaborative robots can also be cheaper to implement in small- and medium-sized enterprises due to the reduced required expertise and lower installation expenses according to Matheson et. al. [6]. The main purpose and the goal of HRC is to combine the accuracy, strength and repeatability of robotic actions with the problem solving and solving skills of humans [5]. According to [5] and [6], HRC implementations can be collapsed into four different categories as follows:

- *Coexistence*, when a robot and human are in the same physical space but have different workspaces. The work object or task might change workspaces so that independent collaboration is possible.
- *Interaction*, when one party guides the other without any physical contact. The task can be the same but is completed sequentially.
- *Cooperation*, when human and robot agents are autonomous in their tasks but can share cognitive or computational resources to aid each other. They can work independently, but sometimes have to wait for the availability of the other agents.
- *Collaboration*, when the human and the robot make a coordinated and joint effort to complete a shared task. Physical contact is also allowed unlike in the previous cases.

Figure 2.1 from the paper [6] presents these collaboration types in addition with the industry standard separated robot work cell configuration. Along with these, there is also the concept of symbiotic HRC, in which the human and robot co-exist and interact with each other in order to solve difficult tasks. Together they form a symbiotic HRC system that possesses the capability to assess situations and have reasoning to solve tasks while having

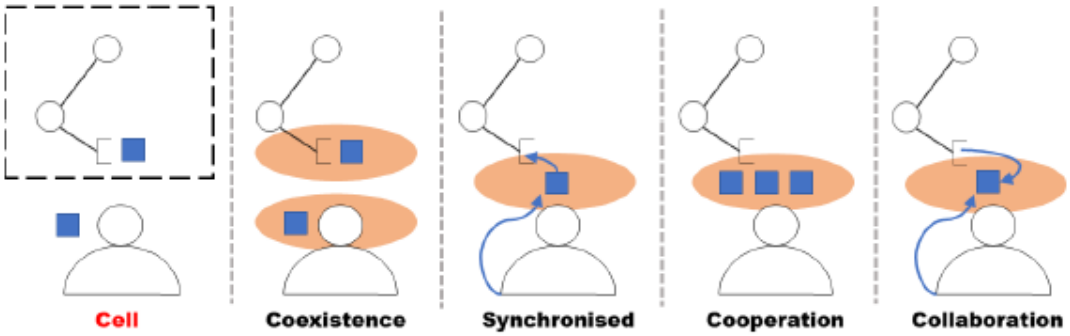


Figure 2.1. *Types of use of a collaborative robot. Taken from [6]. The leftmost image titled "Cell" is the traditional industry way to implement robots that do not collaborate with humans in any tasks.*

to adapt to the environment. The HRC system is a complete entity that is contextually slightly separate from the listed items and is an extension and full realization of collaboration. Out of the items listed above, the most interesting case for industry is collaboration, even though for some tasks the increased speed of spearated agents is preferable. In this thesis, the human and robot do tasks in the same workspace, but the examples with which the framework was trained do not contain examples of human body parts. With the implemented framework, the human and the robot can act in the same workspace and do collaborative tasks such as handover, but the robot might be confused if human body parts are present in the examples. There was also no effort to implement avoidance behaviour, since the used robot was considered to be generally safe even in close proximity to humans and collision situations were generally considered to not be particularly dangerous. The implemented framework does not move the robot at maximum speed.

2.2 Machine learning applications in robotics

Machine learning plays a big role in robotics where the tasks are not predictable in some way. Machine learning is also commonly used with robotics to infer context from the workspace and to extract task-relevant object locations. There are also vision-related concepts in robotics that are not directly tied to machine learning but are useful to understand. One such concept is the idea of eye-in-hand and eye-to-hand perception.

2.2.1 Computer Vision

In collaborative robotics, machine learning and neural networks can be used to further improve safety and to facilitate the unpredictable nature of human actors. Environments where humans act can also change and if the robot is mobile, it usually has to react automatically to these changes to be efficient. The robots might also face completely unknown situations, which makes traditional behaviours based on conditional arguments

prohibitively difficult to implement. Some robotics applications have been developed for image classification frameworks, such as BLIP presented by Li et. al. [7] or the thesis-relevant CLIP discussed by Radford et. al. in [8]. These frameworks mainly describe seen items in images and more advanced models also give the option of drawing bounding boxes around those detected items. Applying spatial transforms and correcting for camera occlusion and distortion allows for 3D positioning of the image items in real space in coordinates that are simple for robots to understand. The process computer vision problem of visually identifying items can also be reversed with multiple different approaches, such as the modern approach of stable diffusion employed by frameworks such as BLIP-Diffusion presented by Li et. al. in [9] or DALL-E2 presented by Radford et. al. in [10].

2.2.2 Natural Language Processing

Human actors also communicate with each other using speech both deliberately (sentences) and instinctively (yelps, gasps), which makes natural language processing an attractive option for coordinating collaboration with robotic actors.

Voice Activity Detection (VAD) is the first step of capturing speech, by using binary classification to detect speech being present in the a sound input signal. Various VAD methods have been proposed and implemented that differ in terms of latency, accuracy, computational cost, and sensitivity. One current and popular method is WebRTC [11], which processes 30ms audio chunks with a small latency of less than one millisecond. Another method, Silero VAD [12], is based on a multi-head attention network and performs slightly better in terms of accuracy likely due to the large language corpora it was trained on.

Automatic Speech Recognition (ASR) has also been a popular research topic in past years and is another step needed when human speech needs to be detected, recognized, and converted to text. Earlier works produced large ASR models due to the amount of data used and the complexity of the models, which increased latency and reliability due to network connectivity. More modern ASR implementations have moved towards offline speech recognition with models such as Kaldi presented by Povey et. al. in [13] and VOSK [14]. These models are typically comparatively small (i.e., <100MB), enabling real-time streaming on mobile devices.

Natural Language Processing provides an understanding of language in text, for which past efforts have utilized Recurrent Neural Networks (RNN) as stated in the paper by Yin et. al. [15]. According to Vaswani et. al. [16], More recent attention mechanisms have utilized a transformer architecture in approaches such as Devlin et. al. with BERT in [17] and Reimers et. al. with SBERT in [18].

2.3 Related Works

This chapter discusses some of the most important relevant works and similar implementations to the implemented framework. The previous work from which this thesis continues is also introduced along with relevant concepts. Some possibly relevant topics (such as robotic manipulation or specifics on neural networks) are not discussed in this thesis due to them being presented in the previous work.

2.3.1 Previous works

In the thesis [1] O. Hassan introduces a framework for transforming natural language instructions into robot instructions that are tied to the real world through observations. This thesis can be considered a continuation and verification of that work. Some changes to the overall framework were necessary but not in a significant way and the underlying architecture was otherwise left untouched except for moving from a Wi-Fi-phone-microphone -solution into using just a USB plugged microphone. This change did not require any rewriting of functionality and did not appear to affect the framework execution. Repeating the implementation details presented in the thesis [1] here is unnecessary, and the information presented in my thesis tries to not repeat anything except when necessary.

Some functionality was added to the framework. The speech recognition tool was edited so that it can understand some numbers and it also received a new command. Speech tool changes are discussed further in Chapter 3.4. The original robot moving tool frequently stopped due to high acceleration values and was automatically configured to send an "all clear" message so that execution of the task could continue. This behaviour was removed to better adhere to the collaborative nature of the robot, and instead a "recover" command was added so that the user could resume operation with the robot without having to restart all the required tools. The acceleration values were also limited, and maximum velocities were implemented so that the robot would not always try to move at max speed. These velocities were also configured so that the robot moved slower when approaching the object to pick and when moving it compared to moving the arm unladen.

A lot of effort was made in order to implement repeating tasks, and the details of this are discussed in Chapter 3.6. The implemented solution is somewhat limited in that it can only repeat known tasks and more complex tasks would still require manual writing of relevant locations. More advanced tasks exhibited by frameworks such as CLIPort were not implemented. Some complex, unimplemented, and industry relevant task examples include aligning an object to some surface element, pushing objects around and sorting objects by type.

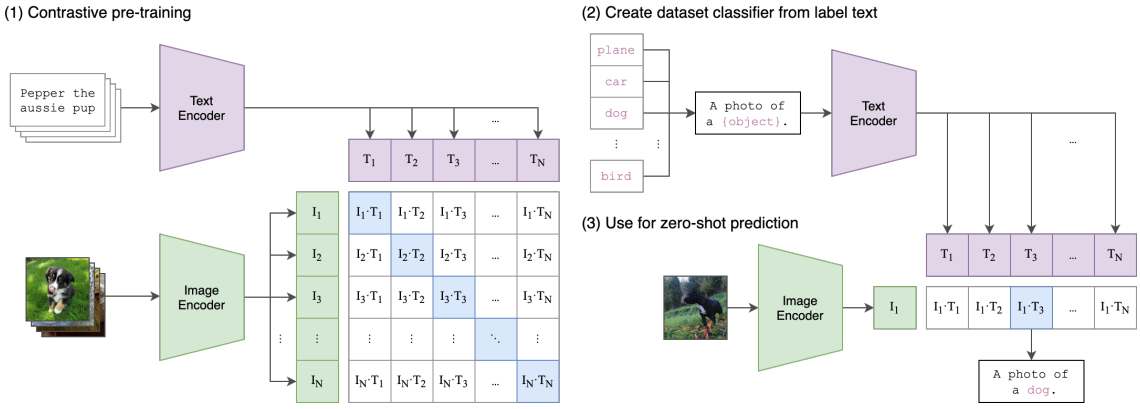


Figure 2.2. *Summary of the CLIP detection approach directly given in the GitHub page [19] and the paper [8]*

2.3.2 CLIP

CLIP is a framework that gives natural language context to content in pictures. According to the CLIP GihHub page: "CLIP (Contrastive Language-Image Pre-Training) is a neural network trained on a variety of (image, text) pairs. It can be instructed in natural language to predict the most relevant text snippet, given an image, without directly optimizing for the task" [19]. Radford et. al. introduce CLIP in the paper [8] and discuss the differences in recent image classification frameworks. The paper also studies the behaviours of classification algorithms at scales in between the least example-rich cases and most example-rich cases, but for this thesis the differences are not relevant and the approach chosen by CLIPort is the only considered option.

The produced CLIP framework utilizes a zero-shot predictor, which in this case refers to the capability of classifying unseen object categories in an image. Unseen objects are objects that haven't been present in examples during training. The images CLIP categorizes as known objects were not found during the literature review, but it can be assumed that the engine parts used as objects in this thesis are unknown due to their relative obscurity. These objects are shown in Chapter 4.1.1. The example amount in manufactured datasets presented in Chapter 4 were largely chosen based on the dataset presented by Hassan O. [1] and CLIPort descriptions on physical test datasets. Neither this thesis or the previous work [1] utilizes CLIP in its default form, but it is a core part of CLIPort [2] (discussed in Chapter 2.3.3) is used extensively.

Figure 2.2 gives an overview on the CLIP architecture with the dual-encoder approach visible. According to the paper [8], normal image classifiers jointly train the feature extraction and linear classifiers while the approach by CLIP has an image encoder and a text encoder be trained jointly. This allows for better scalability.

2.3.3 CLIPort

CLIPort introduced by Shridhar et. al. in the paper [2], available at [20] is an end-to-end imitation-learning agent that can learn a single language-conditioned policy for various tabletop tasks [20]. It is a solution to the problem of a human communicating tasks to robots with natural language and can be trained with relatively few examples. CLIPort has also been demonstrated to function with a real robot, which makes it a very good fit for the framework implemented in [1]. CLIPort can be trained to utilize either multi-task policies or single-task policies. A multi-task policy is capable of executing multiple steps in a sequence to achieve more complicated behaviours where as single-task policies are most applicable in tasks that can be collapsed to single task executions. During the work for this thesis, a multi-task policy option was investigated but it was found to not provide significant benefits over using single-task policies. This investigation is further inspected in Chapter 4.1.3. The CLIPort GitHub page notes that multi-task policies can have better or comparable performance to single-task policies [20].

2.3.4 Other works related to machine learning in robotics

This chapter lists other cobot-related works and natural language processing related works in robotics which were examined during research, but do not tie directly to CLIPort. Related works with direct ties to CLIPort are discussed in subsections of Chapter 2.3.3

VLMbench

Zheng et. al. introduce VLMBench in [21] as a robotics benchmark for *Vision-and-Language Manipulation* (VLM) in simulated 3D environments. To investigate the difficulty of the benchmark, they utilize several partially modal solutions and a special CLIPort modification, 6D-CLIPort. 6D-CLIPort differs from the unmodified implementation mainly in that instead of utilizing just the pose generation for the 3D location, the output pose is generated as a 6 Degree-of-Freedom (DOF) pose that also contains rotation data for the tool. It an also output an arbitrary number of poses unlike the unmodified CLIPort, which only outputs two positions representing the pick and place actions. These two factors combined allow 6D-CLIPort to have greater control over the trajectory the robot takes in task execution. This modification of CLIPort is stated to achieve an execution accuracy of 28.28% in the pick & place action, which can be considered to be comparable to the pick & place actions executed in this thesis based on the task explanation noted in the VLM-Bench paper appendix A.1 [21]. The performance is likely lower due to the benchmark also utilizing unseen containers, unlike the tests in this thesis.

LEMMA

Gong et. al. introduce the LEMMA benchmark in [22]. The article considers 8 types of tasks of varying complexity and makes comparisons to different robotics benchmarking solutions such as VLMBench [21]. This work is particularly interesting since it uses CLIPort as the low-level planning solution in both multi-robot and single-robot tasks, although they use a multimodal agent in contrast to the agents trained during this thesis. The focus of the paper considering CLIPort is to test the language processing portion which they achieve by using both high-level instructions and human instructions, which differ by the latter being more complicated. During this thesis work, human instructions are accepted only in some specific forms and are collapsed with sentence similarity matching to known high-level instructions also in contrast to the testing approach pursued in the LEMMA paper. As in this thesis, the LEMMA paper utilizes CLIPort to predict the pick and place locations. They also introduce a modular version called M-CLIPort which won't be discussed further here. With high-level instructions, the single-step model achieves an accuracy of 87±0.67% in the pass task, which can be considered a comparable task to two of the pick & place tasks introduced later in this thesis in Chapter 4.

CALVIN

Mees et. al. introduce CALVIN in [23] as a benchmark for language-conditioned long-horizon robot manipulation tasks. Long-horizon tasks in the context of robotics means tasks, which require the successful execution of various sub-tasks for the greater goal task to be considered successful or possible. The long-horizon tasks [23] considers has these sub-tasks be completable in an arbitrary order, but this might not be possible in all long-horizon tasks. Compared to the tasks executed in this thesis, the tasks evaluated by CALVIN require much more contextual understanding of the environment and task. The tasks are also comparatively more difficult to complete due to requiring multiple steps (up to five).

VIOLA

Yifeng et. al. present VIOLA in [24] as an object-centric imitation learning approach to learning closed-loop visuo-motor policies for robot manipulation. *Imitation Learning* (IL) is a long-standing robotics paradigm for acquiring manipulation policies, which in turn are used for task execution. VIOLA generates a transformer-based task policy from object-centric representations, which in turn are extracted from workspace image features. This allows the framework to be highly adaptable to tasks in the robotics context. The policies are tested with the MAGICAL benchmark [25] where VIOLA outperforms similar state-of-the-art competitors. VIOLA is also tested on real-world tasks where it also outperforms the discussed competitors.

LATTE

Bucker et. al. introduce LATTE in [26]. It is a language-based trajectory transformer model and is very comparable to the CLIPort implementation due to also utilizing CLIP in extracting image features and BERT in inferring natural language. Unlike with CLIPort path generation, the goal of LATTE is to modify an existing robot trajectory to obey an user's semantic commands and avoid trajectory obstacles. The implemented solution is not limited to just robot arms and should theoretically work with any robot trajectory. The paper proposes such trajectories executed with helicopter drone mobile robots. LATTE is tested in the real world with a Franka PANDA [27] robot arm of the same model as presented in a later chapter in this thesis with user validations. The LATTE paper notes that during real-world testing, 48% of the used words were not present during training making the implemented solution highly flexible since the model failed only in 24% of the cases.

PALM-E

Driess et. al. introduce a single embodied multimodal machine learning model PaLM-E [28] for solving sequential robotic manipulation planning and other reasoning tasks. One main goal of PaLM-E is to represent images and text as multimodal sentences similarly to VIMA and to show that this approach can achieve state-of-the-art performance in addition to solving problem modern zero-shot solutions have with embodied reasoning problems. Since the approach is multimodal, the model is capable of solving long-horizon tasks and does not have to rely on task-specific fine-tuning to achieve comparable results to other modern language-based policy generators. In visual-question-answering (VQA) tasks, PaLM-E is shown to outperform other modern models such as ViT-4B [29] and OSRT [30] in some cases. In the robotics domain, PaLM-E is tested in three robot environments and tested on Task and Motion Planning (TAMP). In the environment PaLM-E plans and executes pick-and-stack tasks, which are a slightly more advanced form of the pick-and-place tasks the implemented framework is examined on in this thesis. Certain PaLM-E configurations are shown to achieve a task planning success rate of 94.9% in the TAMP environment.

RoboCAT

Aguinaldo et. al. introduce a robot interoperability framework called RoboCat in [31]. The stated goal of RoboCat is to decrease the expertise required for developing robot task policies by abstracting the policies behind conceptual templates which can be transformed into production plans. Other benefits from the descriptive approach include the reduced need for writing bespoke controllers for different robots, as the framework could translate high-level instructions into individual process plans customized for each manufacturer.

The aim of RoboCat is to increase productivity by abstracting primitive robot behaviours into goal-oriented programs. Modern industrial programming methods already include graphical programming languages to increase productivity, such as with the graphical user interface of Franka Emika [27].

VIMA

In the paper [32] Jiang et. al. introduce a transformer-based robot agent, **Vi**suo**M**otor**A**ttention agent (VIMA), and a simulation benchmark which can represent a wide spectrum of robot manipulation tasks with multimodal prompts called VIMA-Bench. The goal of VIMA is to produce an agent which can execute any task which can be expressed by multimodal prompts. The produced model is scalable and sample-efficient with the paper noting that using certain language generalization levels, the model can match other solutions such as Mask R-CNN while utilizing just 1% of the training data. The introduced benchmarking tool is also extensible with both extra objects and textures to increase the amount of possible multimodal prompts and to generate a larger number of tasks. Some tasks, such as Task 02 in [32] appendix B is comparable to the pick and place tasks examined later in this thesis.

DALL-E-bot

Kapelyukh et. al. introduce DALL-E-Bot in [33] as a zero-shot, open-set and autonomous scene rearranging engine with the goal of mimicking human behaviour in everyday actions such as in setting the table with tableware. They succinctly state "... our system gives web-scale diffusion models an embodiment to actualize the scenes that they imagine.". The scenes are framed with a static workspace with all the desired objects present. The rearranged scenes are rated with real human reviews showing that compared to random non-colliding or geometric schemas, the DALL-E-Bot results are on average the most preferred. DALL-E-Bot also includes a sample-and-filter approach to cull any unrealistic generated images which are still common with diffusion models, such as DALL-E 2 [10].

3. EXPERIMENT SETUP AND CHANGES TO PREVIOUS WORK

This chapter discusses the previous solution and the changes made to it. For a better understanding of the system, one can familiarize themselves with the thesis by Hassan O.[1].

3.1 Framework overview

No changes were made to the original Python nlihrc framework apart from moving from a phone as a microphone to utilizing a USB connected microphone due to network limitations. This does not change the framework function or flow in a relevant way, and the framework can be told to use the old method with a configuration parameter.

3.2 Hardware setup

This chapter lists all the thesis-relevant hardware specifications for the components used in the framework for real world task execution.

3.2.1 Desktop computers

There are 3 PCs which were utilized on the regular during this thesis, two for machine learning and heavy GPU loads and one real-time capable PC. CLIPort predictions were done with a Ubuntu 20.04 machine equipped with an Nvidia GeForce RTX 3080 [34] GPU and an AMD Ryzen 7 5845 [35] 8-core CPU (*PC 1*). Due to memory limitations, the models were trained on a different computer equipped with an Nvidia GeForce RTX 4090 [36] GPU and an Intel Core i9-13900k [37] 24-core CPU (*PC 2*).

A two-computer approach was used for controlling the robot due to real-time processing requirements present in robotic commands and the general difficulty of processing machine learning with time gates necessitated by real-time instructions. On the real-time PC, the GPU was not utilized in command processing and the CPU was an 8-core Intel Core i7-4770 [38] (*PC 3*). The RTX 3080-equipped PC mentioned earlier in this chapter (*PC 1*) was also configured so that it could be launched in a real-time mode in order to facilitate

dataset capturing. All presented and produced dataset samples were captured in that configuration, apart from the data from the pre-existing dataset which was extended.

3.2.2 Microphone

The previous project used a smartphone application-based speech capture solution but transferring the WiFi-dongle based solution caused problems during the new computer installation. This is why during this thesis work the voice was captured with a USB-connected microphone instead. The microphone used was the microphone on a Jabra EVOLVE 20 MS Stereo headsets [39]. Using the microphone changes nothing in the workflow for Python nlihrc, except that no WiFi communication is necessary.

3.2.3 Intel D435 Camera & recording

The camera used for capturing both the RGB and depth information was an Intel D435 camera [40]. Experiments were also done with an Intel D435i camera [41], but this was observed to not improve detection and caused calibration issues which is why its integration was abandoned and won't be discussed further in this thesis. The camera was used with the same calibration parameters as discussed in the thesis [1]. The camera was not used at its maximum resolution of 1920x1080 for RGB and 1280x270 for depth, but instead in a 640x480 mode which synced colour and depth so that the image elements and depth elements have a correlation in every pixel. This was done also in the thesis [1] for better depth perception accuracy and ease of implementation.

3.2.4 Robot

The robot used in this thesis is a Franka Emika Research 3 [42] robot arm equipped with a fully integrated 2-finger parallel gripper as an end effector. The Franka Emika robot arm is a cobot, which makes it comparatively safe to work around. It has force/torque sensors in all its seven links of the arm body, which are used to detect collisions. The signals the robot monitors are joint position, velocity, torque cartesian position and force. The arm can be fitted with either a vacuum gripper or the 2-finger gripper, which was used in this thesis. The robot itself is controlled with a real-time kernel desktop PC. The goal locations are not calculated on this real-time computer, but are received from a prediction PC, as described in Chapter 3.2.1. All communications happen via a local Ethernet connection both to the robot and between computers.

3.3 Vision-Language Model

The framework utilizes CLIPort [2] to execute tasks specified with spoken language on a perceived image. CLIPort is examined more thoroughly in Chapter 2.3.3. Natural spoken language is turned to text via a previuously implemented framework tool, which utilizes SBERT [18] in transcribing the spoken sentences. Sentence similarity is used to further increase the precision of the detection model for relevant sentences. The CLIPort-nlihrc implementation however uses the similarity on a word-by-word basis, and thus has a slightly worse performance collapsing whole sentences, but in turn also functions for single words.

3.4 Text Processing

Some task-specific improvements were made to the text processing capability of the framework as well as some new tasks were introduced, such as handover and repeat. One conscious limitation with the implementation of the repeat task is that only the first number present in each sentence is considered. Due to limitations with implementation time and low prioritization, this number is presented in a command separate from the command that has the framework repeat some task. Further discussion on the repeat task is found in Chapter 3.6. In addition to the added tasks, the framework has received the capability to understand certain aliases for numbers such as "once", "twice", "thrice" etc. New objects and related sentences were implemented to the framework to support the added industrial tasks. The frameworks vocabulary was also extended to include these words when doing sentence similarity.

3.5 Speech recognition

One concern raised during the implementation of the previous work [1] was the belief that the speech recognition portion of the framework would not recognize other users' speech well enough for smooth operation. This was not perceived to be a significant problem with the framework due to the implemented functions that improve speech recognition, such as sentence similarity matching. In addition to sentence similarity, the framework attempts to collapse heard wave forms to a pre-defined vocabulary of words, which limits the possible sentences the system understands while increasing prediction accuracy for relevant sentences. The system was only tested with male speakers and is yet untested with female voices. The framework has also not been tested with children, who share a higher pitch in vocal tones with female speakers. Doing robust tests on the framework would also have required voice profiling and along with the added scheduling complexity, this recognition verification was not considered to be a significantly relevant portion of this thesis even when it was one of the original research goals. The verification leans

into verifying the function of the backend BERT models referenced by Devlin et. al. in the paper [17] and whose performance is discussed by Turc et. al. in the paper [43]. Anecdotally, the speech recognition was not observed to be detrimentally deficient for robot operation except with some particular words, such as "panda" which was used in one of the voice interface commands.

3.6 Multi-step workflow

A possibility to execute multiple successive tasks was added into the framework. Originally this functionality was intended to be achieved using multiple confidence peaks from a single prediction, but this approach was not used due to unpredictable confidences. Examples of this unpredictability are presented and further discussed in Chapter 5. Finding a good criterion was deemed too difficult and multi-step functionality was instead achieved by repeating single tasks a predetermined number of times, which also has some benefits with the cost of increased execution times.

The framework can repeat any command it knows by the number of times the user has specified before issuing the command to repeat. This allows for picking up multiple objects in a consecutive fashion but forces the user to identify how many objects are in the workspace. The machine will also take longer to execute this type of consecutive flow compared to a hypothetical multi-peak solution, since a new prediction is made after every pick & place task. A multi-peak solution would theoretically have needed only a single prediction. The framework was tested with repeat tasks that included no more than ten repeat cycles, even though there is theoretical support for thousands of repetitions.

One benefit of the implemented repeating prediction and execution has, is the increased resistance to changes in the workspace. Since a new prediction is done after every pick and place action from the same pose, the framework can correct mistakes it might have made in just a single multi-peak prediction. In testing done on the real robot it was perceived that in very difficult and cluttered workspaces, the robot task execution could cause changes in the locations of objects. This is due to realities such as object collisions and object movement, which is in no way predicted by the framework even though these are simulated by the ravens simulator used for training CLIPort originally. CLIPort also makes its predictions based on a single image and reacting to changes in the workspace would require making auto-corrective predictions during task execution from a video-like stream, which is not supported. Examples of workspace changes perceived during testing include the conditions that cause errors presented in Chapter 5.5.

One particular problem is raised in Chapter 5.5, namely "object collides with target container". This error happened more often than other discussed errors during the testing due to the path step design. The robot was also perceived to sometimes collide with its own frame with the carried object, usually causing the task to stop. Similar task-stalling

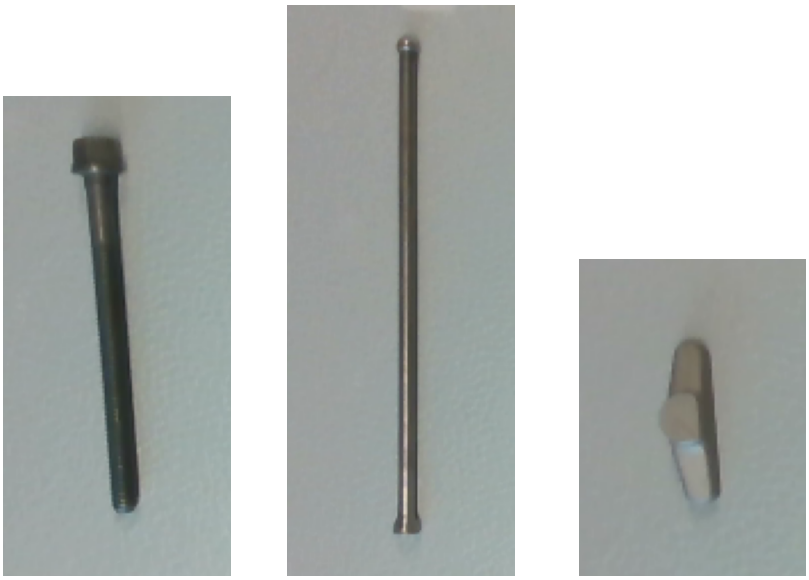
errors are also discussed in Chapter 5.5. In tests done with a real robot it was observed that most of the changes done to the workspace during task execution did not inhibit a successful completion of a new consecutive task, if the object was still perceivable from the home position the robot used for making position predictions.

4. DATASET FORMAT AND PURPOSES

The chosen models and their dataset makeups are discussed in this chapter. The used objects are also discussed along with presenting the template images later used in Chapter 5 to discuss model successes and failures. Test presumptions are also given for what results were expected prior to carrying out accuracy tests on the real robot or simulations. One important aspect to understand about the data collection process is, that all the data was collected with a previously implemented demonstration tool. The training process also did not use image augmentation available in CLIPort. Using the demonstration tool guarantees that the location examples present in the dataset are recorded from positions the robot went to, so every example and saved task is a user-verified valid task from that location. Situations where the task was blocked or otherwise interrupted during the task execution resulted in states in which the robot position could not be correctly saved and there was no functionality implemented for editing the currently executed tasks. Task-stalling errors are discussed in Chapter 5.5 along with change-causing errors. The latter of these is not considered in the dataset as they were not execution-critical and can't be properly predicted by the single-task approach pursued in the framework.

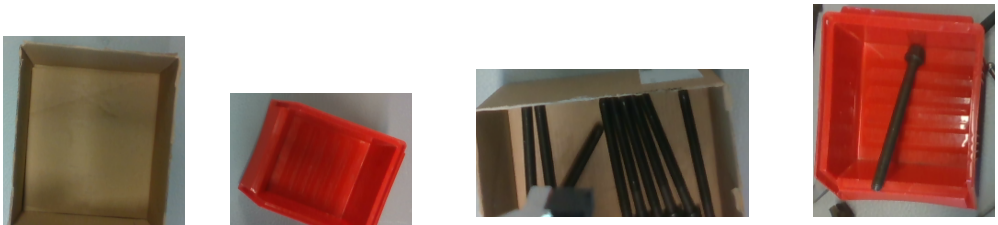
4.1 The presentation of the format and purpose of the datasets

This chapter goes into further detail on the three trained datasets as well as presents the used objects. All dataset sample examinations later in Chapter 5 follow the four views presented and discussed in detail in Chapter 4.2.1. The images are later presented in a four-image grid in order to save space configured so that the top-left corner image is equivalent to the RGB view as presented in Figure 4.4 along with the success criteria boxes and actual pick & place positions as demonstrated in Figure 4.8. Notably the colour schema for pick and place used in Chapter 5 examples follow Figure 4.8 colours. The top-right view is the shape-conformed height-map data equivalent to the data presented in Figure 4.7. The two bottom views correspond to the pick and place confidence maps so that the bottom-left view is equivalent to the pick confidence produced by CLIPort as exhibited by Figure 4.5. The bottom-right view is similarly equivalent to the place confidence exhibited by Figure 4.6.



(a) *Bolt* ***(b)*** *Push rod* ***(c)*** *Rocker arm*

Figure 4.1. *General examples of the main industrial parts. The bolt uses the alias "long screw" in some models.*



(a) *Brown box* ***(b)*** *Red box* ***(c)*** *Brown box with several bolts inside* ***(d)*** *Red box with a bolt inside*

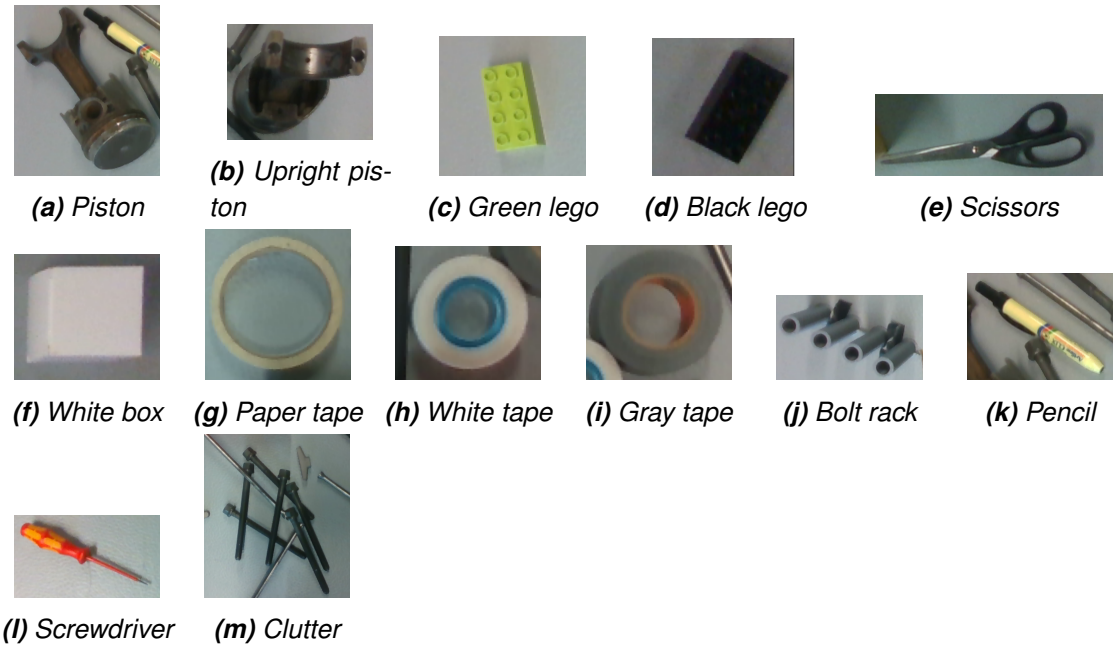
Figure 4.2. *General examples of the goal boxes (a,b) and cluttered boxes (c,d).*

4.1.1 All objects present in the datasets

This chapter presents every object present in the training data with image examples regardless of whether they were classified in examples or not.

Figure 4.1 presents the three main industrial objects that were trained for the system. They were selected mainly due to available duplicates and especially due to the similarity in shape and texture when viewed from above. In a depth map such as shown in Figure 4.7, the bolt and push rod especially seem similar. The objects presented in Figure 4.1 are not to scale and are cut from camera captures from the home position, so the pixel resolution is the same as seen by the camera.

Figure 4.2 presents the two goal objects in Figure 4.2a and Figure 4.2b and two augmented variations in Figure 4.2c and Figure 4.2d. In the presented models, the red box goals were only present for the final model, but the red box featured heavily in other trained models for the framework. To adequately compare the final model to the older



(a) *Piston* ***(b)*** *Upright piston* ***(c)*** *Green lego* ***(d)*** *Black lego* ***(e)*** *Scissors*

(f) *White box* ***(g)*** *Paper tape* ***(h)*** *White tape* ***(i)*** *Gray tape* ***(j)*** *Bolt rack* ***(k)*** *Pencil*

(l) *Screwdriver* ***(m)*** *Clutter*

Figure 4.3. *General examples of the obstructions present in the validation dataset. Note that green lego was a known goal object for model 3.*

models, the brown box was kept as a goal object. The presented cluttered examples in Figure 4.2c and Figure 4.2d are taken from the final model training dataset.

Figure 4.3 presents every clutter object present in every dataset except for distracting objects. Distracting objects are objects in the workspace that are duplicates of the pick task goal object, that in some ways are unfavourable to the actual goal object. In some datasets, these objects were inside the place task goal object and in some others, they were too clustered for the claw to accurately pick up. An example of clutter created with goal objects, namely bolts and pushrods, can be seen in Figure 4.3m. Most of the clutter objects presented in Figure 4.3, namely Figure 4.3c, Figure 4.3d, Figure 4.3e, Figure 4.3f, Figure 4.3g, and Figure 4.3i were present as clutter only in the model 3 dataset. Some objects, such as the black lego shown in Figure 4.3d, were omitted from later datasets since they were considered to not normally appear in an industrial context.

4.1.2 Final model dataset

The final model (model 1) is the best performing model and final one trained during the thesis. Its main purpose is to demonstrate sufficiently that the extension on the previously made CLIPort framework can discern between industrial objects of similar shape and texture. In addition to this, it was expected that the model could function sufficiently with the selected tasks (pick & place) even when significant debris is either fully or partially present in the workspace. The model is also expected to function well if multiple valid choices or duplicates of the target object are present. One interesting facet of inspection

Task name	Amount
put bolt in brown box	16
put bolt in red box	22
put pushrod in brown box	11
put pushrod in red box	25
put rocker arm in brown box	11
put rocker arm in red box	25
Total tasks	**110**

Table 4.1. *Data-set tasks for the final model.*

was also how well the system handles objects on unseen backgrounds, as all data-set examples were collected with the same background and at the same height (comparative to the camera). The model performance is examined more thoroughly with examples in Chapter 5.2.

The data-set for this model consists of a roughly equal number of demonstrations for the bolt, the push-rod, and the rocker arm. The general makeup of this data-set is presented in Table 4.1 and its more specific task-object makeup is presented in Table 4.7 for clean tasks and Table 4.8 for cluttered tasks. Clean tasks in the context of the datasets referst to tasks which have only one object, the goal object, present. Across all datasets, all industrial objects have examples with both only one object & multiple duplicate objects, but emphasis was placed on recognizing single objects since CLIP, on which CLIPort and by extension this framework is built on, is a general image classifier and should recognize objects from cluttered images if it understands them. The dataset also contains examples with all task items present or a choice between two of the chosen objects (bolt & pushrod, for example). In addition to these, the dataset also has some demonstrations with various clutter items that are not labelled or relevant for the task. There are also tasks present with two different target boxes. From the overall datasets, this set was the fifth assembled during this thesis.

4.1.3 Multistep model dataset

One aspect that was discussed in the previous paper but was not implemented was the possibility of using the multi-step tasks for which support exists ready-built in the CLIPort framework. Initial steps were taken to bring this into reality, but the work was later abandoned due to timeline difficulties and performance-related irrelevance as discussed in Chapter 3.6 and later examined in Chapter 5. The training done with the initial multi-step model did not yield observable benefits to the pick & place task in the examined demonstrations. Instead, multi-step tasks were achieved in the framework by repeating

Task name	Amount
put all long screws in brown box	30
put all push rods in brown box	17
Total tasks	47

Table 4.2. *Data-set tasks for the multistep model.*

single pick & place tasks with the operator specifying the number of repetitions prior to execution, as discussed in Chapter 3.6. This approach was observed to be successful for five consecutive repetitions, which was deemed sufficient for the purposes of this thesis. One sought difference with this model was a difference in the confidence maps. It was presumed that the multi-model would detect multiple valid confidence peaks when duplicate multiple variations of the goal object are present in the task. This behaviour was not observed to increase sufficiently compared to the single model which was discussed in Chapter 4.1.2, and some supporting evidence is presented also in Chapter 5.

The second model dataset only consists of examples between multiples of both the bolt and the push-rod. There are also demonstrations of just a single object in a task. There are also examples where both objects are both present in various relative quantities (all the way to singles). The generic makeup of the dataset tasks can be seen in Table 4.2. Due to this model being mainly used for development insights and testing, its data was captured with slight negligence, which results in the datasets asymmetric dimension. One task with multiple pushrods was not captured, but this was deemed irrelevant since the pushrods were added as a proof-of-concept and the focus of the model was on multiple bolt detection. One additional thing to note about this database is that the recorded examples use the old moniker, "long screw", for the bolt. This trait is shared with the dataset for the extension model, which was the first dataset assembled in this thesis with this being the third.

4.1.4 Extension model dataset

The first dataset and model created for the industrial object manipulation during this thesis was done as a direct extension to the model trained in the previous thesis done on this subject, [1]. This model was partially used as a baseline and tested due to the possibility of extending existing datasets. One issue overlooked in the data-set collection process was that the camera used in the original dataset (prior to the newly recorded extensions) was captured with a camera that was later deemed missing. In addition to this, the used robot was in a different location with different lighting conditions & background elements. The impact of these could have been inspected had data been collected for both environments, but that was deemed to go beyond the thesis subject.

Task name	Amount
put white box in brown box	30
put white tape in brown box	30
put red screwdriver in brown box	30
put black lego in brown box	15
put green lego in brown box	15
put long screw in brown box	13
Total tasks	**133**

Table 4.3. *Data-set tasks for the extension model.*

Task name	Amount
put bolt in brown box	19
put push rod brown box	19
put rocker arm in brown box	16
put bolt in red box	12
put push rod in red box	12
put rocker arm in red box	12
put green lego in brown box	4
Total tasks	**94**

Table 4.4. *Tasks in the validation dataset.*

In addition to what was present in the original dataset, this data-set was extended with 13 examples of the bolt (with the alias “long screw”). Three of these examples were made into validation points. The general makeup of the model dataset tasks can be seen in Figure 4.3.

4.1.5 Validation dataset

All of the models presented in this thesis were validated using the same dataset with a focus on industrial tasks. This causes the extension to perform dis-proportionally badly, since only the bolt was presented to it and the framework has no examples for red box goal tasks. This is taken into account in the weighted average success rates in Chapter 5.4. The multistep model and the extension model datasets also used older aliases for the bolt (long screw), which was considered in the implemented model comparison tool. The validation dataset was collected separately after all models had been trained and the models intended to be presented in the thesis were selected so that it could represent all of the models at least in some respects.

Table 4.4 presents the task split in the validation dataset. Since the 2nd and 3rd model

only understood the brown box as a task model, an emphasis was put on tasks with that place task goal object. The brown box tasks also include tasks with many objects' multiples, whereas the red box task were either single object tasks or tasks with a single distracting object.

4.2 Model successes criteria

This chapter discusses the model criteria and verification process for the created models, which are presented later in Chapter 5. Due to the long time a move takes to execute on a real robot, the models are verified on simulated pick & place actions. The handover task was not separately rated, since its success rate is the same as the pick success for a model. Note that all figures in this chapter are taken using the final model, model 1, and the data displayed is from a different tool used to examine data during real robot task execution. The one exception in this chapter is Figure 4.8, which is a special capture demonstrating the success policy in a greater resolution than similar demonstrations in Chapter 5. The figures in Chapter 5 are taken from a simulated environment, where the real robot is unnecessary but the points are predicted by the same framework.

4.2.1 General example of a pick & place task on a bolt

Figure 4.4 displays both goal objects, the brown box (top left) and the red box (bottom left). It also displays two of the industrial objects, the bolt (middle object close to a green dot) and the pushrod (far right). Note that in the green dot signifying the pick position for the bolt is clearly banking left from the actual bolt, but this is only due to a quirk in the position visualization software. No significant leftward positional error was observed in the actual task execution process and the correctly placed example points are similarly offset as can exhibited by later examples such as in Figure 4.8. A similar error is present also in the blue dot signifying the place position even though that spot is still inside the goal. The same observation stated earlier about the correct example point also holds for the place point, the example is similarly offset compared to the predicted position. Note that the object is not grasped in the geometric middle due to a conscious choice to pick the bolt at the rough centre of gravity for the object in the training examples. This was observed to increase grip the robot had on certain objects, but with the bolt there is a small risk of getting an error lengthwise along the bolt in such a way that the bolt is grasped instead at the rim visible just below the tightening surfaces in Figure 4.4. This can cause the grip to be point-like instead of giving full surface contact to the gripper fingers, which in turn allows the bolt to turn after it is grasped. This can cause collisions with objects and especially the goal objects.

Figure 4.5 gives us the confidence map from which CLIPort makes its pick pose predic-

Figure 4.4. *RGB image view for “put bolt in brown box” with pick position (green dot) and place position (blue dot) from the left lens of the D435 camera.*

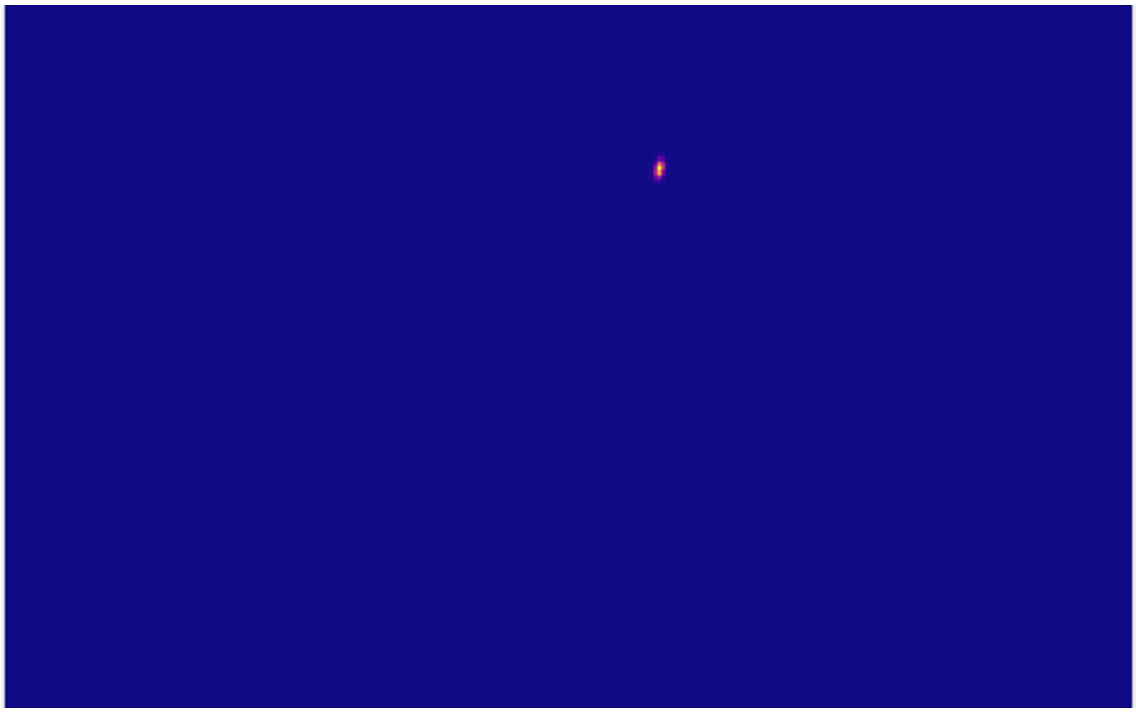


Figure 4.5. *CLIPort confidence for the pick goal of the same task present in Figure 4.4*

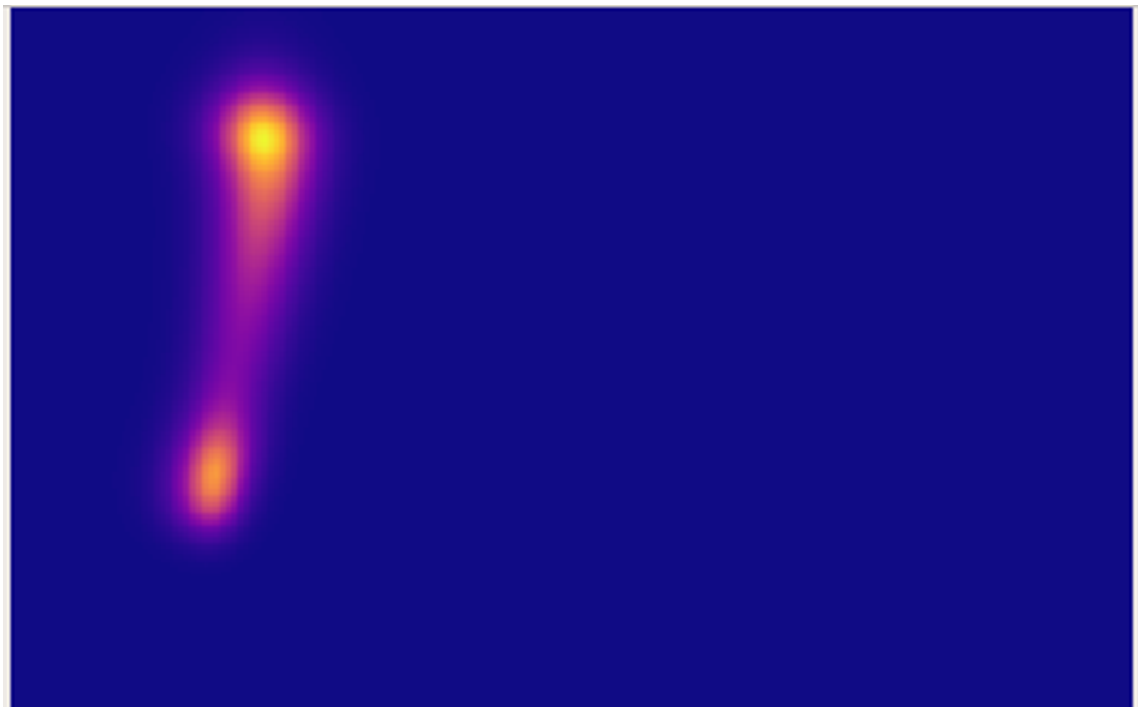


Figure 4.6. *CLIPort confidence for the place goal of the same task present in Figures 4.4 and 4.5*

tion. The location is chosen based on the highest peak value, denoted by a brighter color. Note that in the Figure, both the pixel count and aspect ratio are different compared to the RGB equivalent picture given in Figure 4.4 that was examined earlier. This might be due to the mapping done to correct for occlusion and geometry. Prior to execution it was expected that the model would exhibit significant confusion between the bolt and the pushrod, but this was not perceived in a significant capacity. In a typical case where the framework mis-classified an image where both a bolt and a pushrod were present the pick task goal generally was for the pushrod.

Figure 4.6 gives us the confidence map from which CLIPort makes its place pose prediction. Note that the action confidence the figure gives two significant peaks due to similarity in the box objects' shape. The framework was generally not observed to have confusion with these objects if they were placed sufficiently far away from each other, such as on opposing corners or sides of the perceived workspace. This confusion is likely at least partially synthetic due to a large portion of the examples which exhibit both goal boxes choosing to locate them either very close to each other or directly side-by-side. This fact combined with an overall low epoch count comparative to the CLIPort suggestions or other comparable frameworks suggests that this behaviour might be solved with a mix of longer training and a larger more varied dataset. One additional noteworthy observation on this image is that the chosen plasma colour-map exaggerates features in the data. With a different colour-map, the purple bridge between the two yellow peaks is not necessarily as apparent.

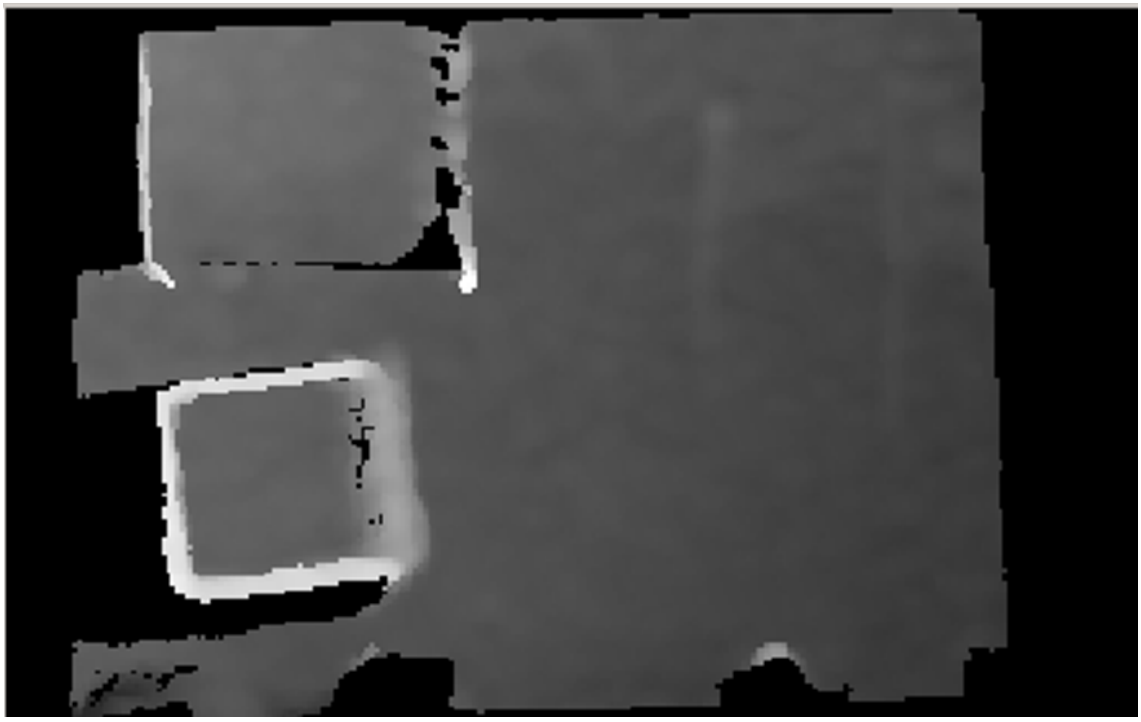


Figure 4.7. *Geometry-corrected height-map for the same task present in Figures 4.4, 4.5 and 4.6*

Figure 4.7 gives us a visual on the height-map image data cast on a two-dimensional image plane. Note that the black sections in the image especially around where the two boxes can be observed in Figure 4.4 are areas that the camera occlusion hides. They should not be mistaken for shadows, even though their function and placement are directly tied to being blocked by task space geometry as light would. The framework was not tested with objects that are transparent in both visible and infrared light, which could cause confusion in geometry detection.

4.2.2 Criteria for pick success verification

Fig 4.8 gives us an example of the RGB based success criterion that is later used to evaluate the success of the model in all pick & place tasks. Figure 4.8 is presented because it presents both the idea behind the policy and an example case where the policy fails. This failure was observed often and thus the framework success was also evaluated by hand. The idea behind using a box around the predicted value was that often the objects were perceived to be picked up even when the predicted position was far from the actual recorded position, which made using strict direction-based criteria unreliable. The box is shaped so that the pick position is accepted if the actual pick position is horizontally (laterally in Figure 4.8) closer than half of the maximal possible width of the claw. This way, the robot could have picked up the object up at either extreme. The height (longitudinally in Figure 4.8) of the box is longer due to the shape of the chosen industrial objects. The

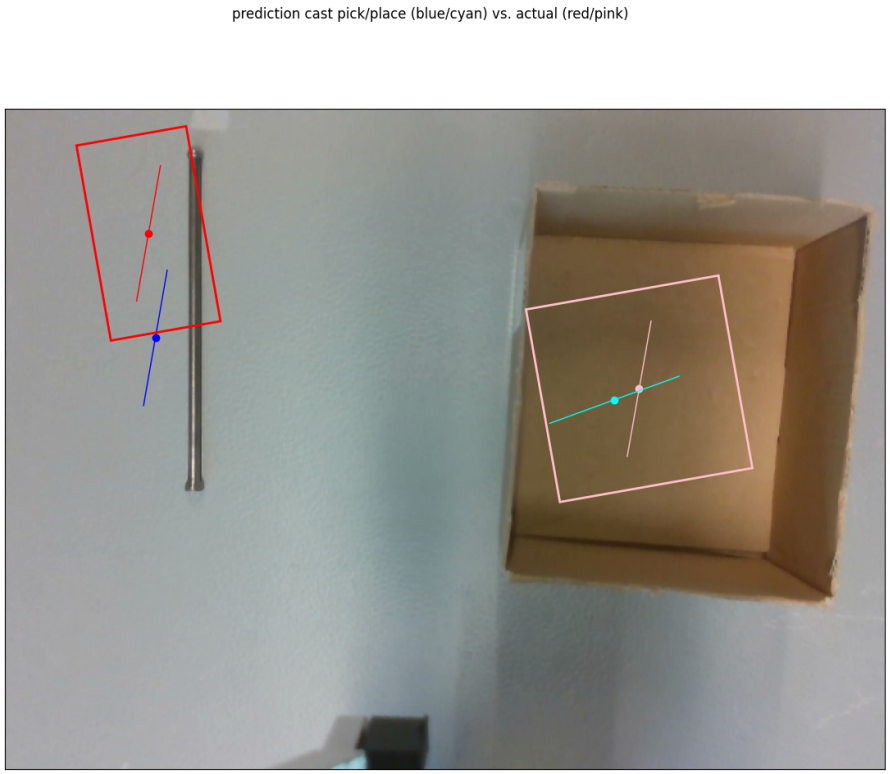


Figure 4.8. *RGB image view for "put push rod in brown box" with the predicted pick position (red dot), predicted pick angle (red line), pick success criterion (red box), predicted place position (pink dot), predicted place angle (pink line), place success criterion (pink box), actual pick position (blue dot), actual pick rotation (blue line), actual place position (cyan dot) and actual place rotation (cyan line)*

prediction criterion does not consider the angle of the picked up object in a relevant way since the final trained model did not seem to ever predict a different angle from the one presented in Figure 4.8. Due to these policy Future improvements should look into either separating the rotation module from the location predictor or fixing it so that it does predict other angles as well. Future improvements of this work should also consider having the criterion box shape conform to the object being picked, since Figure 4.8 presents a valid pick position that would be correct if a taller criterion box was chosen. The current shape is based on the dimensions of the bolt and the width of the used claw. Another conceptual improvement would be to instead predict the criterion box around the known and correct place position since this position also contains the correct rotation data. Currently the predicted position could hit a falsely good position based on the criterion but be in such an angle that the object would not be picked up. Such a situation could happen, for example, with an object that is perpendicular to the predicted angle.

task 1	'put bolt in red box'
task 2	'put bolt in brown box
task 3	'put push rod in red box'
task 4	'put push rod in brown box'
task 5	'put rocker arm in red box'
task 6	'put rocker arm in brown box'
task 7	'put green lego in brown box'
task 8	'put all long screws in brown box'
task 9	'put all push rods in brown box'
model 1	The final model
model 2	The multistep model
model 3	The extension model
tot. avg.	average success of all tasks
wgt. avg.	weighted average success of tasks

Table 4.5. *List of aliases used in tables*

4.3 Dataset parameters

This section lists the parameters and aliases used in the dataset. Note that some tasks changed names between models and tasks are presented only with the newer name even though the datasets contain different names. Table 4.5 gives the complete breakdown of abbreviations used in other tables in this chapter and Chapter 5 such as Table 4.1, Table 4.2, Table 4.3, Table 5.1, Table 5.2, and Table 5.3.

Table 4.6 presents a breakdown of the differences between the presented datasets with all relevant parameters. The differences between the training times between the models is mainly due to the criterion for stopping training. In general the lab computer (PC1) or the training computer (PC2) presented in Chapter 3.2.1 were left to process the dataset for the duration of a single weekend. The training was considered complete if the model was perceived to successfully execute tasks on the real robot and when this was not the case, the training was extended for an additional day. Note that the object breakdowns under "known objects" represent the maximum duplicates of objects in any task, there was no task in the final model dataset that contained 9 bolts, 4 pushrods and 3 rocker arms. Note that multitasks have multiple tasks per example which is why "Total" in Table 4.6 is higher for the Multistep model compared to the Final model.

Table 4.7 and Table 4.8 present the exact makeup and split of the tasks on which the first model was trained and validated on. Cluttered tasks in the table mean tasks that had some other objects than the goal object, while clean tasks are tasks with only the goal object present. Tasks with only duplicates of the goal object are considered clean in the

	Demonstrations			Demonstrated tasks		Pick-and-place targets			
	Training	Validation	Total	Single-object tasks	Multi-object tasks	known objects	unkown objects	Goals	Training time [h]
The Final model dataset	88	22	110	60	50	9 bolts 4 pushrods 3 rocker arms	6	brown box red box	89
The Multitask model dataset	39	8	132	0	47	3 bolts 2 pushrods	0	brown box	26
The Extension model dataset	119	14	133	133	0	bolt white box green lego black lego screwdriver paper tape	1	brown box	107.5

Table 4.6. *Data-set Parameters for the three presented models.*

Task type	**Amount**
bolt to red box	10
bolt to brown box	10
pushrod to red bx	10
pushrod to brown box	10
rocker arm to red box	10
rocker arm to brown box	10

Table 4.7. *Final model clean task breakup*

tables. The training validation dataset samples were taken from the different task types roughly equally, but there were more bolt tasks in the end than the other items, which might have contributed to the slight bolt bias present in the Final model discussed later in Chapter 5.2. Unknown objects represent the number of objects that were not labelled in the tasks, but were present as clutter. Note that all success rates were measured using the validation dataset, which was discussed in Chapter 4.1.5.

Task type	Workspace objects	Amount
bolt to box	bolt, pushrod	2
bolt to box	bolt, rocker arm	2
bolt to box	all 3 objects	2
bolt to box	multiples of all 3 objects	2
bolt to box	multiples of all and unseen objects	2
pushrod to box	bolt, pushrod	2
pushrod to box	pushrod, rocker arm	2
pushrod to box	all 3 objects	2
pushrod to box	multiples of all 3 objects	2
pushrod to box	multiples of all and unseen objects	2
rocker arm to box	bolt, rocker arm	2
rocker arm to box	pushrod, rocker arm	2
rocker arm to box	all 3 objects	2
rocker arm to box	multiples of all 3 objects	2
rocker arm to box	multiples of all, unseen objects	2
bolt to specific box	bolt, both goal boxes	4
pushrod to specific box	pushrod, both goal boxes	3
rocker arm to specific box	rocker arm, both goal boxes	3
sequential item moving	4 bolts, 3 rocker arms, 3 pushrods, 3 goal boxes	10

Table 4.8. *Final model cluttered task breakup*

5. RESULTS

This chapter discusses the accuracy of the final model with examples taken during task execution. The model success is also compared to two prototype models and their purpose in making the final model is explained. Other prototype models are also briefly discussed. Each of the images are captures from the python library matplotlib, and inadvertently include a text over the image: "prediction cast pick/place (blue/cyan) vs. actual (red/pink)". This text is there to clarify image context during the data collection phase and signifies that the world-space coordinates predicted by the CLIPort framework have been transformed back into image-space coordinates and drawn over the presented RGB image. The bounding boxes follow the schema discussed in Chapter 4.2 and in general the four views presented by figures such as Figure 5.1 are the same views as in Chapter 4 but follow a different colour scheme and pixel scale. Sections of this chapter aim to:

- Present & discuss the final model
- Present & discuss the prototype models
- Present & discuss the success criteria & shortfalls
- Present & discuss notable common errors
- Present & discuss factors contributing to common errors
- Present & discuss discovered possibilities for improvement
- Present & discuss success rates for all models

5.1 Final model examples

This chapter discusses examples of typical pick & place situations with four-view examples following the schema outlaid in Chapter 4. Every example in presented here in Chapter 5 is faulty in some unique way, but the faults are mostly presented as a case for reinforcing the decision to not follow the multi-peak solution for multi-step workflow in Chapter 3.6. Similar faults generally do not cause the whole task to fail, but some systematic failures were observed.

Figure 5.1 presents the RGB-, heightmap-, pick confidence-, and place confidence views in order from top-left to bottom-right as outlaid in Chapter 4.2.1 for a generic bolt place

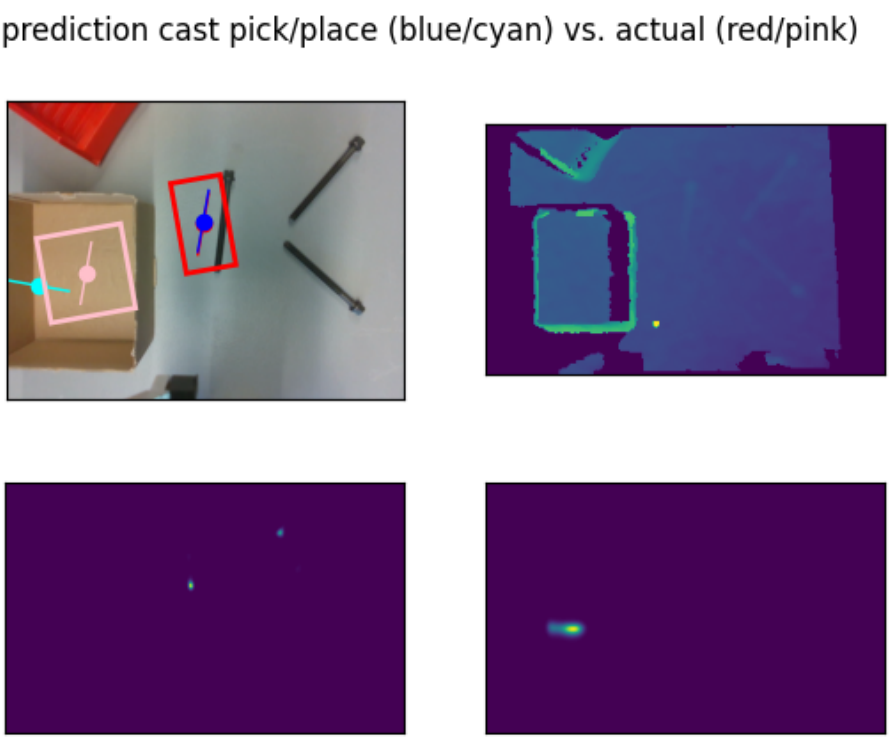


Figure 5.1. *An RGB image with classification points (top left), height-map (top right), pick confidence (bottom left) and place confidence (bottom right) scenario showcasing a missing bolt in the pick confidence map for a "put bolt in brown box"-task*

task with the final model, where the pick position can be considered optimal compared to the recorded position. The recorded correct pick position is marked with the blue dot representing position and a blue line representing the correct pick angle, and the predicted pick position is similarly marked with red along with a red box denoting the mathematical policy to gauge success. Note that unlike in later examples, the position is so accurate that most of the red colour for the position and angle in the predicted pick is not visible. The recorded place position is marked with a cyan dot and line also representing position and angle respectively, and the predicted place position is denoted by a pink dot, line and square for place, angle, and policy respectively.

The place position is not as successful and the presented policy (pink square) considers the task failed even though with manual inspection we can tell that the place position is within the goal box, even when considering the leftward bias exhibited by the pick position. The task presented by Figure 5.1 also demonstrates why the leftward bias was not considered to be detrimental in Chapter 4.2.1 or to inhibit the function of the framework. The only relevant success metric has the prediction be close to the recorded position and all of the recorded positions were successful action executions on the real robot. Having the real robot move to a prediction close to the position demonstrated by the recorded

data and being in correct orientation should result in an action which closely resembles the recorded and successful action. It is unclear how the position bias affects CLIPort predictions though, it could be argued that the predictions would be more accurate if they were made to the middle of the object (center of detection) instead of somewhere left of it, but the effect of this was not measured. Also, during the thesis work and implementation process it was postulated that the positional error is caused due to bad camera transformations, since actual pick positions were generally not observed to differ much from a perceived central line of the object. In other words, the object pick positions looked correct, and the object didn't roll or move significantly when the claw fingers closed which signifies centrality in the prediction process.

Another noteworthy aspect demonstrated in Figure 5.1 is that in the pick confidence window (on the bottom left) does not contain a confidence peak for all the bolts that are present in the workspace. The three bolts are visible and unobstructed so in an optimal case, the framework would have produced three confidence peaks with a bias toward the closest bolt. As discussed in Chapter 4.1.2, the dataset was made to always prefer the closest valid object except in some cases as will be seen in Figure 5.2. The lack of all peaks in Figure 5.1 is also supporting evidence for the choice to not continue developing the multi-peak confidence culling approach and instead opt for the repeated single predictions approach when developing the multitask functionality, which was discussed in Chapter 3.6.

Figure 5.2 presents the RGB-, heightmap-, pick confidence-, and place confidence views in order from top-left to bottom-right as outlaid in Chapter 4.2.1 for a task, where the framework behavior was investigated on tasks where there were two roughly equal choices. The dataset contains an example for a similar case where the bolt to the right of the box would've been the demonstrated and recorded bolt position. Likely due to carelessness, the bolt that is further a way from the box is considered correct which goes in direct opposition to the intended structure of the dataset, as it was intended that the correct option would've always been the bolt closest to the box. In this case, however, the bolt is so close to the box that it's likely that the claw and by extension the robot couldn't actually reach the goal object and would instead collide with the brown box presenting an example for a task-stalling error which are further discussed in Chapter 5.5. In considering prediction success, object collisions were not considered and this task would count as a successful execution in the accuracy prediction and the values presented in Table 5.1. Unlike in Figure 5.1, the place policy correctly identifies the place prediction to be successful.

Figure 5.3 presents the RGB-, heightmap-, pick confidence-, and place confidence views in order from top-left to bottom-right as outlaid in Chapter 4.2.1 for a task, where the framework behaviour is investigated again in a task where there are multiple equally valid goal objects. The pick confidence does not exhibit all the valid bolts, as one bolt is partial and thus not detected. The prediction also favours a wrong bolt, even though we can see

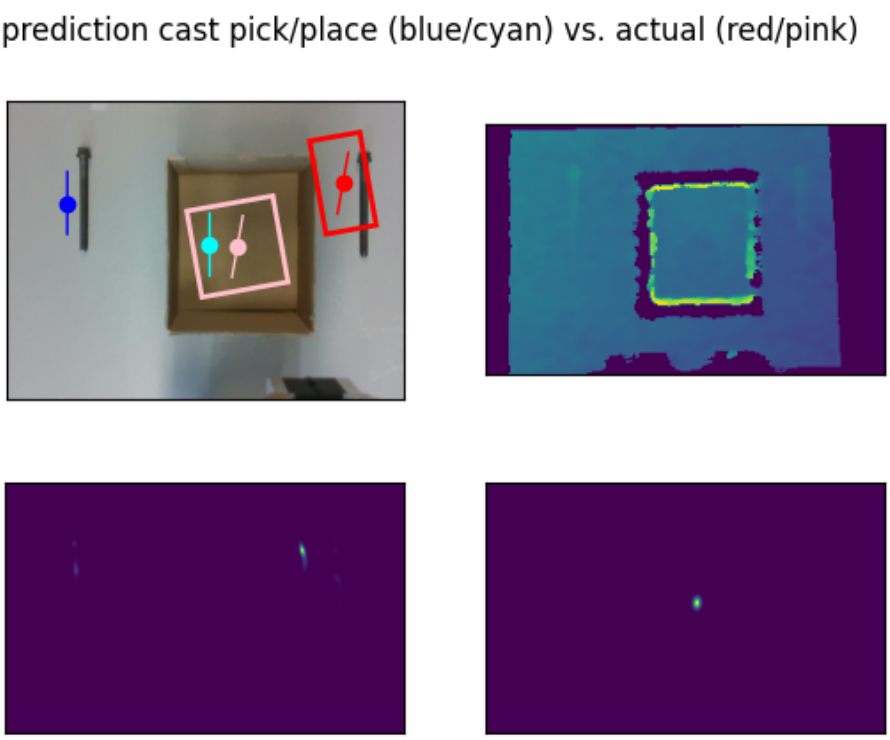


Figure 5.2. *The 4 views presented in Chapter 4.2 showcasing a scenario where a wrong bolt is picked and is falsely classified as a fail in a "put bolt in brown box"-task.*

from the pick confidence image that the correct bolt was an option as well. The policy incorrectly fails the task even though in a real-world execution, the task goal of "put bolt in brown box" would be successful since it does not strictly define that the leftmost bolt should be moved. The place policy also fails to detect a successful placement much like previously in Figure 5.1. A bolt is also present inside the goal object which is correctly not detected, as the framework should never move a bolt from inside the box back into itself, even though this is also not strictly forbidden by the language goal.

Figure 5.4 presents the RGB-, heightmap-, pick confidence-, and place confidence views in order from top-left to bottom-right as outlaid in Chapter 4.2.1 for a task, where the framework behaviour is investigated in a task where the object was not taught in any demonstration. The place goal is a known goal, and since the pick & place predictions are done separately, these actions are largely successful. For the pick task, it was postulated that CLIPort could infer the colour green from the image and have a partially accurate prediction from just that information, since CLIPort itself has been demonstrated to be able to discern between different colours. We see something similar to this behaviour from the left-side prediction bias exhibited by the bottom-left image in Figure 5.4, but this is by no means conclusive and can't be considered supportive evidence. In general, when the prediction network was confused, it was observed to have a very significant

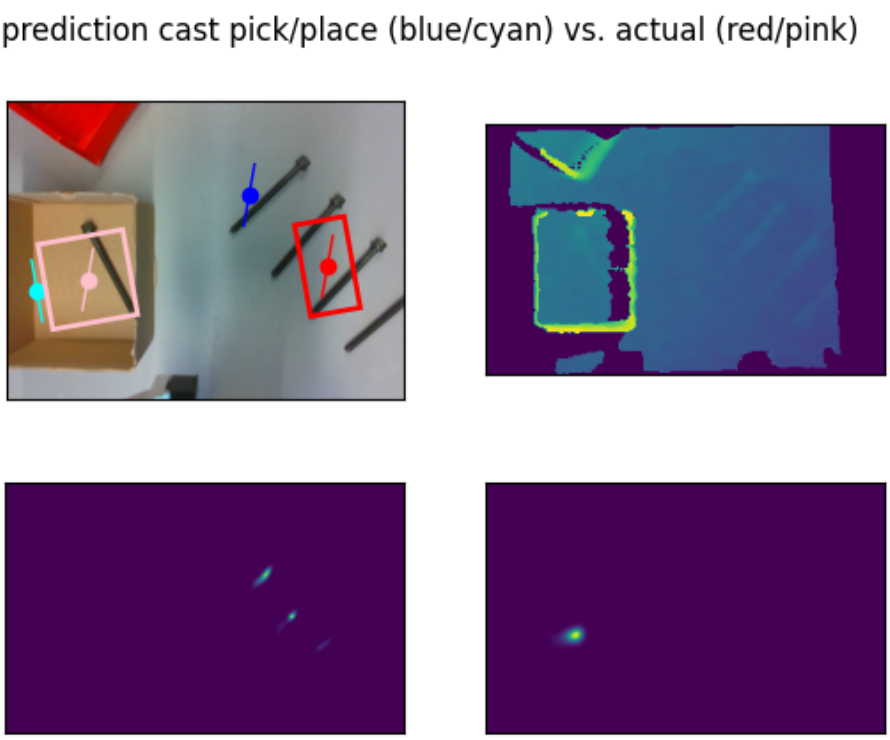


Figure 5.3. *The 4 views presented in Chapter 4.2 showcasing a scenario where a wrong bolt is picked and is falsely classified as a fail in a "put bolt in brown box"-task. One bolt is not given as an option.*

bias to the top edge of the image. Please note as well that this behaviour is very much exaggerated by the chosen colour map, and the fact that every similar view presented with normalized data to see peaks with the best clarity. When these random data predictions were observed during the thesis work, they contained no significant peaks but rather a mat of very small peaks, many orders of magnitude smaller than the prediction peaks seen in previous figures such as Figure 5.1 or Figure 5.2.

Figure 5.5 presents the RGB-, heightmap-, pick confidence-, and place confidence views, in order from top-left to bottom-right as outlaid in Chapter 4.2.1 for a persistent error with "put push rod in red box"-tasks. There are examples of "put push rod in red box" in the final model dataset, as seen in Table 4.1, but the place prediction was observed to be consistently wrong and relatively in the same position, outside the bottom-right corner of the workspace. This is also interesting because the place prediction is done irrespective of the picked item and the place tasks concerning the red box were successful with different objects. The cause of this error was not discovered during the thesis work due to time restraints, but the behaviour was not observed in real-robot test runs. These errors were not omitted from the weighted average success rates in the final model success discussion. The place task success for task 3 can be seen as 0% in the success table

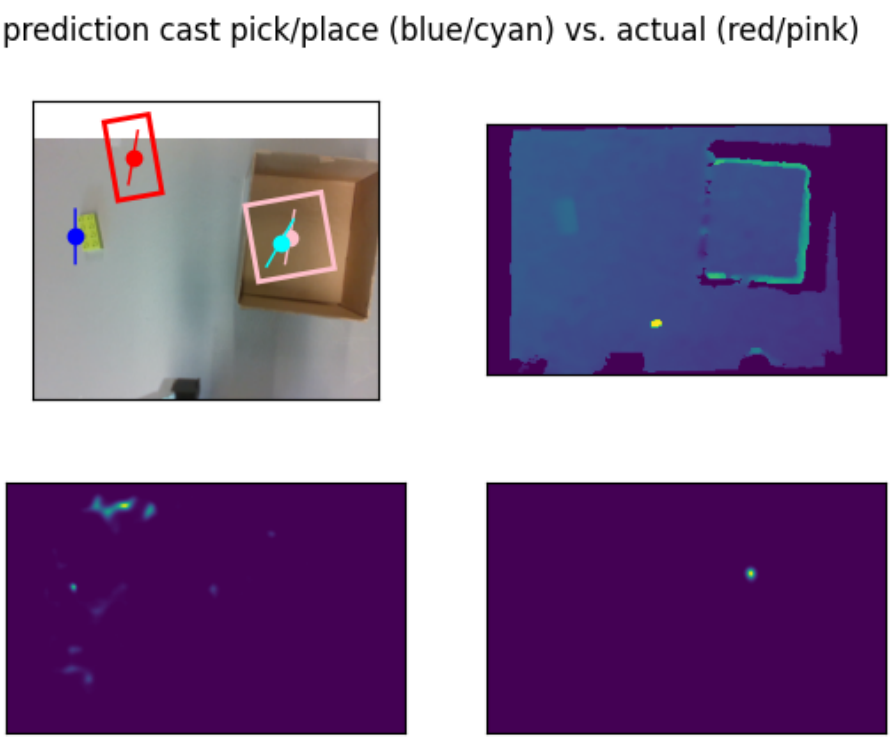


Figure 5.4. *The 4 views presented in Chapter 4.2 showcasing a scenario where the framework is asked to do a pick prediction on an object it hasn't seen before. The pick confidence is random while the place object is normal since it is known.*

later due to none of these tasks being successful, which artificially brought the accuracy of the model down.

Figure 5.6 presents presents the RGB-, heightmap-, pick confidence-, and place confidence views, in order from top-left to bottom-right as outlaid in Chapter 4.2.1 for a generic and very successful "put rocker arm in red box" task. The case is also presented as further supporting evidence to the decision to not pursue the previously discussed multi-step workflow in Chapter 3.6. Trying to do peak-culling to the pick-confidence map presented in the bottom-left image in Figure 5.6 would result in many peaks being created for the same rocker arm before the one on the left side would be found. There is also noise that creates the colour illusion of a rocker arm present right of the chosen rocker arm, which might be detected. In a badly chosen peak selection, all of these four positions would be considered valid locations, and the framework would command the robot to execute unnecessary moves. Trying to create a more sophisticated policy which would do things such as peak cluster detection was considered too difficult due to some objects being presented in clusters in some examples.

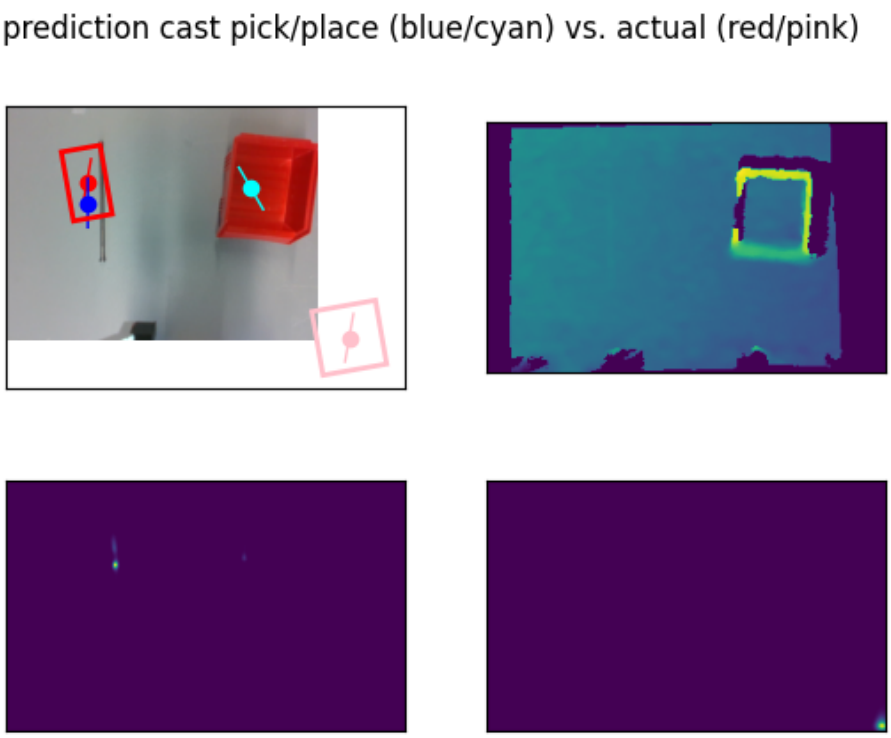


Figure 5.5. *The 4 views presented in Chapter 4.2 showcasing a typical scenario of a task the framework failed repeatedly in the same spot for a "put push rod in red box"-task.*

5.2 Final model success rate

This section displays the success rates for the final trained model in success tables along with some other quality criteria presented in box plots. The quality criteria presented in the box plots were intended to be utilized in selecting the final model, but due to implementation difficulties did not provide useful information in a timely fashion. The criteria are presented anyways since they give context to the thought process behind validating which samples were successful and which were not.

The final model success rates per task are listed in Table 5.1, where *policy pick* and *policy place* are the success rates evaluated with the policy that was introduced in Chapter 4.2.2, and *actual pick* and *actual place* being the success rates as evaluated by a human actor. The tasks under **task** use the same aliases as presented earlier in Chapter 4.3 Table 4.5. The total average percentage (**tot. avg.**) is the average success percentage of all measured tasks and the final percentage (**wgt. avg.**) represents the average success for the tasks that contain objects the model was trained to recognize.

The weighted average for *actual pick* & *actual place* are the numbers which best represent the real-world accuracy of the framework. We can see that the policy is not very successful in evaluating the actual framework success rate, being too strict and having a

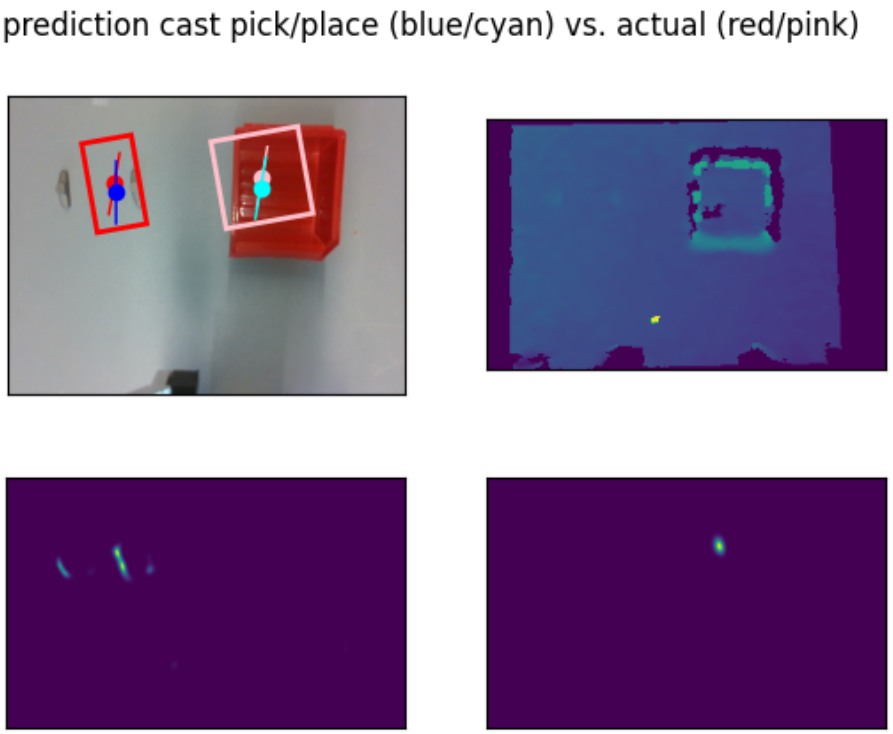


Figure 5.6. *The 4 views presented in Chapter 4.2 showcasing a scenario where the table texture is confusing the pick prediction for a "put rocker arm in red box"-task.*

The final model				
task	**policy pick**	**policy place**	**actual pick**	**actual place**
task 1	66.67%	91.67%	100.00%	100.00%
task 2	63.16%	68.42%	89.47%	89.47%
task 3	50.00%	0.00%	100.00%	0.00%
task 4	26.32%	89.67%	73.68%	100.00%
task 5	91.67%	91.67%	100.00%	91.67%
task 6	50.00%	87.50%	81.25%	100.00%
task 7	0.00%	50.00%	0.00%	100.00%
tot. avg.	**49.69%**	**68.39%**	**77.78%**	**83.02%**
wgt. avg.	**57.67%**	**67.71%**	**90.42%**	**83.02%**

Table 5.1. *The final model task successes*

success difference of up to 32.72%. This difference is mostly due to the dataset emphasis on multiple goal object presentation in the same image and the rigidness of the success policy evaluation around a single correct option among equals as exhibited by examples seen in previous figures, such as Figure 5.2.

The final model achieves a combined task success rate of **86.72%** for every taught task, with a **75.07%** rate for flawless task execution.

For pick items, the model exhibited a combined task success rate of 94.74% for bolt tasks, a combined task success rate of 68.42% for push rod tasks, and a combined task success rate of 93.23% for rocker arm tasks.

For goals, the model exhibited a combined task success rate of 81.95% for red box tasks and a combined task success rate of 88.91% for brown box tasks.

The most successful task for the final model was "put bolt in red box" with a combined task accuracy of 100.00%. It is noteworthy that this might not hold if the dataset contained more examples, and the model was perceived to sometimes fail in this task in real world executions due to reasons discussed in Chapter 5.5.

Compared to comparable models that were found during the literature review and research portion of the thesis work, a success rate of 86.72% seems comparable to existing solutions but is slightly lower than expected. For the handover task, which does not consider the place prediction, this accuracy goes up to 90.42%. For reference, LEMMA [22] was reported to have an accuracy of 87±0.67 (%) in comparable tasks and PALM-E [28] was reported to have an accuracy of 94.9% in comparable tasks. Both of these accuracies were from simulated executions with thousands of more repetitions than the real-world data predictions presented in this thesis.

5.2.1 Final model quality criteria

Some additional quality-assessing measurements were also done on the models. These were done on a task basis and two are presented.

Figure 5.7 presents the box plots for the measured delta values for the task "put bolt in red box" in units that are inconclusively presumed to be meters. The measurements are based on the absolute Pythagorean distance delta values from the predicted position to the recorded position as reported by CLIPort compared to the recorded datapoint X-, Y-, and Z-position. As a task, "put bolt in red box" is known and should represent very successful distance measurements. From Table 5.1 on task 1 we see, that 100% of these tasks were considered successful for both the pick & place actions. In the chart, values closer to 0 are considered better as this implies better prediction accuracy.

Figure 5.8 presents the box plots of location delta values for the task "put green lego in

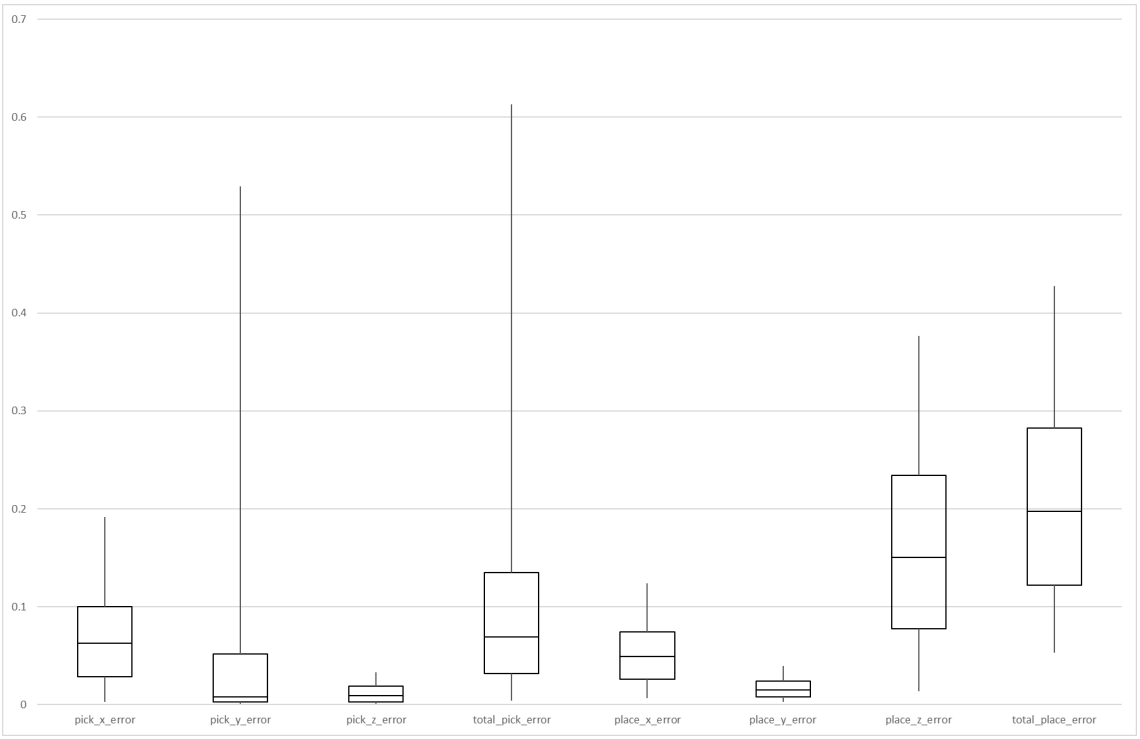


Figure 5.7. *Box plot of all "put bolt in red box"-task accuracies measured in task-space units for the final model.*

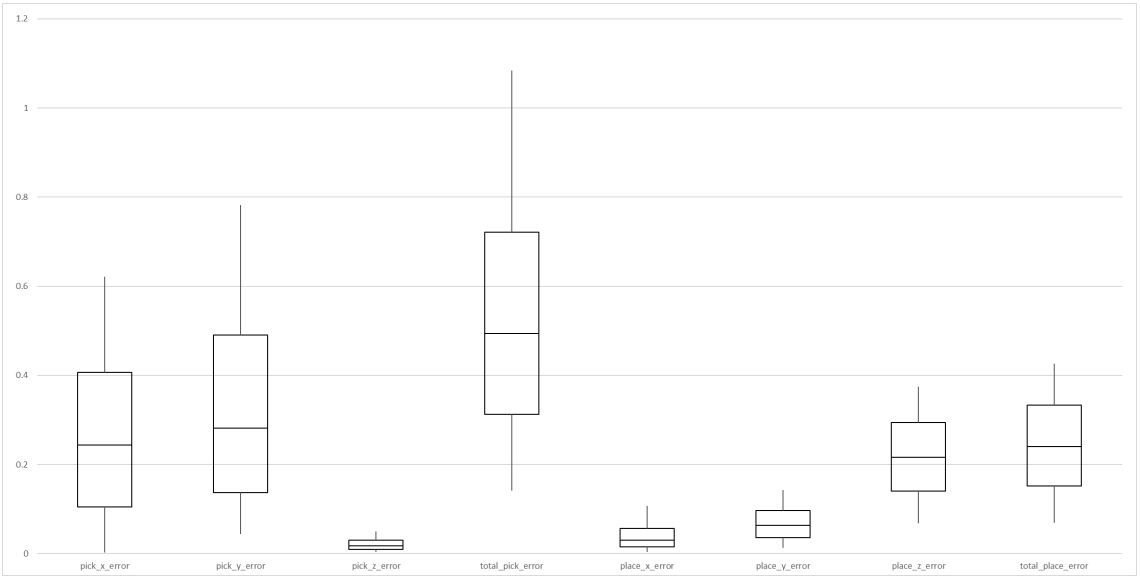


Figure 5.8. *Box plot of all "put green lego in brown box"-task accuracies measured in task-space units for the final model.*

brown box" in the same units as in Figure 5.7. This task is unknown and represents the untrained behavior of CLIPort. From Table 5.1 task 7 we see, that 0% of these tasks were considered successful for the pick action, while 100% of the place actions would've been successful.

From both box plot figures discussed for the final model we can see that the Z-dimension prediction is by far the most accurate for the pick action, while it seems to lose a lot of quality with the place action. This is probably since the place actions varied in height in the demonstration data due to the different shapes of the boxes, while the objects-to-be-moved were always picked up from the workspace desk surface. In Figure 5.7 we see that all average distance errors except place-z-error fall under the 0.1 line on average, which is still quite far compared to the object dimensions.

As stated before, Figure 5.8 data from the "put green lego in brown box" task represents random location data, and we can see that the pick action suffers and exhibits much worse distances than in the properly functioning "put bolt in red box" task data. Curiously, the pick z height is still largely accurate, and the place action seems comparable to the red box place distances in Figure 5.7. One noteworthy aspect is that the brown box is larger than the red box, which contributes to the greater distances CLIPort could predict from the demonstration point while still being inside the goal object.

5.3 Other models

This chapter discusses some model-related details and errors for the two selected prototype models. Performance discussion and comparison is done in Chapter 5.4

5.3.1 The multistep model

This section discusses an older model that was used in investigating the multitask functionality in CLIPort. The tests ultimately proved inadequate for the desired functionality and no significant increase in detection accuracy was observed. Two examples are examined, one for poor performance and another for desired performance. These are presented in Figures 5.9 and Figure 5.10. As can be seen from Table 4.6, this dataset was trained for the shortest duration of time and its function might have improved with longer training times. However, other prototype models, that were discarded during the thesis work, and which are not presented, showed faster rates of improvement. The prediction portion of the multistep model also seemed to not work in the test set tasks, manifesting as a 0% placement accuracy in Table 5.2.

As discussed in Chapter 4.1.3, this model (the multistep model) is different from the final model and the extension model in that the recorded data points had multiple demonstrations marked in a single demonstration image. The idea was that the training model would

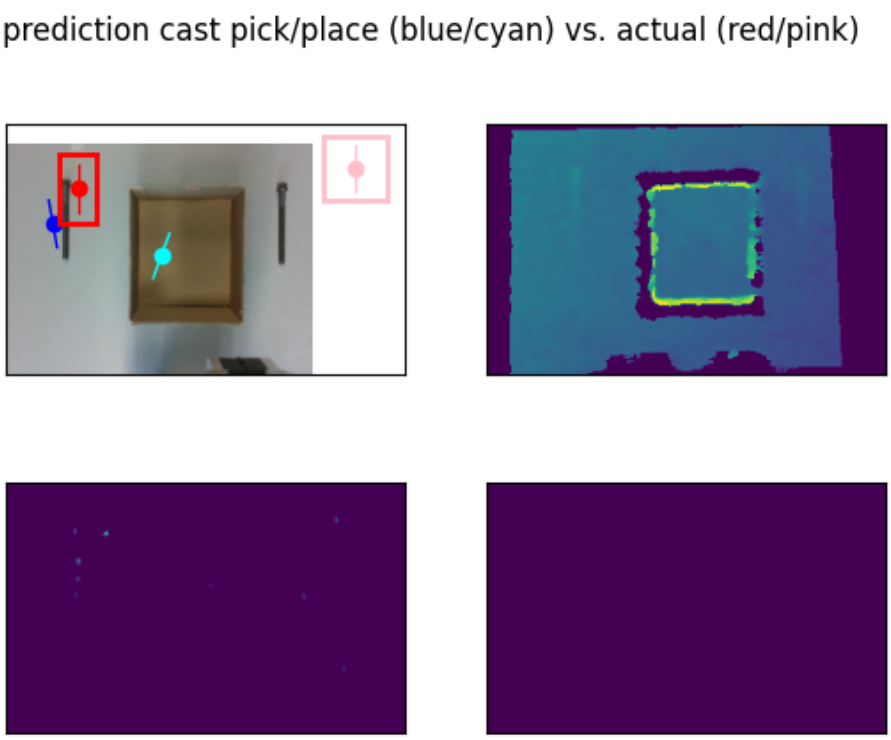


Figure 5.9. *The 4 views presented in Chapter 4.2 showcasing a prediction done with the model discussed in Chapter 4.1.3 for a "put all bolts in brown box"-task. The place fails in a way typical for the model and there are too many options for the bolts.*

better understand the shape of a given goal object when they are in clusters if all of them had a training demonstration at some point. The previously discussed policy in Chapter 4.2.2 would also be more suitable to a demonstration set where all of the options would have been marked. To facilitate multiple sample training, the functionality of the previously implemented framework was extended, but since the multistep model results exhibited no proof of desired functionality, these changes were deemed irrelevant and won't be further discussed in the thesis.

Figure 5.9 presents presents the RGB-, heightmap-, pick confidence-, and place confidence views, in order from top-left to bottom-right as outlaid in Chapter 4.2.1 for a "put all bolts in brown box"-task. The example image is the from the validation set as presented earlier in Figure 5.2 but the scale of the RGB image is changed due to the pink placement prediction being outside of the workspace image coordinates. This image serves as an example of two failures for the second model, first being the persistent failure of the placement module and the second being unpredictable confidence noise. The pick confidence image on the bottom left displays 9 confidence peaks of which a decently good prediction is selected because the pick can be visually considered successful. Pursuing a policy which would have utilized peak selection from the confidence map would be impossible

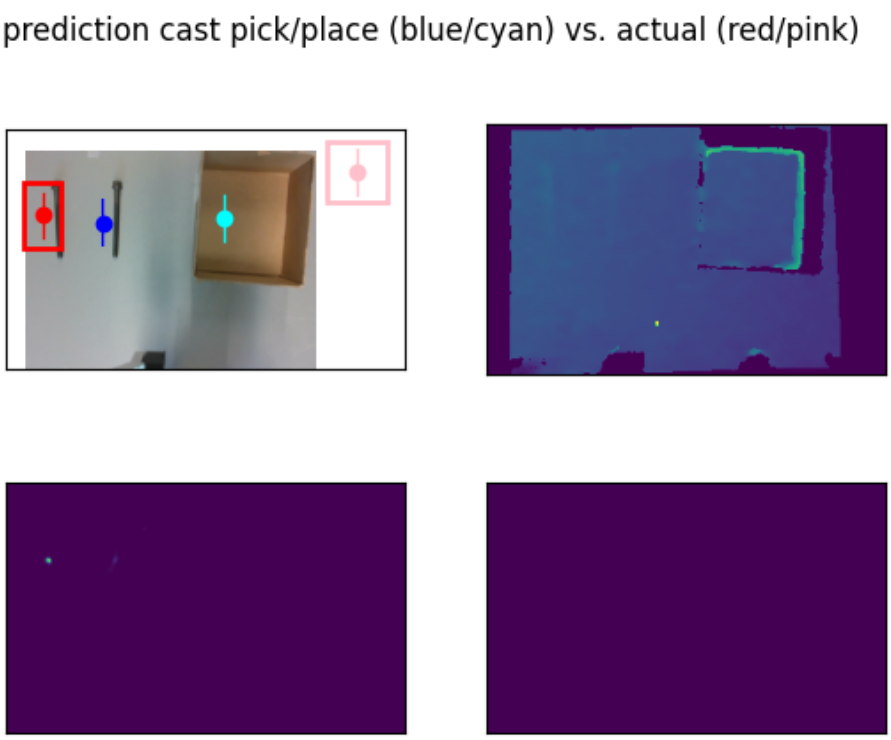


Figure 5.10. *The 4 views presented in Chapter 4.2 showcasing a prediction done with the model discussed in Chapter 4.1.3 for a "put all bolts in brown box"-task. The place fails in a way typical for the model but there are a right amount of options for the bolts. Note that this correct behavior was perceived much less commonly than the incorrect behavior.*

with this image if the goal was to move both bolts into the brown box, since the other bolt is just barely detected. It's difficult to say where exactly the three rightmost confidence peaks land on the actual image due to the scale changes caused by the out-of-bounds place prediction, but it could be argued that executing 9 moves for two bolts would seem faulty to a collaborative human operator and they would terminate the operation due to fears of malfunction. Due to time constraints and the scarcity of training time allocated to the used computers, this model was not trained further, and it remains unclear whether this performance would approach the desired functionality of producing two roughly equal peaks.

Figure 5.10 presents presents the RGB-, heightmap-, pick confidence-, and place confidence views, in order from top-left to bottom-right as outlaid in Chapter 4.2.1 for a "put all bolts in brown box"-task, which was considered to be a successful from a policy-implementing perspective and could be considered an example of the model functioning as hoped and intended during the thesis research and implementation process. The function of the place module is irrelevant for multistep model since it was not changed from the extension model or the final model which might indicate that the training time was

underutilized. The pick confidence view on the bottom-left corner displays two confidence peaks roughly where we would expect the two bolts from the RGB view on the top-left to appear. These sort of contained peaks could be detected with a confidence cutoff value to create multiple consecutive tasks that would have the robot go to the centre of these points. This type of behaviour was very rare with the examples from the validation dataset and can not be considered typical for the model.

5.3.2 The extension model

The extension model was the first model trained during this thesis. It was trained on a dataset which was extended from the dataset manufactured during the prior work, [1]. After this dataset was successfully extended and a model was sufficiently functional, this approach was left as it was more of an implementation guideline for future models. The objects present in the old dataset were not considered relevant enough to the overall goal of engine assembly and thus new example recording was deemed wasted effort. The previously recorded examples could not be included in future datasets due to the camera issues mentioned in Chapter 4.1.4. This model does not have any example figure presenting the four views for a template task but will still be compared to the other models presented in the thesis.

The key discovery for work on this model's performance and dataset was the importance of having a largely balanced dataset in regards to sample object amounts. The added bolt tasks did not feature the other objects in the old dataset, and the performance of the model seemed to have decreased compared to the one implemented in the prior work, [1]. The old dataset also used a very structured form of data in experiment complexity variation with most examples being largely similar, in opposition to the datasets implemented during this work which had a focus on variety outside examples with just single objects. Another possibility for the poor performance of the model is that the bolt examples always featured just a single bolt and the goal, and thus the prediction was confused if any other object was present in the workspace. This possibility is however not supported by the collected data as can be seen from Table 5.3, where the pick task fails totally.

5.4 Other model success rates

Since both of the other presented models were mainly used for furthering the implementation of the framework as well as testing, their performance discussion will be largely superficial with a focus on comparisons with the function of the final model. Table 5.2 and Table 5.3 display the success rates for two other models used during investigation and noted in this thesis. They use the same parameter aliases as the success table for the final model, Table 5.1 in Chapter 5.2, and the alias list can be found in Table 4.5. Table 5.2

The multistep model				
task	**policy pick**	**policy place**	**actual pick**	**actual place**
task 4	0.00%	0.00%	6.25%	0.00%
task 5	8.33%	0.00%	16.67%	0.00%
task 7	0.00%	0.00%	0.00%	0.00%
task 8	26.32%	0.00%	47.37%	0.00%
task 9	52.63%	0.00%	78.95%	0.00%
tot. avg.	**17.46%**	**0.00%**	**29.85%**	**0.00%**
wgt. avg.	**39.47%**	**0.00%**	**63.16%**	**0.00%**

Table 5.2. *The multistep model task successes*

presents the success rates for the multistep model and Table 5.3 presents the success rates for the extension model. Note that some models use different task names for the same object due to unifying object names with other projects between models.

5.4.1 The multistep model success rate examination

Following the pattern in Chapter 5.2, the multistep model success rates per task are listed in Table 5.2, where *policy pick* and *policy place* are the policy success rates, and *actual pick* and *actual place* are the success rates evaluated by a human actor. The tasks under **task** use the same aliases as presented earlier in Chapter 4.3 Table 4.5. The total average percentage (**tot. avg.**) is the average success percentage of all measured tasks and the final percentage (**wgt. avg.**) represents the average success for the tasks that contain objects the model was trained to recognize. The pick module worked surprisingly often for tasks 4 and 5 even though the rocker arm was not a trained object as exhibited by Table 4.2. These success rates are likely due to random chance, but in the extension model performance discussion we will see that random actions typically resulted in a 0% success rate.

The multistep model achieves a combined task success rate of **31.78%**. None of the place tasks were successful as the place module was completely nonfunctional with the validation dataset even though it was observed to function on the real robot after training. Since the place task never succeeded in validation, the multistep model achieves a flawless task execution rate of **0%**.

For tasks that the framework understood, it achieves a pick accuracy of 63.16% with the pick portion of the task "put all push rods in brown box" being 78.95% accurate. Note that even though the language goal has the words "put all", framework execution would only move one push rod due to the multi-step workflow followed in the thesis and this sort of multi-step functionality being abandoned in implementation. Thus, the pick module could

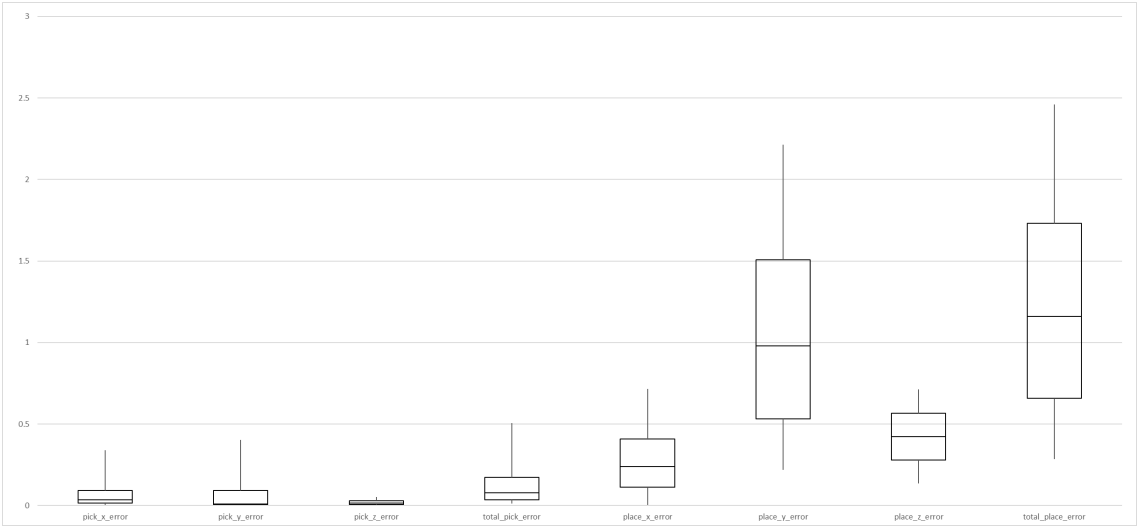


Figure 5.11. *Box plot of all "put all push rods in brown box"-task accuracies measured in task-space units for the multistep model.*

be used for single object handover tasks, with the same accuracy as the average pick tasks.

Compared to the final model, we see that even when omitting the non-functional place module, the final model achieves a better performance in known pick tasks. For the final model, only its two worst performing tasks, task 4 and task 6, are comparable in performance to the best performing task of the multistep model, task 9. The comparatively low training time of the multistep model is likely a large factor in this, but during the thesis work models with a similarly short training time provided better or comparable results. This coupled with the fact that multi-step datasets were much more laborious to collect was also a factor in deeming the approach irrelevant and to pursue the multistep workflow discussed in Chapter 3.6. There were also other benefits to the chosen implementation which are also discussed in Chapter 3.6.

Figure 5.11 presents the box plot of location delta values for the task "put all push rods in brown box" in the same units as in Figure 5.7. This task is the best performing out of the tasks done on the multistep model and represents the optimal behaviour of the model. From Table 5.2 task 9 we see, that 78.95% of these tasks were considered successful for the pick action, while 0% of the place actions were. Compared to the untrained behaviour discussed in Chapter 5.2 with Figure 5.8, we can see that the place errors of Figure 5.11 follow a similar pattern to the random pick errors in Figure 5.8 with much higher inaccuracies in the random portion. For the multistep model, since the predictions were always outside of the workspace, the difference between the working pick portion and the random place portion are especially large. Even though it is difficult to see from the figure, the pick location accuracies largely resemble the location accuracies of properly working tasks in the final model, such as in Figure 5.11 and actually exhibit smaller extremes with

The extension model				
task	**policy pick**	**policy place**	**actual pick**	**actual place**
task 1	0.00%	0,00%	0.00%	0.00%
task 2	0.00%	15.79%	0.00%	52.63%
task 3	0.00%	0.00%	0.00%	0.00%
task 4	0.00%	10.53%	0.00%	52.63%
task 5	0.00%	0.00%	0.00%	0.00%
task 6	0.00%	25.00%	0.00%	56.25%
task 7	0.00%	25.00%	0.00%	25.00%
tot. avg.	**0.00%**	**10.90%**	**0.00%**	**26.64%**
wgt. avg.	**0.00%**	**15.26%**	**0.00%**	**46.63%**

Table 5.3. *The extension model task successes*

a slightly better total pick error. Note as well that all models were validated with the same unseen dataset, so there should be no preferential bias over any of the datasets used for training. As discussed in Chapter 4.1.5, the dataset was separately collected after the training process was complete for all models.

5.4.2 The extension model success rate examination

Following the formula of the two previously discussed success rate tables in Chapter 4.1.2 and Chapter 4.1.3, the extension model success rates per task are listed in Table 5.3, where *policy pick* and *policy place* are the policy success rates, and *actual pick* and *actual place* are the success rates evaluated by a human actor. The tasks under **task** use the same aliases as presented earlier in Chapter 4.3 Table 4.5. The total average percentage (**tot. avg.**) is the average success percentage of all measured tasks and the final percentage (**wgt. avg.**) represents the average success for the tasks that contain objects the model was trained to recognize.

One of the most surprising aspects of the validation process was that the original model worked so poorly with the validation data, since it was deemed a successful model on the real robot. The red box as a goal object was not present in any of the training data so it being at 0% was expected, but the long screw tasks (aliased tasks 1 and 2) failed. In addition to this, the green lego task was also not functional even though it utilized data from the older dataset and should have achieved comparable performance to the known tasks with the final model in Table 5.1.

Compared to the final model this model does not seem functional, but it's main purpose for the thesis was to aid in investigating the function of the work done in the past. The place module seems to function in some capacity and achieves a place task success

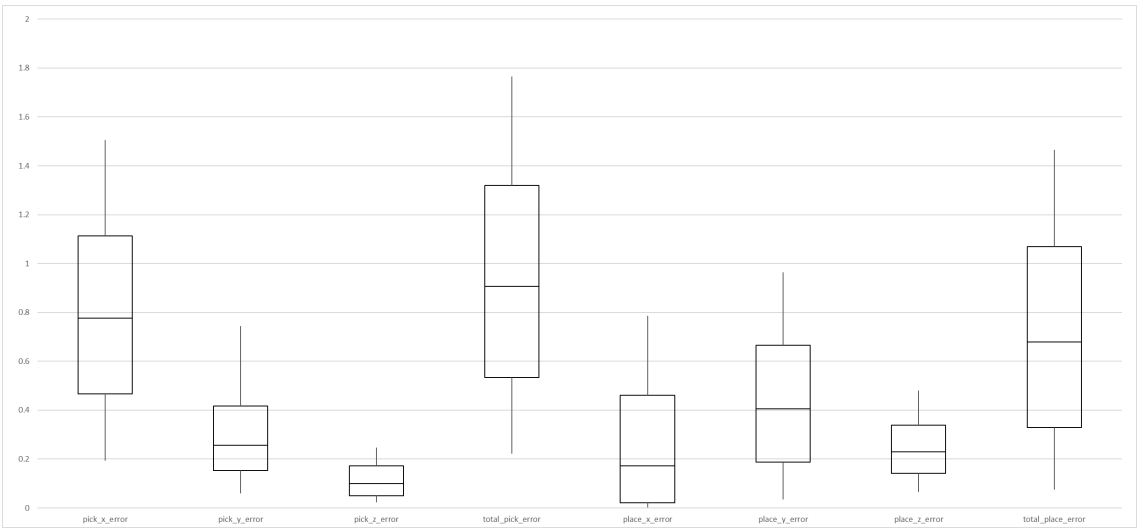


Figure 5.12. *Box plot of all "put green lego in brown box"-task accuracies measured in task-space units for the extension model.*

rate of 46.63%. As a whole, the extension model achieves a combined task success rate of **18.65%** and a flawless task execution rate of **0%**, since the pick module was not perceived to function on the validation dataset. The best understood task was "put rocker arm in brown box" with the place portion of the task being 78.95% accurate. Curiously the rocker arm was not present in the model training dataset, but the goal object, brown box, was present. This is possible since the place prediction was done simultaneously but separately from the pick prediction, and irrespective of what object the robot would be carrying.

Figure 5.12 presents the box plot values for the location deltas on the task "put green lego in brown box" in the same units as in Figure 5.7. The location errors seem to be worse than in the previously discussed box plot figures either for the multistep model or the final model. Compared to the values presented in the box plot figures in Chapter 5.2, even the somewhat functional place positions seem quite bad. The green lego task for this model represents a known task, unlike with the other models, so it was expected that this performance would resemble the accuracy of task 1 for the final model.

Figure 5.13 presents the box plot values for the location deltas on the task "put long screw in brown box" in the same units as in Figure 5.7. Note that "long screw" is an old alias for the bolt object. This chart was intended as a location accuracy comparison to the green lego accuracy Figure 5.12, with the intent to discuss the impact of a smaller example set on the prediction accuracy. For the pick portion, we can see that the two tasks exhibit very similar total pick errors, which is to be expected given the lack of function displayed in Table 5.3. The place accuracies exhibit a smaller variance compared to the green lego task, which is reflected in the 27.63% increase in rate of success when comparing task 7 to task 2.

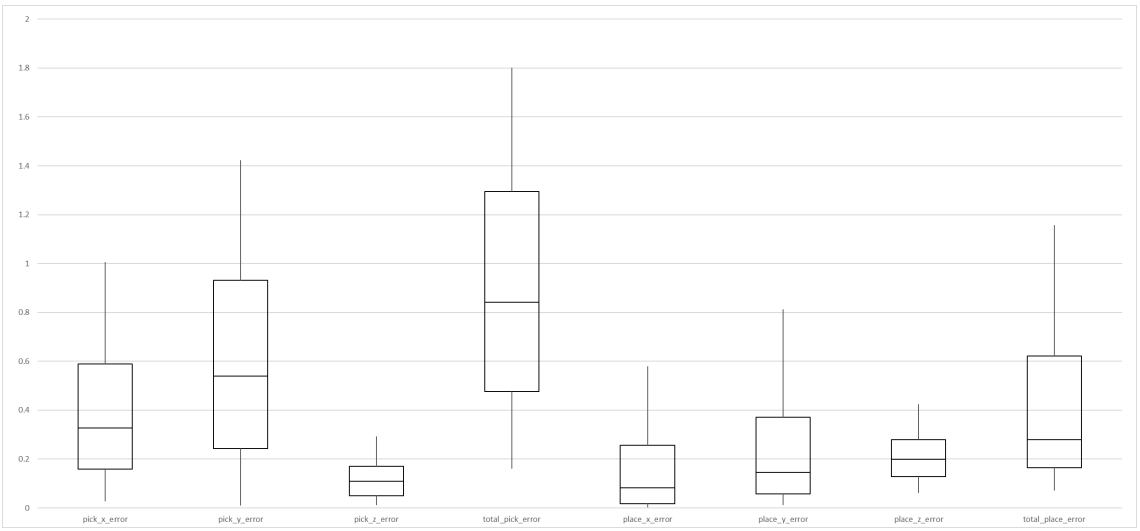


Figure 5.13. *Box plot of all "put long screw in brown box"-task accuracies measured in task-space units for the extension model.*

5.5 Discussion on errors

Multiple things in the workspace can cause the task to fail, and during the thesis work it was observed that operator error was the most common. The most disruptive errors for the framework are referred to in this paper as task-stalling errors. These errors cause the collaborative robot to stop for some reason and often mandate a full restart of some modules which makes them especially disruptive to any workflow. The other type of errors presented in this chapter are workspace change causing errors, which were of special interest during the multistep workflow discussion in Chapter 3.6. These errors do not cause the task to stall or stop, but instead may fail due to other factors which are not predicted by the framework due to copious complexity which would necessitate a different approach, such as incorporating path-altering solutions like LATTE [26].

Most of the task-stalling errors can be categorized as follows:

- The robot collides with the object when trying to pick it up causing large unexpected forces and activating the cobot stop signal.
- The robot tries to move too fast causing swings and burst-like movement or vibrations which cause large forces and activate the cobot stop signal.
- An object the robot has picked up collides with some other solid object causing large forces. Such objects include the robot frame, other heavy task-space objects, human limbs, and the workspace desk surface.
- The predicted path has the robot assume a position which is possible in the MoveIt [44] control interface, but can’t be realized by the real robot. Joint limits are reached, and the robot stops to allow continued use.

All of these situations cause the robot to stall due to it conforming to the collaborative robot requirement of user safety. The robot must then be brought back online by the user cycling the emergency stop which necessitates restarting the robot control interface. To expedite this process and to lessen the required workload, an additional command, "recover", was added to the speech interface which will attempt to send the robot an "all clear" signal after which the robot might be able to move without additional user input. In the case where robot joint limits were reached, this signal will often not work, but in other cases it was perceived to be useful. All of the tasks which were interrupted by any of the states described above would count as fails in real-robot execution, but they are not considered in the simulated success rates for the robot. In general, most of these errors have little to do with the created framework and are often the result of user error in object placement or errors caused by components in the framework which were not developed during the thesis work, such as MoveIt [44]. Also like mentioned in Chapter 4.

The task execution process was observed to cause changes in the workspace in some common ways, which include:

- The robot successfully picks up a long item in the wrong orientation, causing it to turn and collide with another object. The workspace is changed due to the other object moving from this collision.
- The robot successfully picks up multiple sleek-profile objects that are closer together than the robot tool can open. The workspace is changed due to missing target objects. Additionally, the decreased grip the robot has on the objects due to unpredictable slippage can cause them to drop causing further changes in the workspace.
- The robot or an item it carries collides with the target container or another object during the place task. Since the model doesn't take the object shape into consideration, it is possible that the generated path goes either through the target object or has the picked-up object collide with the target, causing it to move. The workspace is changed due to the collided object being displaced from the predicted position.

These changes were also factors in the decision to pursue the discussed multi-step workflow discussed earlier in Chapter 3.6. In the cases where the collided objects moved away from the workspace, user input is required to bring them back if the task was concerning multi-step execution. In cases where just a single action was being executed, these collisions were perceived to only affect the task success a little.

5.5.1 Factors contributing to errors

Common factors contributing to errors during execution include:

- Poor lighting conditions, causing the goal object to mask with the surface material

- Wrong material in the background, causing confusion to the framework
- Distracting objects too close causing identification to fail
- Multiple equally valid goal objects being present in the workspace, causing the wrong to receive the strongest classification

Note that physical problems such as objects being too close to pick are not measured in the simulation on which the framework was benchmarked on, due to such interactions not being simulated from the images. Common problems relating to the physical space are discussed in Chapter 3.6, and such factors contribute to the errors in real-world execution. In the dataset examples, the goal objects were always placed in such a way that the object was possible to pick up and the goal was reached. These two conditions are guaranteed to be true in a successful task recording by the demonstration capture tool implemented by O. Hassan [1]. One physical problem-causing aspect that was not discussed previously is the fact that different materials or covers slightly raise the height of the items in the workspace. Due to how limits were implemented in the framework, if the table surface rises significantly compared to the robot base, the robot tool could collide with the surface before stopping and cause a stalled task. The height limits given to the position are hardcoded to go only to a certain depth due to unpredictability in the CLIPort z-dimension prediction.

5.6 Future works and improvements

This chapter discusses some possibilities for future improvements first with significant overhauls requiring rewrites. Possibilities for extending the current framework are presented with discussion, and finally dataset improvements are discussed. Note that the dataset changes would directly lead into model changes, which is why the models and datasets are considered in the same space instead of dedicating sections to both.

5.6.1 Framework re-write requiring improvements

The first and possibly the most interesting change to the framework would consider swapping the CLIPort prediction and training pipeline with some other state-of-the-art solution, such as VIMA or PALM-E. This along with the tools and pre-existing datasets could make for an interesting facet of examination especially in the context of training time compared to achieved quality. Pursuing either solution would likely require very significant changes to the existing solution and in the worst case would also require the recording of new datasets, which would also necessitate the changing of the data collection tool. Another major future improvement could attempt to better utilize the human in true collaboration and better-defined collaboration tasks. The presence of an actual engine with a manufacturing expert could be worth investigating since the goal of collaborative robotics is to

make manufacturing more efficient and the expert likely would need time to adjust to a different workflow. The framework might also be extended to multiple robots of the same model.

Another relevant improvement that could be made for the framework considers moving the camera solution from camera-in-hand to a camera-to-hand problem so that the whole robot body and workspace are perceived from a third-party perspective. Implementing this perspective change could have beneficial effects on task success rates as well as allowing the robot to react to moving humans without requiring physical contact. Implementing this approach would pose new problems such as having to place the camera in a known position every time, or implementing an auto-scaling locator tool for the camera that would run every time the framework was used.

5.6.2 Framework extension

Some deficiencies still exist with the user experience with the framework, and one possible extension could consider the implementation of a graphical user interface (*GUI*) to ease this. Significant problems and a major portion of the thesis work was also spent on just finding the correct launch parameters and -commands, to which a proper GUI could prove to be workload-reducing. In addition to these, future extensions might want to consider containerizing the whole application so that its development is quick and easy. In an optimal case, the framework would only have two shortcuts, one on each PC that are required for the prediction and robot moving.

Object avoidance is a relevant problem in collaborative robotics since human actors are often unpredictable and in the best cases, contact could be easily avoided as far as space limitations go. Introducing a path conforming pipeline, such as LEMMA, might improve the performance of the model as it would not collide with itself so often. Creating a well constrained MoveIt restriction would also serve to these purposes but only for static assets.

The language goals in the thesis were quite limited and do not really follow natural English language speech patterns by missing things such as articles before words. New more natural and ambiguous language goals could also be taught along with gestures allowing tasks such as "give that to me" etc.

The last thing to improve and the most obvious in the test examples would see the improvement of the success policy. The success policy in this thesis did not prove to be useful in deciding whether a task was a success or not. In its idealized implementation, the policy would have been completely automatic and so reliable, that laborious eye confirmation would not have had to be necessary.

5.6.3 Dataset improvements

The most obvious improvement to both the training and validation datasets would be to increase the demonstration amount. Also, a more careful selection process to the exhibited objects could lead to better results especially with multi-object demonstrations. If a theoretical future model would want to handle many more different objects, such as manifolds, gaskets, and different bolts to name a few, the dataset should obviously contain examples where just those items are present. In this thesis, better results were achieved when presenting goal objects with multiple instances of similar clutter or just single distracting objects. It could be argued that due to the only a single object being considered the "correct" one in single action tasks, a dataset should not contain examples with multiple goal objects like in this thesis except in the verification phase (where possible). In this model, every time multiple goal objects were present, the one closer to the goal object was demonstrated as the correct one which might also have contributed to the error rate in unmeasured ways.

Some clutter examples especially in the training dataset were probably too difficult for the used model and less cluttered tasks would probably have been more useful for training. The choice to also include multiple images of a sequential task was also not grounded in any previous observation and its impact on the validity of the model remains unverified. As single demonstrations the sequential tasks are not worse than any other demonstration with the notable exception of containing multiples of the goal objects which leads to the same classification problem mentioned before with only one option among equals being considered valid.

The current model was also trained with a dataset with a semi-random number of objects already present inside the goal box. Future models might want to consider patterning these and including multiple examples with the goal box partially covered by both the goal objects and unrelated objects. While CLIP has been demonstrated to include the capability to classify objects within objects, it is unclear whether the CLIP used in CLIPort can detect such recursive details, such as with a box containing bolts when both are unseen data. Such classifications could also be made to improve the language fluidity by allowing tasks such as "put the bolt in the correct box" or "put the bolt in the box with bolts" or "put the bolt in the bolt box" instead of requiring the user to specify the colour box after identifying. One another goal-related improvement could consider dataset examples with directional inference, such as with the inclusion of tasks such as "put the bolt in the box on the left" etc.

CLIPort was demonstrated in simulations to be capable of sorting tasks. Such behaviours might be implemented with industrial objects and could be modified to include the pick & place type task flow instead of the CLIPort demonstrated pushing. Another industry-relevant capability in CLIPort is the align task, where an object is pushed to align with a

surface element. Such tasks could be useful in robot-to-robot collaboration. Collecting real-world data for these tasks could prove to be very difficult, especially since many industrial objects can roll for long distances even with minute forces.

Lastly, one clear improvement over the current model would include multiple backgrounds and utilize more interesting goal containers over the presented boxes. The implemented framework can be quite accurate with the pick tasks, so having a stricter place position should be possible if the goal position is clearly classifiable from the image. CLIP is also likely capable of classifying human anatomy such as hands, but demonstrations were not recorded with human limbs visible. For collaboration, tasks such as following a human hand or going to a position pointed by a human finger could be useful by allowing such tasks as: "give that (specific) bolt to me" or "put it here".

6. CONCLUSIONS

This chapter gives a conclusion to the thesis by discussing the success of the thesis goals, and postulation on future improvements. A short summary is given at the end.

6.1 Discussion on Research Objectives

Chapter 1.3 presented the research objectives for the thesis and this chapter discusses their success in separate sections.

Assess the accuracy of the speech model in CLIPort nlihrc

One concern raised in the previous work considered the possibility that the implemented speech recognition tool would not properly work with different voices. Further discussion on why this research objective omitted can be found in Chapter 3.5.

Assess the prediction accuracy of CLIPort nlihrc using multiple models with different samples and amount of training epochs

A significant portion of the thesis, namely Chapter 5, considers the accuracies of the manufactured models, but only one presented model can be considered successful. Most of the developed datasets and trained models were used for issue discovery and furthering development, and comparison with partially trained models with low epochs was considered banal since longer trained models performed better as one would expect in machine learning in general. One noteworthy factor in the training improvement over time is that the used epoch counts were very low compared to the CLIPort simulation suggestions. No model was trained to the CLIPort suggested epoch count due to time constraints, and thus the exhibited models had likely not yet plateaued in prediction accuracy. For training loss examination, the models were perceived to train much as in the previous thesis on which this extends. This makes the similarly efficient training cost expected behaviour, since there were no changes made to the prediction pipeline. For these reasons the training loss charts were not presented in the thesis. The best-case location prediction accuracy of CLIPort was 100.0% in the validation tests, where just one portion of the overall task is considered manifesting as a single prediction. The implemented framework with

the best model can be expected to flawlessly execute a given task roughly 75.07% of the time.

Generate dataset(s) and demonstrate suitability for industrial objects

A significant portion of the thesis, namely Chapter 4, considers the structure of the utilized and assembled datasets. Function with real industrial objects was demonstrated with some example objects taken from a real engine. The used objects were selected due to their similar shape and texture. Most of the other engine assembly related objects were either too heavy or too large to properly act as goal objects with the implemented framework. Possible ways to subvert these limitations are briefly discussed in Chapter 5.6. The datasets featured similarly shaped objects of the same colour in different configurations and could successfully coax the implemented models to have a distinction between similar objects in the workspace.

Demonstrate potentially useful tasks for industrial applications using an existing solution back end

The thesis utilized the backend mainly to execute pick & place tasks whose role in industry is significant, but which are not necessarily crucial in collaborative engine assembly. The framework pick module can be utilized in handover tasks, which are very relevant for collaborative robotics, but the implemented handover position was hardcoded. These types of handover tasks and the framework as a whole can be categorized to be *Interaction*, which is an approach in collaborative robotics in general.

Research similar systems

The thesis presents 9 relevant papers and implementations albeit superficially. There were more discovered frameworks which are not presented due to their age and for the sake of brevity. Special emphasis was also placed on solutions that utilized CLIPort in some way. The research field was found to be relatively new and highly active, as many implemented solutions are newer than CLIPort. Even though CLIPort was introduced in 2020, it can be considered an old approach already in some contexts.

Compare CLIPort nlihrc results with results from similar systems

The thesis compared the implemented framework model validation results with some other solutions and found that the performance of the implemented framework was slightly worse than the numbers presented by state-of-the-art solutions. The other models were only compared with tasks that were similar to the framework validation dataset pick & place tasks.

6.2 Summmary

This thesis presents a modification to a previously done CLIPort framework that allows for collaborative tasks using industrial objects. A model for working and detecting these objects was manufactured and tested on real-world data. The thesis goes through relevant research topics and discusses the previous work. There is a large emphasis on dataset collection and makeup as well as model performance and discussion on found and postulated error sources. Supportive evidence is also provided to give insight on discovered errors and chosen improvement directions, such as with the multi-step workflow. The conclusion chapter discusses future improvements on the subject with emphasis based on model- and dataset improvements. The code used for this project is available in GitHub [45], [46], and [47].

REFERENCES

[1] Hassan, O. *Natural Language Instructions for Human Robot Collaboration*. eng. Informaatioteknologian ja viestinnän tiedekunta - Faculty of Information Technology and Communication Sciences, 2023.

[2] Shridhar, M., Manuelli, L. and Fox, D. *CLIPort: What and Where Pathways for Robotic Manipulation*. 2021. arXiv: 2109.12098 [cs.RO].

[3] *TUG Autonomous Mobile Robots*. Oct. 25, 2023. URL: https://aethon.com/products/ (visited on 10/25/2023).

[4] *Starship Technologies: Autonomous robot delivery*. Oct. 25, 2023. URL: https://starship.co/ (visited on 10/25/2023).

[5] Wang, L., Gao, R., Váncza, J., Krüger, J., Wang, X., Makris, S. and Chryssolouris, G. "Symbiotic human-robot collaborative assembly". eng. *CIRP annals* 68.2 (2019), pp. 701–726. ISSN: 0007-8506.

[6] Matheson, E., Minto, R., Zampieri, E. G. G., Faccio, M. and Rosati, G. "Human–Robot Collaboration in Manufacturing Applications: A Review". eng. *Robotics (Basel)* 8.4 (2019), pp. 100–. ISSN: 2218-6581.

[7] Li, J., Li, D., Xiong, C. and Hoi, S. "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation". *International Conference on Machine Learning*. 2022, pp. 12888–12900.

[8] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G. and Sutskever, I. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV].

[9] Li, D., Li, J. and Hoi, S. C. H. *BLIP-Diffusion: Pre-trained Subject Representation for Controllable Text-to-Image Generation and Editing*. 2023. arXiv: 2305.14720 [cs.CV].

[10] *OpenAI page for DALL-E 2*. Oct. 23, 2023. URL: https://openai.com/dall-e-2 (visited on 10/23/2023).

[11] Google. *WebRTC Voice Activity Detector*. Available: https://webrtc.org/.

[12] Silero Team. *Silero VAD: pre-trained enterprise-grade Voice Activity Detector (VAD), Number Detector and Language Classifier*. https://github.com/snakers4/silero-vad. 2021.

[13] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G. and Vesely,

K. “The Kaldi Speech Recognition Toolkit”. *IEEE Workshop on Automatic Speech Recognition and Understanding*. 2011.

[14] Alpha Cephei. *Vosk Speech Recognition Toolkit*. https://github.com/alphacep/vosk-api. 2023.

[15] Yin, W., Kann, K., Yu, M. and Schütze, H. “Comparative study of CNN and RNN for natural language processing”. *arXiv preprint arXiv:1702.01923* (2017).

[16] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. “Attention is all you need”. *Advances in neural information processing systems* 30 (2017).

[17] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. “Bert: Pre-training of deep bidirectional transformers for language understanding”. *arXiv preprint arXiv:1810.04805* (2018).

[18] Reimers, N. and Gurevych, I. “Sentence-bert: Sentence embeddings using siamese bert-networks”. *arXiv preprint arXiv:1908.10084* (2019).

[19] *CLIP github page*. Oct. 9, 2023. URL: https://github.com/openai/CLIP (visited on 10/09/2023).

[20] *CLIPort github page*. Oct. 10, 2023. URL: https://cliport.github.io/ (visited on 10/10/2023).

[21] Zheng, K., Chen, X., Jenkins, O. C. and Wang, X. E. “VLMbench: A Compositional Benchmark for Vision-and-Language Manipulation”. eng. *arXiv.org* (2022). ISSN: 2331-8422.

[22] Gong, R., Gao, X., Gao, Q., Shakiah, S., Thattai, G. and Sukhatme, G. S. “LEMMA: Learning Language-Conditioned Multi-Robot Manipulation”. eng. *IEEE robotics and automation letters* 8.10 (2023), pp. 6835–6842. ISSN: 2377-3766.

[23] Mees, O., Hermann, L., Rosete-Beas, E. and Burgard, W. B. “CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks”. eng. *IEEE robotics and automation letters* 7.3 (2022), pp. 7327–7334. ISSN: 2377-3766.

[24] Zhu, Y., Joshi, A., Stone, P. and Zhu, Y. “VIOLA: Imitation Learning for Vision-Based Manipulation with Object Proposal Priors”. eng. *arXiv.org* (2023). ISSN: 2331-8422.

[25] Toyer, S., Shah, R., Critch, A. and Russell, S. “The MAGICAL Benchmark for Robust Imitation”. eng. *arXiv.org* (2020). ISSN: 2331-8422.

[26] Bucker, A., Figueredo, L., Haddadin, S., Kapoor, A., Ma, S., Vemprala, S. and Bonatti, R. “LATTE: LAnguage Trajectory TransformEr”. eng. *2023 IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2023-. IEEE, 2023, pp. 7287–7294. ISBN: 9798350323658.

[27] *Franka Emika robot arm information page*. Oct. 10, 2023. URL: https://www.franka.de/ (visited on 10/10/2023).

[28] Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth,

D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I. and Florence, P. *PaLM-E: An Embodied Multimodal Language Model*. 2023. arXiv: 2303.03378 [cs.LG].

[29] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". eng. *arXiv.org* (2021). ISSN: 2331-8422.

[30] Sajjadi, M. S. M., Duckworth, D., Mahendran, A., Steenkiste, S. van, Pavetić, F., Lučić, M., Guibas, L. J., Greff, K. and Kipf, T. "Object Scene Representation Transformer". *NeurIPS* (2022).

[31] Aguinaldo, A., Bunker, J., Pollard, B., Shukla, A., Canedo, A., Quiros, G. and Regli, W. "RoboCat: A Category Theoretic Framework for Robotic Interoperability Using Goal-Oriented Programming". eng. *IEEE transactions on automation science and engineering* 19.3 (2022), pp. 1–9. ISSN: 1545-5955.

[32] Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Li, F.-F., Anandkumar, A., Zhu, Y. and Fan, L. "VIMA: General Robot Manipulation with Multimodal Prompts". eng. *arXiv.org* (2023). ISSN: 2331-8422.

[33] Kapelyukh, I., Vosylius, V. and Johns, E. "DALL-E-Bot: Introducing Web-Scale Diffusion Models to Robotics". eng. *IEEE robotics and automation letters* 8.7 (2023), pp. 3956–3963. ISSN: 2377-3766.

[34] *Nvidia's marketing page for the RTX 3080 GPU*. Sept. 28, 2023. URL: https://www.nvidia.com/us-en/geforce/graphics-cards/30-series/rtx-3080-3080ti/ (visited on 09/28/2023).

[35] *Technical specifications for the AMD Ryzen 7 5845*. Oct. 2, 2023. URL: https://www.amd.com/en/products/apu/amd-ryzen-7-pro-5845 (visited on 10/02/2023).

[36] *Nvidia's marketing page for the RTX 4090 GPU*. Sept. 28, 2023. URL: https://www.nvidia.com/us-en/geforce/graphics-cards/40-series/rtx-4090/ (visited on 09/28/2023).

[37] *Technical specifications for the Intel Core i9-13900k*. Oct. 2, 2023. URL: https://www.intel.com/content/www/us/en/products/sku/230496/intel-core-i913900k-processor-36m-cache-up-to-5-80-ghz/specifications.html (visited on 10/02/2023).

[38] *Technical specifications for the Intel Core i7-4770*. Oct. 2, 2023. URL: https://ark.intel.com/content/www/us/en/ark/products/75122/intel-core-i74770-processor-8m-cache-up-to-3-90-ghz.html (visited on 10/02/2023).

[39] *Jabra's support page for the Jabra EVOLVE 20 headset family*. Oct. 2, 2023. URL: https://www.jabra.com/supportpages/jabra-evolve-20#/#4999-823-109 (visited on 10/02/2023).

[40] *Intel's marketing page for the D435 camera*. Sept. 28, 2023. URL: https://www.intelrealsense.com/depth-camera-d435/ (visited on 09/28/2023).

[41] *Intel's marketing page for the D435i camera*. Sept. 28, 2023. URL: https://www.intelrealsense.com/depth-camera-d435i/ (visited on 09/28/2023).

[42] GmbH, F. E. *franka hw*. Available: https://frankaemika.github.io/docs/franka_ros.html#franka-hw.

[43] Turc, I., Chang, M.-W., Lee, K. and Toutanova, K. *Well-Read Students Learn Better: On the Importance of Pre-training Compact Models*. 2019. arXiv: 1908.08962 [cs.CL].

[44] Robotics, P. *MoveIt Motion Planning Framework for ROS*. Available: https://github.com/ros-planning/moveit.

[45] *GitHub page for the edited CLIPort modification used in the thesis*. Oct. 31, 2023. URL: https://github.com/petajam117/cliport (visited on 10/31/2023).

[46] *GitHub page for the edited data collection tool used in the thesis*. Oct. 31, 2023. URL: https://github.com/petajam117/cliport_label (visited on 10/31/2023).

[47] *GitHub page for the edited language processing tool used in the thesis*. Oct. 31, 2023. URL: https://github.com/petajam117/python-nlihrc (visited on 10/31/2023).