UiA University of Agder

Master

Utilizing Reinforcement Learning and Computer Vision in a Pick-And-Place Operation for Sorting Objects in Motion

Building an autonomous small scale sorting facility using a robotic manipulator with perception from a RGB camera.

KRISTOFFER SAND
TRYGVE ANDRE OLSØY SOLBERG

SUPERVISOR
Assoc. Prof. Per Arne Andersen, University of Agder
Res. Fell. Emil Mühlbradt Sveen, University of Agder

University of Agder, 2023
Faculty of Engineering and Science
Department of Engineering and Sciences

Obligatorisk gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

1.	Vi erklærer herved at vår besvarelse er vårt eget arbeid, og at vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.	Ja
2.	**Vi erklærer videre at denne besvarelsen:** • Ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • Ikke refererer til andres arbeid uten at det er oppgitt. • Ikke refererer til eget tidligere arbeid uten at det er oppgitt. • Har alle referansene oppgitt i litteraturlisten. • Ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	Ja
3.	Vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§ 31.	Ja
4.	Vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert.	Ja
5.	Vi er kjent med at Universitetet i Agder vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens retningslinjer for behandling av saker om fusk.	Ja
6.	Vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider.	Ja
7.	Vi har i flertall blitt enige om at innsatsen innad i gruppen er merkbart forskjellig og ønsker dermed å vurderes individuelt. Ordinært vurderes alle deltakere i prosjektet samlet.	Nei

Publiseringsavtale

Fullmakt til elektronisk publisering av oppgaven Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven. §2).
Oppgaver som er unntatt offentlighet eller taushetsbelagt/konfidensiell vil ikke bli publisert.

Vi gir herved Universitetet i Agder en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	Ja
Er oppgaven båndlagt (konfidensiell)?	Nei
Er oppgaven unntatt offentlighet?	Nei
Er utviklet programvare konfidensiell?	Ja

Abstract

This master's thesis studies the implementation of advanced machine learning (ML) techniques in industrial automation systems, focusing on applying machine learning to enable and evolve autonomous sorting capabilities in robotic manipulators. In particular, Inverse Kinematics (IK) and Reinforcement Learning (RL) are investigated as methods for controlling a UR10e robotic arm for pick-and-place of moving objects on a conveyor belt within a small-scale sorting facility. A camera-based computer vision system applying YOLOv8 is used for real-time object detection and instance segmentation. Perception data is utilized to ascertain optimal grip points, specifically through an implemented algorithm that outputs optimal grip position, angle, and width. As the implemented system includes testing and evaluation on a physical system, the intricacies of hardware control, specifically the reverse engineering of an OnRobot RG6 gripper is elaborated as part of this study.

The system is implemented on the Robotic Operating System (ROS), and its design is in particular driven by high modularity and scalability in mind. The camera-based vision system serves as the primary input, while the robot control is the output. The implemented system design allows for the evaluation of motion control employing both IK and RL. Computation of IK is conducted via MoveIt2, while the RL model is trained and computed in NVIDIA Isaac Sim.

The high-level control of the robotic manipulator was accomplished with use of Proximal Policy Optimization (PPO). The main result of the research is a novel reward function for the pick-and-place operation that takes into account distance and orientation from the target object. In addition, the provided system administers task control by independently initializing pick-and-place operation phases for each environment. The findings demonstrate that PPO significantly enhanced the velocity, and adaptability of industrial automation compared to motion control with MoveIt. The research shows that accurate control of the robot arm can be reached by training the PPO Model purely by applying and training in a digital twin.

The application is showcased in a demo video: Utilizing Reinforcement Learning and Computer Vision for a Pick-And-Place Operation

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BOM	Bill Of Materials
CAD	Computer-Aided Design
cobot	Collaborative Robot
DOF	Degrees of Freedom
FOV	Field of View
GUI	Graphical User Interface
HMI	Human Machine Interface
IC	Integrated Circuit
ICT	Information and Communications Technology
IDE	Integrated Development Environment
IK	Inverse Kinematics
MIL	Mechatronics Innovation Lab
OOP	Object Oriented Programming
OS	Operating System
PID	Proportional Integral Derivative
PLA	PolyLactic Acid
PPO	Proximal Policy Optimization
RAM	Random Access Memory
RL	Reinforcement Learning
ROS	Robot Operating System
RT	Real-Time
SAC	Soft Actor-Critic
SBC	Single Board Computer
TF	Transformation
UiA	Universitet i Agder (University of Agder)
UDP	User Datagram Protocol
YOLO	You Only Look Once
STL	Stereolithography
URDF	Unified Robot Description Format
USD	Universal Scene Description

Preface

This thesis is presented in partial fulfillment of the Master of Science in Mechatronics and Information and Communications Technology (ICT) degree requirements at the University of Agder. The project is carried out between early January and mid-June and is the culminating component of the Master's program in the form of MAS 500 and IKT 590 Master's thesis with 30 ECTS credits. Hence, the outline of the report is explained accordingly in more general terms for then in depth and in more detail, by the reason of a interdisciplinary master project.

The initiative are developed in collaboration with UiA, StellAi, MiL and DeepCobot. We would like to thank these partners for their contributions in this endeavor as well as their advice, equipment and motivation. StellAi will use this project's results for further research and development of autonomous sorting operation, so a thorough explanation and documentation of the project's development stages is emphasized. During the project's execution, a larger portion of code was developed and is confidential. However, for academic use, specific parts will be shared based on request. Complete source codes for implementation can be found at MotionControl, RLPickAndPlace and ComputerVision.

Finally, we would like to express our gratitude to our supervisors Emil Mühlbradt Sveen and Dr.Per-Arne Andersen for their courteous and useful assistance. It has been motivating to collaborate with knowledgeable individuals like you on our Master's thesis. Sincere gratitude.

The pair of Kristoffer Sand and Trygve Solberg
University of Agder, Grimstad campus
The fifth of June

Chapter 1

Introduction

This master's thesis is a result of an interdisciplinary research applying the disciplines of mechatronics and Information and Communication Technology, in particular, applying Robotics and machine learning (ML). A main goal of the study has been to implement an efficient and adaptive small-scale sorting facility driven by the challenges presented in the Problem Statement Section 1.3. Stellai AS (StellAi) has been a main stakeholder of the implementation. StellAi is a startup with a desire to make the recycling industry more sustainable by evolving on state of the art technology. The implementation has been supported by Mechatronics Innovation Lab (MIL) at the University of Agder (UiA), that has provided necessary resources and equipment needed for the implementation.

For enabling an industrial set up and supporting realistic real life testing, a conveyor belt is set up for transporting unsorted objects from which the robot is able to pick and place the various objects into the correct baskets. In the interest of exploring and pushing the capabilities of applying ML algorithms, the application has been restrained to only using one RGB camera, and it is employed on a robotic manipulator for executing the Pick-And-Place operation. For developing a fully automated sorting facility, the overall process has been divided into three phases:

1. Computer Vision with object detection for classification and localization of objects placed with randomized pose at the beginning of the moving conveyor belt. In addition to multi-object tracking with an algorithm to estimate optimal grip position, angle and width.

2. Object handling, the objects are handled by a collaborative robot where one solution uses IK to calculate joint positions throughout the planned trajectory. The other solution utilizes a learned RL policy to move each joint position based on target pose. The robot is picking outside the Field of View (FoV) of the RGB camera and only the world coordinates of the target object is known through estimation by a computer vision pipeline.

3. A virtual replication of the environment as a digital twin in order to trace, simulate and calculate the working of the Robot before moving.

In the quest for developing autonomous systems, a particular challenge is the need to maintain functionality and reliability, while minimizing the dependency on traditional sensor arrays. The focus is to create a sensor-minimalistic system that leverages a single RGB camera for visual perception, eschewing the typical array of standard industry sensors. Such an approach underpins the potential of computer vision techniques in transcending the boundaries of conventional sensor-based systems.

Object detection has been instrumental in enabling real-time, high-accuracy perception in various applications, making it a critical component of proposed sensor-minimalistic systems.

By using sophisticated algorithms, object detection is capable of identifying and localizing objects within a complex environment, thereby enabling the system to interact effectively with its surroundings. This accurate perception forms the foundation of an autonomous system's capability to execute various tasks.

An important factor for the success of the implementation is that developed mechanisms work in a dynamic environment where the localization of objects is random and constantly changing. This is especially pertinent in the context of pick-and-place operations, where the system must effectively identify, localize, and manipulate objects despite the absence of a pre-determined object placement. Consequently, the object detection algorithm must be robust and adaptable enough to handle these environmental variations, providing precise localization and classification of objects in real time. The versatility of the environment mandates a system that is adaptable and resilient, thus, underscoring the significance of advanced computer vision techniques in enabling sensor-minimalistic systems to function effectively in complex, real-world scenarios.

The design, implementation, and assessment of the suggested system according to the main requirements indicated above for the three main phases are main content of this thesis. The emphasize has been on how we have applied machine-learning techniques to significantly improve the sorting procedure. The thesis also include some elaboration of upcoming projects and potential enhancements that could be made to the implemented system to increase performance.

1.1 Background and Motivation

Intelligent solutions are required to automate processes in the waste industry. This is in particular because waste come in various shapes. Intelligent systems that can automatically handle and sort waste has a great value potential as the industry handles large quantities of waste. The newly established StellAI has identified two significant problem as of today that have been investigated in this research. The first problem is to sort out abnormalities in a safe way. The industry has major problems with objects such as batteries entering processes which can lead to explosions and dangerous situations for both people and equipment. Problem number two is to develop a mini sorting facility that can sort waste at material consumers site, in order to avoid having to transport the waste to recycling stations. This project will provide insight into how state-of-the-art technology in mechatronics, ICT and machine learning can help to solve these problems.

Innovation in the Recycling Industry with StellAi

StellAi aims to make the recycling industry more sustainable by evolving on state of the art technology. The company is a startup, with personnel that have background from the recycling industry and extensive expertise in waste and circular economy. To be able to deliver an advanced technological system, the collaboration between the University of Agder and Mechatronics Innovation Lab was desired. The objective is to reach an advanced autonomous pick-and-place system that can substantially improve sorting processes across a variety of industries without sacrificing adaptability or usability. By setting up a mini sorting facility for the consumer, it ensures purer waste prepared for industrial recycling facilities and contribute to a more circular economy. The material consumers pays large amount of money to deliver unsorted waste, and it can further affect environmental tax. In current recycling sorting facilities, material-waste is transported on a conveyor belt, and multiple sensors are used to detect the objects and separate the waste. However, these facilities must manage a vast assortment of objects that are randomly positioned. StellAi want to apply ML

algorithms as its a promising solution for managing the massive amounts of data generated by these systems, allowing for the efficient recognition and classification of refuse materials. Figure 1.1 is visualizing the design of the entire sorting facility, where this project is meant to be employed. The designed process line is explained step by step in Appendix F

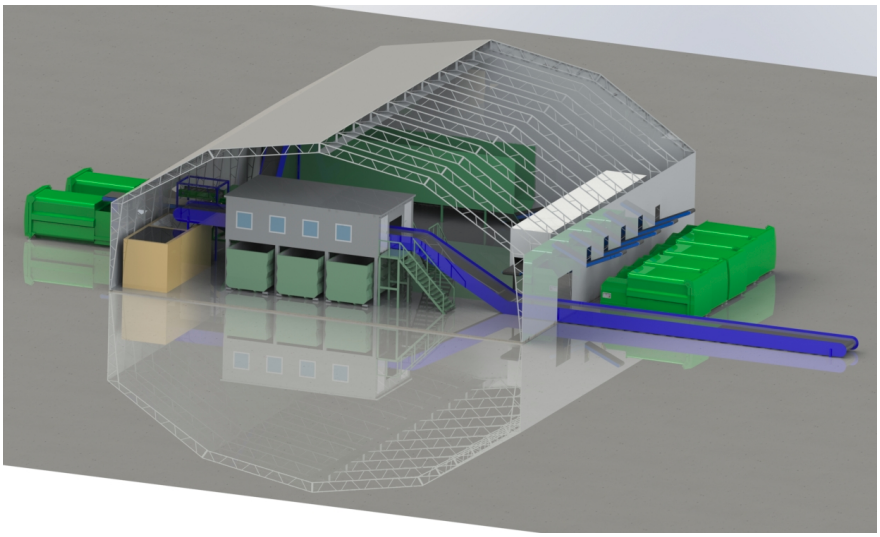


Figure 1.1: Design of Sorting Facility

1.2 Project Scope

This section describes the mandatory requirements specification.

- Perception applying only one RGB camera.
- Pick-And-Place objects at a minimum velocity of 10 cm/s.
- Scalability of objects enabling to add new objects to the detection and pick-and-place pipeline.
- Pick objects of diverse geometries, with dimensions ranging from 3 to 15 cm in length, breadth, and height. The object constraints are defined by robotic gripper limitations of the implemented system.

The main objectives of this research origins in particular from requirements put forward by StellAi that aim to build novel solutions for waste management evolving on state of the art technologies. The main objectives are:

- Develop a completely autonomous sorting facility that utilizes ML algorithms and only a single RGB camera for perception, instead of the current solutions that applies a lot of sensors to detect differences in materials 2.1.
- Develop a robust computer vision algorithm capable of detecting and identifying various objects based on RGB images, as well as the implementation of real-time image processing techniques to optimize performance and reduce latency.
- Scalability, since the industry handles continuous change of materials, the objects, which are subject to continuous variation and change, it is imperative to establish a modular pipeline that is adaptable to new type of objects and waste. The pipeline needs to facilitate the incorporation of new objects and ensure the efficient training

of subsequent models for detecting new objects. Additionally, the camera should be strategically positioned and configured to capture all objects with adequate resolution, image quality and distortion.

- Have the robot executing pick-and-place operations on objects moving at a minimum velocity of 10 cm/s, necessitating the development and implementation of efficient motion planning algorithms capable of achieving this velocity without compromising accuracy and precision.

1.3 Problem Statement

This research focuses on the development of an enhanced control system for a UR10e robotic arm in a pick-and-place setting. The work particularly emphasizes the application of ML techniques, including instance segmentation and reinforcement learning, to improve the manipulator's performance in motion control. The following are the defined problem statements:

- How can data derived from instance segmentation of target objects, captured using a single RGB camera, be effectively utilized for object recognition and tracking in real-world scenarios?
- What are the advantages of applying reinforcement learning for motion control in comparison to using inverse kinematics, specifically in the context of objects traveling on a conveyor belt system? How might advantages of Reinforcement Learning manifest in terms of efficiency and adaptability
- To what extent can a Proximal Policy Optimization (PPO) algorithm provide consistent and precise control of a UR10e robotic arm where high speed and accuracy are essential? What factors might influence this consistency and predictability, and how can they be optimized?
- How can reward engineering enhance the modularity in training robots in a simulation environment? What specific techniques can be used to effectively design reward systems that support modular robot learning?
- How can the redundancy control of an UR10 robot arm be improved by implementing a reinforcement learning model trained in a simulated digital environment that mirrors the real-world operational context? What are the potential limitations and how can they be addressed?

1.4 Design Science

Adopting a Design Science approach, the project pushes the limits of a full-stack software application, prioritizing functional efficacy over exhaustive optimization of individual processes. The focus is on creating a practical solution that functions optimally within the physical implementation. The evaluation method for the technology, which considers the level of performance required for industrial applications, incorporates empirical testing and case studies. The first case study involves a robotic manipulator sorting out waste abnormalities on a moving conveyor belt, and the second one targets the complete sorting of all transported waste fractions.

1.4.1 Main Contributions

This thesis presents an innovative solution for a key challenge in industrial automation: creating a generic interface for fully autonomous pick-and-place operations on a moving conveyor belt using a robotic manipulator, specifically a UR10e. The solution encompasses two

key methods of controlling the robot: IK and RL, each offering its unique benefits. The IK method mathematically calculates the trajectory path to a target pose then moves, while the RL method continuously update a learned control policy based on a target pose.

A key aspect of the presented solution is the utilization of Computer Vision, to give perception to the robot through deep convolutional neural networks that outputs the instance segmentation and object detection in real time. This perception is used for both the Inverse Kinematic control of the robot and the simulation to real approach with RL for a smart dynamic solution. The detection data is used with a multi-object tracking algorithm that passes new objects to a optimal grip point algorithm.

The project presents an implementation of a optimal grip point algorithm that processes the instance segmentation data and outputs target position in the physical 3D space from 2D RGB data, and outputs optimal grip angle and grip width.

To be able to remote control a OnRobot RG6 gripper it was reversed engineered by probing to the fieldbus running on RS485 communication. While the distributor offers a separate computing device for remote control of their devices, it was required to develop a custom communication interface with the gripper. The communication interface with the gripper is built as a python package implemented on top of the ROS2 communication system. The communication node interface is implemented by a Modbus RTU Application Protocol with defined communication properties set based on the reverse engineering results. The communication interface is modular and can be used towards other grippers from the same distributor or others.

This report demonstrates high accuracy control of the robot manipulator through Reinforcement Learning, where it is trained in simulation with up to 4000 parallel environments running simultaneously.

The thesis proposes a novel reward function for pick-and-place with reward engineering, that factors distance and orientation from target pose to calculate reward. The reward function is further used to handle task control by initializing the different phases of the pick-and-place operation for each environment independently.

1.4.2 Equipment

There are three main components that have been arranged for this project, acquired by StellAi and MiL.

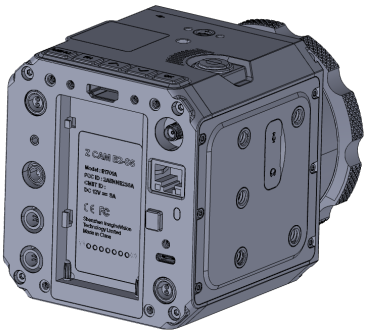


Figure 1.2: RGB Camera

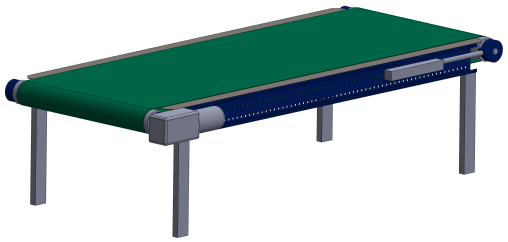


Figure 1.3: Conveyor Belt

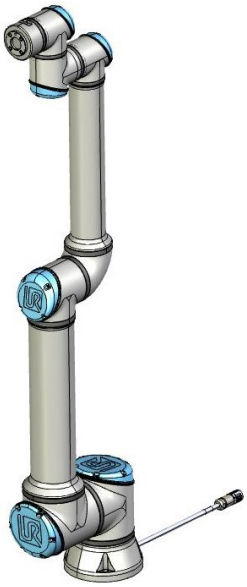


Figure 1.4: Collaborative Robot

1.5 Report Outline

- **Chapter 2** outlines the supporting technology and theoretical underpinnings needed to realize the project.
- **Chapter 3** explicates the project's foundational real-world use case before providing an overview of the system design and the components chosen to realize the design.
- **Chapter 4** utilizing data from instance segmentation of target objects, captured using a single RGB camera and calculating optimal grip point.
- **Chapter 5** details the construction of the autonomous sorting facility, both in its digital and physical manifestations. This chapter explores the process of building the facility and the solutions implemented.
- **Chapter 6** delves into the Reinforcement Learning (RL) aspect of the project. It elaborates the RL methodologies employed, the training of the model, and its integration within the larger system.
- **Chapter 7** presents the experiments conducted and the results obtained. This chapter provides a thorough analysis of the logged data.
- **Chapter 8** engages in a comprehensive discussion of the project's outcomes and delineates potential avenues for future work.
- **Chapter 9** concludes the report, summarizing the key insights and contributions of the study.

Chapter 2

State of the Art and Enabling Technologies

This thesis is situated at the intersection of numerous technological disciplines, each offering its unique methodologies, approaches, and perspectives that contribute to the development conducted in this research. Given the breadth of these fields, it is neither feasible nor necessary to explore each one in exhaustive detail. However, a foundational understanding of the key concepts and developments in these domains is crucial for comprehending the various components of the project and the ways in which they interrelate.

This chapter will therefore provide an overview of the state of the art and the enabling technologies that underpin our work. We will focus on those technologies that play a central role in the project, outlining their core principles, the current extent of their development, and their applications in relevant contexts. This will provide readers with the necessary background knowledge to fully grasp the project's goals, methodologies, and findings.

The chapter will cover areas such as Artificial Intelligence, specifically ML and RL; Robotics, particularly in the field of Manipulation and Autonomous Operations; Computer Vision, focusing on Object Detection and Instance Segmentation; and aspects of Automation and Control Systems relevant to the Industrial Environment.

By highlighting the state-of-the-art techniques and tools in these areas, we aim to showcase the technological landscape that our project inhabits, thereby emphasizing the novelty and significance of our contributions to these exciting fields of research.

2.1 Perception Used in the Recycling Industry

Various perception technologies are implemented in the modern recycling industry for separation of waste materials. Infrared (IR) sensors, near-infrared (NIR) spectroscopy, laser-induced breakdown spectroscopy (LIBS), X-ray fluorescence (XRF), and hyperspectral imaging predominate among these technologies. These technologies are widely employed due to their ability to detect and distinguish between diverse materials, including plastics, metals, and paper, based on their unique physical or chemical properties [7].

- Infrared (IR) sensors are used to detect and distinguish between different varieties of plastic based on their unique infrared absorption characteristics.
- Near-infrared (NIR) spectroscopy: is used to identify and separate various materials, such as plastics and paper, based on their distinct spectral signatures in the NIR range.

- Laser-induced breakdown spectroscopy (LIBS): is a rapid, non-contact technique for analyzing and sorting metals based on their distinct elemental compositions.
- X-ray fluorescence (XRF): is a non-destructive analytical technique that can be used to identify and classify materials based on their elemental composition, especially metals and heavy plastics.
- Hyperspectral imaging combines imaging and spectroscopy to provide high-resolution spatial and spectral information that can be used to identify and classify different materials based on their unique spectral signatures.

2.2 Computer Vision

Computer Vision is a field of artificial intelligence that trains computers to interpret and understand the visual world. Through acquiring, processing, analyzing, and understanding digital images or sequences of images, computer vision systems can extract information from the real world to make decisions or predictions.

Computer vision encompasses a wide array of tasks including object detection, image recognition, video tracking, 3D reconstruction, and semantic segmentation among others. The ultimate goal of computer vision is to automate tasks that the human visual system can do with ease and efficiency. However, the complexity of perception and the vast amount of information available in images make computer vision a challenging task in the realm of AI.

2.2.1 Convolutional Neural Network

A successful tool in computer vision tasks is the Convolutional Neural Network (CNN), a type of deep learning model particularly tailored for processing structured grid data such as images. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from the input data.

CNNs are characterized by their unique architecture, which is designed to take advantage of the 2D structure of an input image. This architecture is defined by three main types of layers: convolutional layers, pooling layers, and fully connected layers. The convolutional layer applies a series of filters to the input data to create a feature map that represents the input data. The pooling layer reduces the dimensionality of the data, which helps to decrease the computational complexity of the model. The fully connected layer is a traditional multi-layer perceptron that uses a softmax activation function in the output layer to output a distribution of probabilities representing the prediction for each class.

CNNs have shown exceptional performance in numerous computer vision tasks, such as image and video classification, object detection [31], and semantic segmentation[12]. One significant advantage of CNNs is their ability to automatically learn and generalize features from the input data, reducing the need for manual feature extraction. This is achieved by using small, local receptive fields in the initial layers of the network that partially overlap, allowing the model to recognize patterns with a high degree of translation invariance.

2.2.2 Object Detection for Real Time Processing

Real-time object detection is a fundamental task in computer vision, aiming to identify and classify objects in digital images and videos while maintaining a processing speed that allows for immediate interpretation of the visual data. This ability to process visual data in real-time enables numerous applications, such as autonomous vehicles, surveillance systems,

video analytics, and interactive systems, where immediate response is crucial [29].

Object detection is commonly achieved by placing bounding boxes around the detected objects and classifying them into different categories. However, doing this in real-time poses challenges due to the computational complexity of the task and the need for immediate processing. Therefore, the key requirements for real-time object detection are speed and accuracy, which have been a trade-off in the development of object detection systems.

2.2.3 Object Detection Metrics

Object detection is a fundamental component of computer vision, providing the technological foundation for diverse applications such as autonomous vehicles, security surveillance, image retrieval systems, and facial recognition. It entails identifying objects within images or video frames, typically by boxing them and classifying them into various categories. Given the increasing prevalence of object detection systems, it is becoming increasingly essential to devise effective performance evaluation metrics. These metrics provide crucial feedback for enhancing models during the development phase and offer insights into their relative performance.

Union over Intersection (IoU): In object detection tasks, Intersection over Union is one of the most frequently employed metrics. It measures the overlap between the predicted bounding box (detection) and the actual bounding box. The IoU is the ratio between the intersection and union areas of these two boxes. A value of 1 implies a perfect overlap, while a value of 0 indicates that there is no overlap at all. The IoU between a predicted bounding box B_p and the ground truth bounding box B_{gt} is calculated as:

$$\text{IoU}(B_p, B_{gt}) = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}} \tag{2.1}$$

where $\cap$ represents the intersection of two boxes and $\cup$ denotes the union of them.

Precision and Recall: Precision and recall are two additional commonly used metrics for object detection. Precision is the ratio of correctly recognized objects (true positives) to all detected objects, including those that were incorrectly identified (false positives). Recall, on the other hand, is the ratio of correctly detected objects to all actual objects in the image, including those that were not detected (false negatives). High precision denotes a low false-positive rate, while high recall denotes a low false-negative rate. Precision (P) and recall (R) are defined as follows:

$$P = \frac{TP}{TP + FP} \tag{2.2}$$

$$R = \frac{TP}{TP + FN} \tag{2.3}$$

where TP represents the number of true positives, FP the number of false positives, and FN the number of false negatives.

Mean Average Precision (mAP): Mean Average Precision, commonly abbreviated as mAP, is a common metric for evaluating object detection models. It is especially beneficial for managing multiple object classes. mAP computes the mean maximum precision value at various recall levels for all classes. High mAP values indicate improved overall model performance, though the precise calculation varies depending on the context or convention.

Trade-Off and Considerations: Precision and recall frequently manifest a trade-off relationship in practice. Attempting to improve precision typically decreases recall, and vice versa. Because increasing precision seeks to reduce false positives, which may inadvertently lead to an increase in false negatives and a decrease in recall. Consequently, it is essential to choose a threshold that balances these two metrics based on the specific application requirements.To calculate mAP, one first needs to compute the Precision-Recall curve for each class by sorting the detection from the highest to the lowest confidence. Then, compute the Average Precision (AP) for each class:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{\text{interp}}(r) \tag{2.4}$$

where $p_{\text{interp}}(r)$ is the maximum precision for recall values greater than r.
Finally, mAP is calculated as the mean AP over all classes:

$$mAP = \frac{1}{C} \sum_{i=1}^{C} AP_i \tag{2.5}$$

where C is the number of classes and AP_i is the AP of the ith class.

The choice of IoU threshold for determining a successful detection is a further crucial consideration when employing these metrics. A stricter threshold (e.g., IoU > 0.7) ensures detections closely align with the ground truth, but may penalize near-correct detections, which may hinder model learning. based

2.2.4 Object Detection with YOLO Algorithm

The "You Only Look Once" (YOLO) algorithm is an influential paradigm in the field of real-time object detection and instance segmentation. This algorithm offers a comprehensive solution to object detection tasks by processing an image in a single stage, detecting objects, and predicting their classes simultaneously[24].

The term "YOLO" underscores the algorithm's capability to achieve its goal in a single examination of the image. This design offers considerable efficiency, enabling YOLO to operate in real-time and even on lightweight hardware. The YOLO algorithm has undergone several iterations, each contributing to its evolution and leading to its current state-of-the-art performance on benchmarks as on the COCO dataset [27].

The initial YOLO model was built upon a Convolutional Neural Network (CNN) backbone with 24 convolutional layers, which processes images to extract high-level features. These features then flow through a series of transformations – known as the 'neck' – before reaching the 'head' of the model, where predictions are generated.[24]

As the YOLO model evolved through its eight main iterations, from the original YOLO to the most recent YOLOv8, its architecture has continually been refined. The model architecture is described in Figure 2.1. The backbone of YOLOv8 is built upon CSPDarknet53[2], an enhancement of the Darknet53 architecture used in YOLOv3[23]. CSPDarknet53 incorporates a Cross Stage Partial Network (CSPNet)[30], which improves information flow across layers, leading to better performance.

The Darknet53 and CSPNet structures form integral parts of YOLO's architecture, enhancing the model's ability to extract useful features from images and thereby improving its detection capabilities.

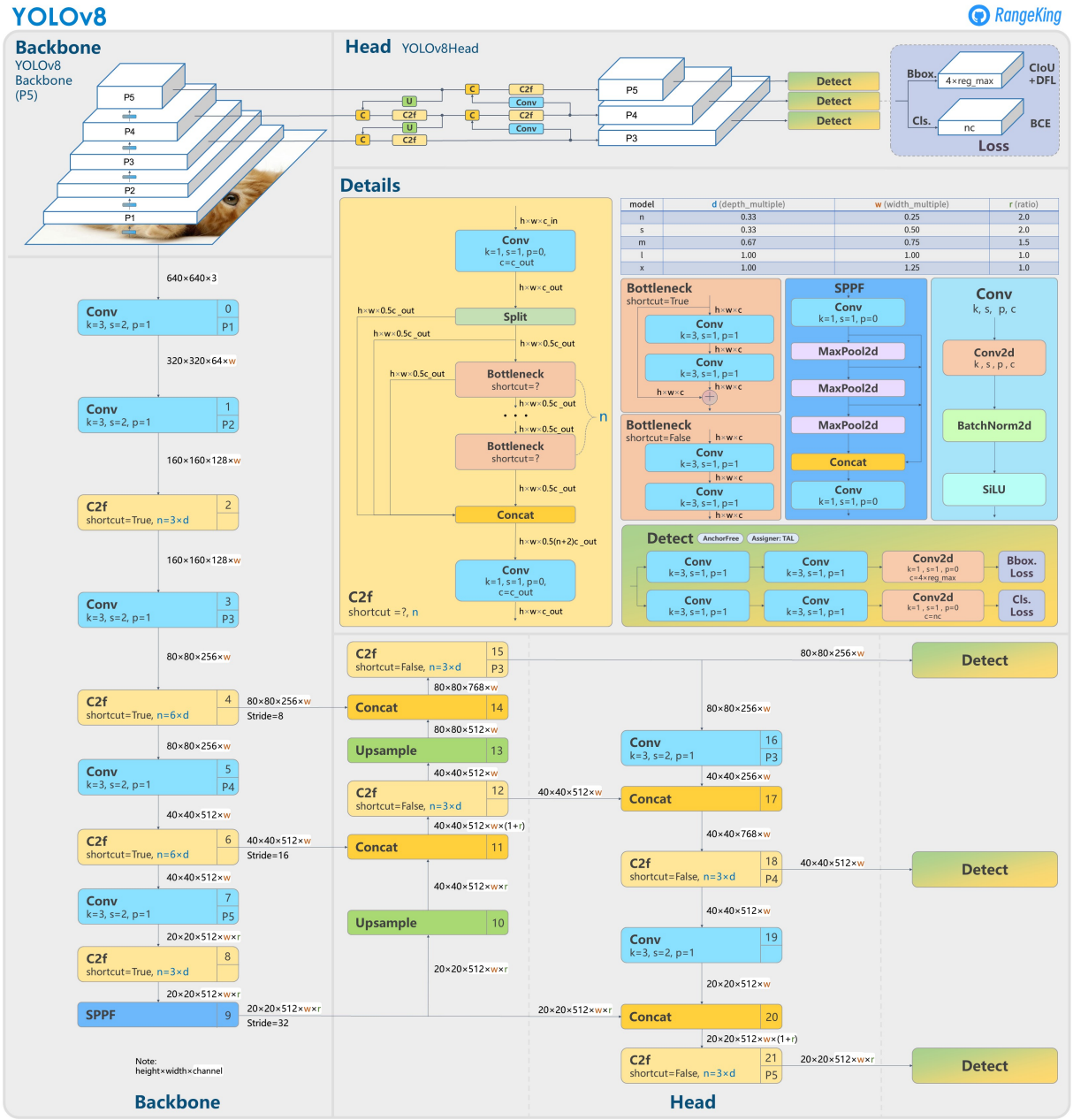


Figure 2.1: YOLOv8 model architecture: Backbone is an continuous development of the CSPDarknet53 [28]

The development of YOLO transitioned from the Darknet framework to the PyTorch framework after YOLOv4, demonstrating the model's adaptability to different programming environments and computational backends. The motivation for transition is the dominant popularity of pyTorch implementations within the research community.

One significant departure in YOLOv8 from its predecessors is its shift to an anchor-free detection model, in contrast to the anchor-based models used in earlier iterations. Anchor-based models define a set number of predefined anchor boxes, dividing the image into grids and assigning a class and confidence score for each predefined anchor box. The ultimate detection output is derived from all the predictions associated with these anchor boxes.

However, the anchor-free model used in YOLOv8 does not limit the number of objects that can be detected within a single grid cell based on predefined anchor boxes. Instead, these models generate predictions based on the features at each spatial location in the feature map.

This approach allows the model to potentially generate multiple predictions for objects of varying shapes and sizes within a single grid cell, offering a significant advantage in certain scenarios.

The YOLO algorithm represents a significant step forward in object detection, providing a comprehensive, efficient, and adaptable solution to this complex task. Its design choices, such as the shift to anchor-free detection in YOLOv8, demonstrate the continual evolution and improvement of the model, contributing to its status as a leading solution in real-time object detection and instance segmentation.

The YOLOv8 model's training regime incorporates techniques such as Multi-Scale Object Detection using a feature pyramid network, which enables the model to detect objects of varying sizes and scales within an image. Additionally, the Mosaic data augmentation technique, an enhancement of the CutMix method, is employed. Mosaic takes four images, resizes them, stitches them together, and finally selects a random cutout of the stitched images to form the final Mosaic image. These techniques collectively contribute to the model's ability to accurately predict and localize objects irrespective of their location within the frame and their size.

2.2.5 BYTETrack

BYTETrack is a multi-object tracking (MOT) model designed to estimate bounding boxes and identities of objects in videos, improving on prior work by better handling objects with low detection scores, including occluded objects [35].

BYTETrack uses a method called tracking-by-detection, which combines object detection with tracking over multiple frames of a video. The model uses a technique known as BYTE that takes into account all detection boxes, both high and low scoring, to create a more effective tracking system.

The BYTETrack model has demonstrated state-of-the-art performance on several tracking benchmarks, including MOT17, MOT20, HiEve, and BDD100K. It's also noted for its efficiency, with a running speed of 30 FPS on a single V100 GPU.

While BYTETrack is designed with a specific focus on multi-object tracking in videos, it's suitable for use cases that require tracking multiple objects over time, such as surveillance or autonomous vehicles.

The novelty of BYTETrack lies in its unique approach to handling low-confidence detection boxes, which are typically eliminated in other models. By treating every detection box as important, it can recover true objects that might otherwise be missed, leading to improvements in tracking accuracy.

BYTETrack uses the YOLOX detector to obtain detection boxes and associates them with its proposed BYTE method. YOLOX is a high-performance detector from the YOLO family, indicating that these models can be combined to enhance performance in certain applications.

2.2.6 Roboflow

Roboflow is a versatile technology platform designed to assist users in creating, managing, and deploying computer vision datasets. With Roboflow, users can convert raw images into

a usable dataset for ML applications. Below is an overview of how Roboflow operates:

1. **Dataset Creation:** Users upload raw images to the Roboflow platform to start creating their dataset. The platform supports a broad range of data formats.

2. **Annotation:** Roboflow offers tools for manually labeling and annotating images, a critical step for supervised ML algorithms. Users can draw bounding boxes, polygons, and points, or add semantic segmentation to their images.

3. **Preprocessing and Augmentation:** The platform provides robust preprocessing and augmentation options. Preprocessing steps include resizing, grayscaling, or normalizing images. Augmentation techniques like random cropping, rotation, flipping, brightness adjustments, and noise addition can expand the dataset and improve the model's ability to generalize.

4. **Dataset Versioning:** Roboflow allows users to manage different versions of their dataset. Each time a user makes a change, such as adding new images or adjusting annotations, Roboflow saves a new version. This feature is vital for tracking dataset iterations and understanding the impact of data changes on model performance.

5. **Exporting and Integration:** Users can export their prepared datasets in various formats compatible with popular ML frameworks like TensorFlow, PyTorch, and Fast.ai. Roboflow also integrates with many popular training environments.

6. **Model Training and Deployment:** Some tiers of Roboflow even allow users to train their computer vision models directly within the platform, and then deploy these models as API endpoints.

7. **Collaboration:** Roboflow supports team collaboration, allowing multiple users to work on the same dataset simultaneously.

Roboflow is a comprehensive platform for dataset creation, specifically tailored for computer vision applications. It streamlines the process of collecting, annotating, preprocessing, augmenting, managing, and deploying datasets, providing an all-in-one solution for users working on ML projects. The interface is illustrated in figure 2.2.

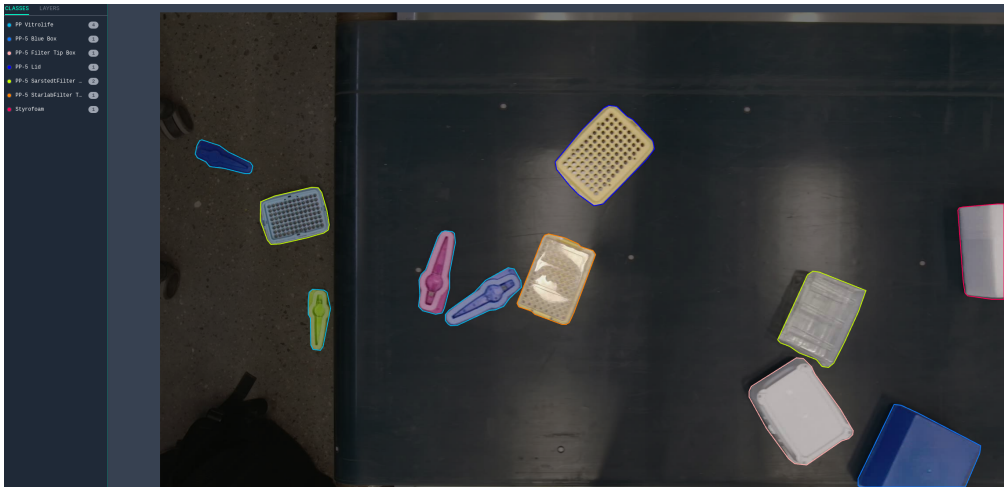


Figure 2.2: Roboflow labeling interface

2.3 Robot Operating System

Robot Operating System (ROS) is an open source framework utilized for robotics research and industrial application where real-time systems and communication with different components are essential. ROS provides a unified and distributed framework that supports a wide range of hardware and software configurations. It includes a communication infrastructure that enables different software modules (nodes) to exchange data through topics, services, and actions, thus supporting modularity and availability on information of each components. There are two main frameworks developed as of today, ROS 1 and ROS 2, but there many different versions within these types of framework.

ROS 1 uses a master to control the network processes (nodes) as communication infrastructure. ROS 1 has limitations, particularly when it comes to performance, communication security, and support for real-time systems and multi-robot systems. These limitations led to the development of ROS 2.

ROS 2 communicates with middleware based on the Data Distribution Service (DDS) standard, which provides improved performance, security, and flexibility compared to the communication infrastructure of ROS 1. It has the advantages of operating machine-to-machine with an Object Management Group (OMG) for the distributed systems using a publish-subscribe pattern. Figure Figure 2.3 is visualizing how ROS is build up with OS, middleware and as a application layer [33].

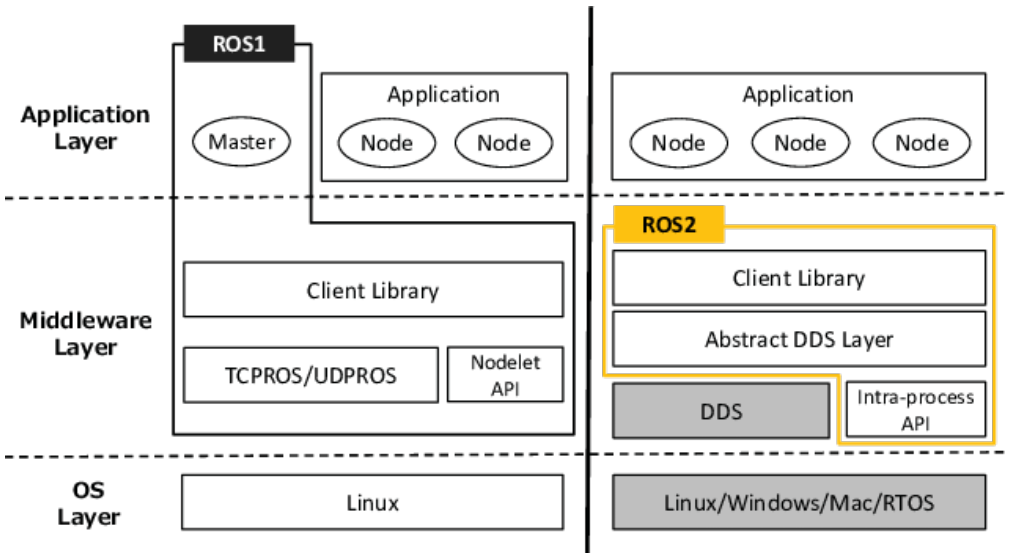


Figure 2.3: ROS1 and ROS2 architecture

2.3.1 MoveIt

MoveIt2 is the second version of the motion planning framework MoveIt, which is an effective collection of tools for autonomous robots. It is an open-source project that provides kinematics, motion planning, trajectory processing, and collision detection capabilities, as the main features. The framework is intended to integrate with the latest iteration of ROS, ROS 2. MoveIt2 is distinguished by its motion planning capabilities. It contains a number of algorithms for path planning, including OMPL (Open Motion Planning Library) and CHOMP (Covariant Hamiltonian Optimization for Motion Planning) [15]. These algorithms enable robots to plan complex manipulation in cluttered environments, making MoveIt2 a valuable instrument for a variety of applications, including industrial automation and robotics research. MoveIt2 also support various types of robot packages developed to control robotic

manipulators such as Universal Robot UR10 [25].

Additional software packages is available for controlling the desired goal movement, as input to MoveIt2. pyMoveIt is a open-source software developed for interfacing developed software with MoveIt2, with Python-based tools and libraries to control robotic manipulators.

2.3.2 RViz

As part of ROS, Robot Visualization (RViz) is an open-source 3D visualization application. It is a interface for visualizing the status, sensor data, and trajectories of a robot in a simulated environment by interacting with ROS nodes and topics. RViz is widely used in the software development process in robotics. RViz can integrate 3D models of the collaborative robot's (cobot's) working environment, enabling developers to visualize the environment, including obstacles and objects with which the cobot will interact. This visualization helps to comprehend the context in which the cobot operates and can aid in software debugging by providing visual signals regarding the cobot's interaction with its surroundings. In a pick-and-place assignment, the cobot must locate an object, pick it up, and then relocate it. RViz is able to display and execute planned paths, providing developers with a visual depiction of the cobot's intended movement. Developers are able to modify parameters in real-time and observe the effects on the cobot's path, enabling an iterative approach to path optimization.

2.4 Modbus Application Protocol

Modbus RTU (Remote Terminal Unit) is a binary protocol that communicates via a master-slave architecture over a serial line, typically RS-232 or RS-485, and was devised by Modicon in 1979. This protocol's distinguishing characteristics, such as its simplicity, dependability, and sturdiness, have played a significant role in its pervasive adoption across industries [32]. Modbus RTU is based on a data paradigm composed of four fundamental data types: discrete inputs, coils, input registers, and holding registers. Each slave device is allocated a unique 8-bit network address, enabling the master device to initiate transactions (queries) utilizing function codes that specify the required action, such as reading from or writing to the slave devices. Modbus RTU queries and responses follow a specific data frame structure. Each frame begins with the address of the slave device, is followed by a function code, and concludes with a CRC for error detection. This frame structure's simplicity and standardization contribute to the protocol's robustness and ease of implementation. Modbus RTU is utilized in numerous applications, including industrial and building automation and remote monitoring systems, due to its adaptability and durability. Its continued relevance derives in part from its capacity to bridge the gap between legacy systems and emerging Internet of Things (IoT) technologies [13].

2.5 Motion Control for Robotic Manipulator

There are several intricate sub-tasks involved in an autonomous pick-and-place procedure where objects are randomly put. The perception of the item, pick trajectory planning, motion control, pick execution, object transfer, and ultimate target location placement are some of these. When things are arbitrarily positioned and/or maybe in different orientations, the complexity rises. Modern technological developments have produced sophisticated techniques for regulating motion in these activities.

2.5.1 Motion Control with Inverse Kinematics

The calculation of joint parameters that place a robot's end-effector at a desired location is known as "inverse kinematics," and it is an essential concept in robotics. Forward kinematics, on the contrary, determines the position of the end-effector based on a set of joint parameters. Inverse kinematics is frequently more pertinent in real-world robotics applications like pick-and-place activities. When the target location is known, the objective of inverse kinematics is to calculate the necessary joint angles to reach the target location. There are mathematically possible for a system to have several solutions or, in rare situations, none at all if the desired place is impassable. Inverse kinematics problems are solved using a variety of methodologies, including geometric, numerical, and optimization approaches. More recently, data-driven machine-learning paradigms have also been investigated [9].

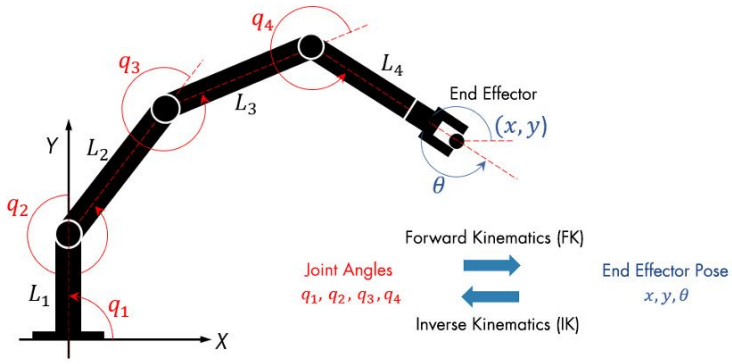


Figure 2.4: Calculating Joint Position with Kinematics [9]

The motion planning method uses the coordinates of the discovered object to determine the robot's trajectory. This trajectory is then executed by the robot control system, but some limitations may apply if insufficient DoF.

2.5.2 Motion Control with Reinforcement Learning

Recently, robot control for challenging tasks like pick-and-place has demonstrated encouraging outcomes when using ML approaches, particularly reinforcement learning (RL). These methods can deal with partial observable and high-dimensional continuous action spaces, both of which are frequent difficulties in such assignments. Instead of requiring explicit motion planning, RL-based systems can develop a control strategy directly from raw sensor data, such as images.

A hybrid approach combines ML approaches with conventional robotic control methods like PID controllers, inverse kinematics, or Jacobian-based algorithms. For example, while traditional control techniques are utilized for motion control, ML can be employed for perception (object detection, posture estimation). Alternately, learning-based techniques can be used to teach the robot's inverse kinematics function, making it easier to regulate the end-effector position for objects that are placed arbitrarily.

2.6 Reinforcement Learning

Reinforcement Learning (RL) is a branch of ML where an agent learns to make decisions by interacting with its environment. The agent takes actions based on its current state, and receives feedback in the form of rewards or penalties. The goal of the agent is to learn a policy that maximizes the cumulative reward over time.

2.6.1 On-Policy

In on-policy learning, the agent learns the value of the policy being carried out by the agent while it is interacting with the environment. The learned policy dictates the future action that the agent will take given the current state. One of the key methods used in on-policy learning is the Monte Carlo (MC) method, which involves learning from complete episodes of interaction before updating the policy.

The most popular on-policy learning algorithm is Proximal Policy Optimization (PPO) which is a policy gradient method that uses the advantages of both the first-order policy gradient and natural policy gradient methods. PPO performs multiple epochs of stochastic gradient descent on a single batch of data, making small updates to the policy after each epoch. This approach mitigates the policy degradation problem found in traditional policy gradient methods, leading to more stable and efficient learning.

Another widely used on-policy algorithm is the Advantage Actor-Critic (A2C) algorithm. A2C is a type of actor-critic method, a family of algorithms that includes two components: an actor, which is used to select actions, and a critic, which is used to estimate the value function of the current policy. A2C uses the critic's value estimate to calculate the advantage, which measures how much better an action is compared to the average action for a given state.

2.6.2 Off-Policy

In off-policy learning, the agent learns the value of the optimal policy independently of the agent's actions. The agent maintains an exploratory policy for learning (known as the behavior policy), and a separate target policy that it seeks to optimize. This allows the agent to learn from past experiences, stored in a replay buffer, and can lead to more sample-efficient learning.

The most well-known off-policy learning algorithm is Q-Learning, which learns the action-value function and uses it to derive a policy. The action-value function, also known as Q-function, represents the expected return of taking an action in a particular state following a policy. Q-Learning uses a form of Temporal Difference (TD) learning, a combination of Monte Carlo ideas and dynamic programming (DP) ideas, to update its estimates based on the Bellman equation for optimal policies.

Deep Q Network (DQN) is a variant of Q-Learning where a deep neural network is used to approximate the Q-function. DQN introduces two key features: experience replay and target Q-networks. Experience replay allows the network to learn from past decisions, improving sample efficiency and breaking the correlation between experiences. The target Q-networks help to stabilize the learning process.

Another off-policy method is the Deep Deterministic Policy Gradient (DDPG) which is an actor-critic algorithm that extends the DQN method to continuous action spaces. DDPG utilizes two networks: an actor network that outputs the deterministic policy and a critic network that outputs the Q-value of the action given by the actor network. DDPG uses a replay buffer to store past experiences, from which it samples mini-batches for training, and uses soft updates to update the target networks, which contributes to stable learning.

Soft Actor-Critic (SAC) is an off-policy algorithm that aims to maximize the expected return and also encourages exploration by maximizing the entropy of the policy. SAC uses a form of the actor-critic method where two Q-functions are learned and used for updating the policy

to encourage exploration. It's especially effective in tasks with continuous action spaces. The PPO algorithm is highly regarded for its balance between sample efficiency, ease of implementation, and computational cost-effectiveness.

2.7 Omniverse Isaac Sim - Gym Environment

Omniverse Isaac Sim Gym is an extension of NVIDIA's Isaac Sim robotics simulation platform, and it is specifically designed to facilitate reinforcement learning (RL) research. It achieves this by leveraging NVIDIA's GPU-accelerated PhysX simulation engine to gather the experience data required for robotics RL, making RL-based training more accessible [18].

Isaac Gym allows for thousands of simultaneous environments on a single GPU, significantly reducing the computational resources required for physically accurate simulations used in training RL algorithms. It includes support for importing URDF and MJCF files, it provides a PyTorch tensor-based API to access the results of physics simulations, and it eliminates costly data transfers between the GPU and the CPU, which can often be a performance bottleneck in RL tasks [26].

Isaac Gym also includes a basic Proximal Policy Optimization (PPO) implementation and supports alternative task systems or RL algorithms. It is compatible with PyTorch, and with some customization, it can also be integrated with TensorFlow-based RL systems. Some additional features of Isaac Gym include support for various environment sensors, runtime domain randomization of physics parameters, and Jacobian/inverse kinematics support.

The core functionality of Isaac Gym is made available as part of the NVIDIA Omniverse Platform and NVIDIA's Isaac Sim. The Omniverse Isaac Gym extension simplifies the process of connecting reinforcement learning libraries and algorithms with other components in Isaac Sim, providing an interface that can be used as a bridge connecting RL libraries with physics simulation and tasks running in the Isaac Sim framework.

There have been various updates to Isaac Gym, such as the Preview 4 Release, which aligns the PhysX implementation in the standalone Preview Isaac Gym with Omniverse Isaac Sim to simplify migration to Omniverse for RL workloads and adds new features such as support for SDF collisions and additional Factory RL samples.

2.7.1 Universal Scene Description (USD)

Universal Scene Description (USD) is the 3D model format in Omniverse and Isaac Sim [17]. USD represents a 3D scene using a hierarchy of objects, each with its own properties, such as geometry, materials, and transformations. It also includes a system for overrides, allowing a base scene to be modified non-destructively. USD files can be binary or ASCII text, and they can include references to other files, allowing complex scenes to be broken into manageable pieces.

2.7.2 Converting to USD Format in Omniverse

NVIDIA Omniverse supports a variety of common 3D model file extensions, which can be converted to USD format. These include but are not limited to [16]:

- **.fbx** (Autodesk FBX)
- **.obj** (Wavefront Object)

- **.3ds** (3D Studio)
- **.dae** (COLLADA)
- **.stl** (Stereolithography)
- **.ply** (Polygon File Format or Stanford Triangle Format)
- **.gltf** and **.glb** (GL Transmission Format)

The conversion process can vary depending on the specific format and complexity of the model. Some formats may be converted directly in Omniverse Create, while others require an intermediary tool or extension. URDF and MJCF, as mentioned in Section 2.7, can be converted through an extension. Post-conversion, manual tweaking may be necessary to ensure the model looks and behaves correctly in Isaac Sim.

2.8 Related Work

2.8.1 "Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching"

This study introduces a robotic pick-and-place system capable of grasping and recognizing both familiar and unfamiliar stationary objects in congested environments. The system is capable of processing a wide variety of object classes without requiring task-specific training data for novel objects. Utilizing a category-independent affordance prediction algorithm, it selects and executes one of four grasping primitive behaviors. The system then recognizes objects using a framework for cross-domain image classification that matches observed images to product images. Since product images are readily available for a wide variety of objects (e.g., from the internet), the system functions out of the box for novel objects without the need for additional training data. The strategy was a component of the MIT-Princeton Team's system that won the stowing competition at the 2017 Amazon Robotics Challenge [34].

2.8.2 "Reinforcement Learning for Pick and Place Operations in Robotics: A Survey"

In recent years, significant progress has been made in robotics, with an emphasis on training robotic agents with reinforcement learning (RL) for pick-and-place operations. Typically performed by logistics robots, these tasks are progressively being completed without the assistance of a robotics engineer. The process of reinforcement learning incorporates a number of fundamental concepts, such as value iteration, policy search, reward shaping, and imitation learning. In the context of pick-and-place tasks, pose estimation and simulation environment are additional factors to consider. In spite of the substantial progress made in the field, additional research is required to advance experiment validation, model generalization, and grasp pose selection.

> "Generalizing the training samples could make the pick-and-place task more widely applicable. The idea is conceptually equivalent to domain randomization in which the task is randomized rather than the environment." [6]

2.8.3 Challenges within pick-and-place with AI

Challenges in robotics and AI, potential limitations could involve the system's ability to handle complex environments, its precision in object recognition, or its reliance on product images for identifying novel objects. There is little existing research on how to pick and place objects based on only the view from one RGB camera of moving objects that are placed randomly. In general, systems like these often have limitations related to their ability to generalize to new situations, handle noisy or incomplete data, or function reliably in real-world conditions. Our research shows that recent progress in the various technologies we have combined in our implementation leads to promising results for pick and place using only one RGB camera.

Chapter 3

System Design

3.1 Experimental Setup

The hardware applied in our system implementation is a conveyor belt for moving the objects, an RGB camera for vision, and a collaborative robot arm for the pick-and-place operation. The rest of the resources used, with specifications, are listed in the table below.

Table 3.1: Main Equipment Used in the Project

Name	Producer	Type
Robot	Universal Robot	UR10e
Gripper	Onrobot	RG6
RGB Camera	ZCam	E2-S6
GPU hosting PC	Nvidia	RTX4090
Conveyor belt	StellAi	200cm x 82cm x 55cm
Single Board Computer	Jetson	Nano

For the stands and attachments, mechanical adaptations are produced specific to this project and were made at the University of Agder with different types of steel.

3.2 Selection of Instance Segmentation Model

In this section, we present a comprehensive analysis to determine the most suitable instance segmentation model for our project. The selection criteria were based on several key performance indicators, including:

- **Training Speed:** The time required to train the model has significant implications for the project timeline and resource allocation. Faster training speeds allow for more iterations and refinements to the model.
- **Inference Speed:** The speed at which the model can process new data and make predictions is crucial for real-time applications. Lower latency during inference ensures the model can deliver timely results.
- **Box Average Precision (BAP) @0.5-0.95:** BAP is an important measure of a model's accuracy. It quantifies the average precision value for a range of intersection over union (IoU) thresholds (from 0.5 to 0.95 in this case).
- **Mask Average Precision (MAP) @0.5-0.95:** Similar to BAP, MAP is another important metric in evaluating instance segmentation models. It provides an overview of how well the model performs across different IoU thresholds for mask prediction.

- **Performance on Small Amount of Data:** The ability of a model to perform well with limited training data can be critical in scenarios where data collection is challenging or expensive.
- **Modifiability:** The ease with which a model can be modified to fit specific project requirements is also a key consideration. This could encompass changes in the architecture, loss functions, or training regimen.

This project necessitated a state-of-the-art instance segmentation model capable of real-time operation. The application required minimal latency combined with high accuracy. During the search for an appropriate model, two prominent ones were considered: RTMDet-Ins and YOLOv8-seg.

- **YOLOv8 for Real-Time Instance Segmentation:** YOLOv8 is a state-of-the-art real-time instance segmentation model known for its high accuracy and fast execution speed. It is user-friendly, easy to train, and implements a normalized coordinate system for bounding polygons [5].
- **RTM-Det for Real-Time Instance Segmentation:** RTM-Det is a high-performance real-time object detector that can be easily extended for tasks such as instance segmentation. It uses large-kernel depth-wise convolutions and advanced training strategies to achieve high accuracy and speed [8].

RTM-Det and YOLOv8 achieve state-of-the-art performance on the COCO benchmark, and RTM-Det succeeds in performing slightly better. RTM-Det-Ins-X achieves 44.6 mask AP and YOLOv8 43.4 mask AP. YOLOv8 was still picked as the better choice for real-time instance segmentation due for the following reasons:

1. **Ease of Training:** YOLOv8 is known for its simplicity in training, which can significantly benefit teams with limited resources or projects with tight deadlines. It provides satisfying results even with small datasets and short training times, making it more practical for many real-world applications.
2. **Performance:** YOLOv8 excels in terms of both accuracy and execution speed, indicating that it can deliver superior results in real-time instance segmentation tasks.
3. **Normalization of Coordinates:** YOLOv8's normalized coordinate system for bounding polygons simplifies the interpretation and utilization of the model's outputs. This can be particularly valuable in real-time instance segmentation, where quick and accurate interpretation of model outputs is crucial.
4. **Wide Adoption:** As part of the widely adopted YOLO series, YOLOv8 benefits from a broad community of users and developers. This wide adoption can lead to better support and resources, making troubleshooting issues, finding pre-trained models, or accessing tutorials and other educational materials easier.

3.3 Selection of Motion Control Technique

Guidelines were clarified before the project started. A desire to use state of the art technology, especially in the use of AI. Motion control is one of the most essential aspects of operating robotics, where Inverse Kinematics is the most common choice for moving the toolpoint on a robotic manipulators to a desired location. Reinforcement learning is also a proven method for controlling the movement of a manipulator to a desired goal, but not as explored and tested as IK. An analysis has been made, based on the task and the objectives.

IK is a well-known technique for controlling robotic arms, hence it has been proven effective and robust. IK can offer the necessary joint settings for the cobot to successfully complete the operation given the object's coordinates from the detection algorithm. Since it is based on mathematical equations it has a deterministic behavior and allows the cobot movement to be foreseen and accurately regulated, which is vital in settings where reliability and safety are top priorities. Once computed it is able to execute operations in real-time, which is crucial in a dynamic environment with moving objects. IK may struggle with more sophisticated activities, such as manipulating the object while picking or placing, yet working well for simple pick-and-place actions. The techniques has lack innate adeptness. The IK method might not adapt without modification if the environment changes, such as the appearance of additional items [9].

Reinforcement learning on the other hand may be able to adjust to changes in the task or environment more effectively. The cobot may learn from its interactions with the environment and gradually enhance its performance. It often has the ability to learn complex tasks, with the potential to learn more intricate manipulation techniques that may be challenging to simulate with IK. To develop an efficient policy, RL needs a lot of training time and data. It might be difficult to ensure consistent performance for RL algorithms due to stability and convergence problems.

Due to its deterministic nature and real-time performance, IK appears to be a more viable option for straightforward pick-and-place operations in light of the above factors. However, RL could be able to perform and adapt more effectively in surroundings or tasks that are more challenging. Based on the unique application requirements, and limitations, each method was selected to be implemented, analyze, and compared.

3.4 Selection of Reinforcement Learning Algorithm

This section conducts a technology evaluation and analysis to determine the best reinforcement learning (RL) algorithm for controlling the motion of a collaborative robot (cobot) performing the pick-and-place operations. The primary objective is to select an RL algorithm that can effectively and efficiently learn and implement the motion control policies required for the cobot to handle various objects autonomously during the sorting process.

To determine the optimal RL algorithm, the following criteria must be taken into account:

- **Convergence speed:** The RL algorithm should be capable of rapidly learning the optimal control policy, minimizing training time and allowing for rapid deployment in real-world scenarios.
- **Robustness:** The algorithm should be able to manage a wide variety of objects and scenarios, such as variations in object size, shape, and material, as well as environmental changes.
- **Scalability:** The chosen RL algorithm must be able to accommodate changes in the sorting process's complexity, such as the addition of new object classes, without a significant performance degradation.
- The algorithm should be simple to implement and incorporate with the existing system architecture and hardware.

Several RL algorithms have demonstrated potential for controlling robotic systems for a variety of tasks, including pick-and-place operations. The primary applicants for this position are:

- **Deep Q-Network (DQN):** integrates Q-learning and deep neural networks, allowing the RL agent to deal with high-dimensional state and action spaces. DQN has shown success in a variety of robotic control tasks, but its convergence pace may be slower than that of other algorithms.
- **Proximal Policy Optimization (PPO):** is an algorithm for policy optimization that balances the exploration and exploitation trade-off. It has demonstrated excellent performance in numerous robotic tasks and exhibits quicker convergence than DQN.
- **Deep Deterministic Policy Gradient (DDPG):** is an off-policy actor-critic algorithm that can manage continuous action spaces, making it suitable for robotic system control. DDPG has shown excellent performance in a variety of robotic tasks, but it may be more sensitive to hyperparameter tuning than PPO.
- **Soft Actor-Critic (SAC):** is an off-policy actor-critic algorithm that incorporates an entropy term to encourage exploration. It performs well in robotic control tasks and tends to converge more quickly than DDPG.

On the basis of the aforementioned criteria and the performance of the candidate algorithms in similar applications, we recommend using the Proximal Policy Optimization (PPO) algorithm for the motion control of the cobot during pick-and-place operations. PPO provides a balance between exploration and exploitation, converges faster than other algorithms, and is relatively simple to implement and integrate.

In addition, PPO has demonstrated its robustness and scalability in a variety of robotic control tasks, making it well-suited for managing the complexities of the pick-and-place operations in the recycling sorting process. To ensure optimal performance and seamless integration with the existing system architecture and hardware, it is necessary to execute exhaustive testing and validation of the chosen RL algorithm in the context of the specific application.

3.5 Design of Communication

A communication setup is desirable for sharing data from the vision system with the different motion control systems. A flowchart describing the process at each program package is at the end of each respective section.

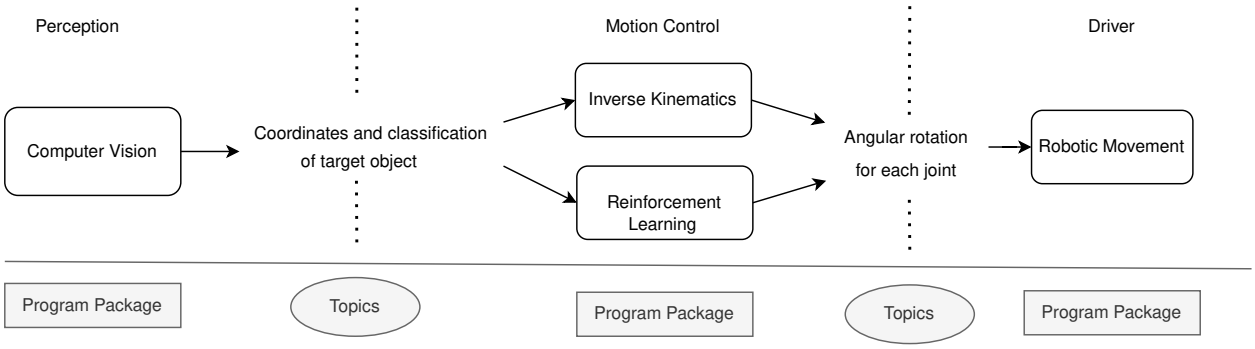


Figure 3.1: Design of System Architecture

Chapter 4

Computer Vision with Real-Time Instance Segmentation

This chapter presents the developed methodology for RT object detection and instance segmentation using machine learning and describe how it is implemented in the context of this master thesis project. The system is designed to output grip point in world coordinates based on perception from one RGB camera. This chapter also discuss the implementation of object tracking, optimal grip point calculation, and real-time control of the robot and gripper using ROS nodes.

4.1 Applying Perception to a Robotic Manipulator

The approach combines several technologies, such as ROS2, YOLOv8 with segmentation (YOLOv8x-seg), and BYTETracker, to deliver a robust and real-time solution for object detection, tracking, and instance segmentation. This forms a vital part of the system that enables accurate and efficient manipulation of objects by a robot.

4.1.1 Real-Time Performance

In this context, real-time performance doesn't only mean speed but also the ability to process data as it comes in and to provide output that is timely and actionable in a continuously running system. The overall approach, combining a real-time object detection and instance segmentation model with a robust tracking solution and efficient data handling procedures, ensures the achievement of real-time instance segmentation in this system.

4.2 Object Detection and Tracking

To achieve a usable pick and place robot in the context of the challenges faced in this project, it is required to gain the knowledge about whether an object has already been identified or if a new object have entered the field of view. The object detection computes one image at a time, and the accuracy of the tracking algorithm is essential to not pass object detection data multiple times for the pick-and-place operation.

4.2.1 Object Detection and Instance Segmentation with YOLOv8

For the task of object detection and instance segmentation, the project employs YOLOv8, integrated within the ROS2 DDS node system. The YOLOv8 algorithm was launched on January 10th, 2023. Since then it has undergone numerous iterations for improvements and bug fixes. One part of the workflow has been to periodically update to the the latest version,

modify source code for expected performance and fix bugs.

The project leverages Roboflow to streamline the data pipeline, storing the dataset in the cloud, tracking dataset versions, and facilitating custom dataset creation (Section 2.2.6). The YOLOv8x-seg, a large segmentation model pre-trained on the COCO segmentation dataset, was utilized in the project. Using transfer learning, the model was fine-tuned to our custom dataset, leading to impressive results. With just a dataset of 50 images, each containing about 10 instances of each class, the model achieved a mean Average Precision (mAP) at 50% Intersection over Union (IoU) threshold of 92%. This demonstrated that creating an extensively large custom dataset was unnecessary, provided that appropriate augmentation techniques were chosen and labeling was conducted thoroughly.

The augmentation techniques for dataset creation were chosen based on logical assumptions and experimental results. The environment, while homogeneous, experiences slight variations in lighting. To cope with these changes and ensure model robustness, several augmentation techniques were applied, such as flipping horizontally and vertically, rotating, and adding noise. Further, the objects were manually placed on the moving conveyor belt with all sides up, preferrably with a new yaw orientation. Both the manual placement and added augmentation for a homogeneous environment proved to be adequate.

The output of the final model can be seen in Figure 4.1, which was trained on a dataset comprising 943 images. Of these, 822 were used for training, and 121 for validation.

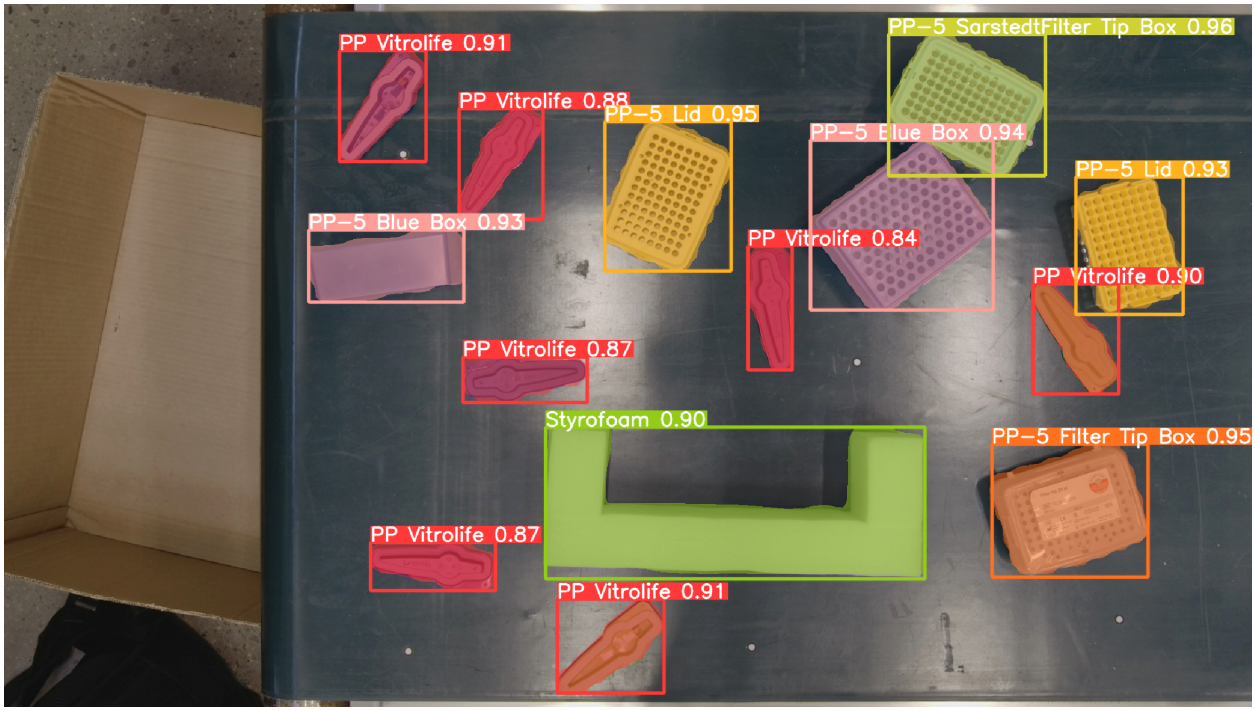


Figure 4.1: Instance Segmentation with YOLOv8

4.2.2 Object Tracking

The BYTETrack object tracking algorithm has been implemented on top of the YOLOv8 detection result generator. The tracking algorithm is based on bounding box coordinates, confidence score and class id. The tracker matches detected objects with their historical data, assigning unique tracker IDs to newly detected objects. These IDs enable the system to track objects across multiple frames, proving to be reliable even when objects move in and

out of the frame. The configuration of the tracker is governed by the thresholds presented in Table 4.1.

Attribute	**Default Value**
track_thresh	0.25
track_buffer	30
match_thresh	0.8
aspect_ratio_thresh	3.0
min_box_area	1.0
mot20	False

Table 4.1: Attributes and their set values for the Object Tracker argument class.

4.3 Camera Calibration

The placement of the RGB camera, which provides the only sensory perception of the physical environment, is essential to the success of the object detection algorithm. Positioned directly above the conveyor belt, the camera provides a view from the top down, with zero tangential distortion. By mounting the camera parallel to the conveyor belt, distortion is limited to one direction since the objects passes by and the image get captured in optical center. This placement strategy is especially important in the absence of additional sensing technologies like LiDAR, as it enables the system to maintain a high degree of accuracy in object detection and subsequent operations [10].

Camera calibration is performed using OpenCV to obtain accurate camera intrinsic and extrinsic values. Functions for transforming coordinates from image space to world space are implemented.

Camera calibration, in the field of computer vision and photogrammetry, is a process that determines the parameters of a camera. This process is crucial because it provides the relationship between the 3D world coordinates and the 2D image coordinates. Understanding this relationship allows for accurate extraction of metric information from 2D images.

The main parameters that are determined during camera calibration are:

- Intrinsic Parameters: These are specific to the camera and include information such as focal length and optical centers. They also include lens distortion parameters.
- Extrinsic Parameters: These pertain to the camera's position and orientation in the world.

Once these parameters are known, they can be used to correct images, compensating for lens distortion and allowing for accurate measurement within the image. For translating image coordinates to world coordinates the intrinsic parameters is the only needed information, which is further explained in Section 4.3.1.

The intrinsic parameters are represented by the camera matrix A as:

$$A = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where f_x and f_y are the focal lengths along the X and Y axes, c_x and c_y are the coordinates of the optical center (also called principal point), and s is the skew coefficient.

The extrinsic parameters describe the position and orientation of the camera in the world and are represented by a rotation matrix R and a translation vector T.

Camera calibration was done by taking images of a checkerboard pattern, and then solving an optimization problem to estimate these parameters. The optimization problem seeks to minimize the difference between the observed image points and the projected 3D points using the estimated camera parameters.

This is formulated as minimizing the re-projection error e, defined as:

$$e = \sum_{i=1}^{N} ||x_i - x_i'||^2$$

where x_i are the observed 2D points in the image, x_i' are the projected 2D points using the estimated camera parameters, and N is the total number of points.

The 3D object points and corresponding 2D image points are needed to solve this optimization problem. The 3D object points are the corners of a checkerboard pattern and are assumed to lie on a plane at z=0 in the world coordinate system. The 2D image points are the corresponding checkerboard corners detected in the image.

The relationship between the 3D object points in the world coordinate system X_w, the 2D image points x_i, and the perspective projection model can describe the camera parameters as:

$$sx_i = A[R|T]X_w$$

where s is the scale factor.

The estimated camera matrix A, rotation vectors R, and translation vectors T can then be used to relate points in the 3D world to points in the 2D image plane, thus enabling accurate computer vision tasks such as object detection, pose estimation, and 3D reconstruction.

4.3.1 World coordinates

The focal length of the camera is essential when dealing with distortion. The world coordinates are calculated using the equation (4.1). For the project there is a static Z value with minor differentiation based on the depth of the object in the image.

$$X = \frac{x_p - C_x}{F_x} \cdot Z \qquad Y = \frac{y_p - C_y}{F_y} \cdot Z \quad (4.1)$$

X	World coordinate in X
x_p	Pixel coordinate in x
C_x	Optical center point in x
F_x	Focal length in x
Y	World coordinate in Y
y_p	Pixel coordinate in y
C_y	Optical center point in y
F_y	Focal length in y
Z	World coordinate Z

4.4 Optimal Grip Point Prediction

An algorithm is developed for predicting optimal grip points, accounting for the object's aspect ratio, and considering the presence of inactive pixels within an object. The algorithm calculates the optimal grip points' distance, angle, and edge coordinates. The optimized grip point algorithm is implemented to run along with the ROS Node, publishing to the topic grip_x, grip_y, grip_angle (radians), and grip_width in world coordinates. The following pseudocode describes the general implementation of the Algorithm 1:

Algorithm 1 Optimal Grip

Require: $mask\_data, focal\_length\_x, focal\_length\_y, optical\_center\_x,$
$optical\_center\_y, min\_grip\_width$
$w, h \leftarrow mask\_data[1]$
$center\_x, center\_y \leftarrow mask\_data[3]$
$grip\_width \leftarrow \min(w, h)$
if $w > h$ **then**
    $y\_bound \leftarrow \lfloor h/2 \rfloor$
    $x\_bound \leftarrow \lfloor w * 0.25 \rfloor$
    $roi \leftarrow mask\_data[4][(center\_y - y\_bound) : (center\_y + y\_bound), (center\_x - x\_bound) : (center\_x + x\_bound)]$
    $grip\_x, grip\_y, grip\_angle, grip\_width \leftarrow width\_greater(roi, min\_grip\_width, grip\_width)$
    $grip\_x \leftarrow center\_x - x\_bound + grip\_x$
    $grip\_y \leftarrow center\_y - y\_bound + grip\_y$
    $grip\_x, grip\_y \leftarrow$ **to_world_coordinates**$(grip\_x, grip\_y, optical\_center\_x, optical\_center\_y, focal\_length\_x, focal\_length\_y)$
else
    $x\_bound \leftarrow \lfloor w/2 \rfloor$
    $y\_bound \leftarrow \lfloor h * 0.25 \rfloor$
    $roi \leftarrow mask\_data[4][(center\_y - y\_bound) : (center\_y + y\_bound), (center\_x - x\_bound) : (center\_x + x\_bound)]$
    $grip\_x, grip\_y, grip\_angle, grip\_width \leftarrow height\_greater(roi, min\_grip\_width, grip\_width)$
    $grip\_x \leftarrow center\_x - x\_bound + grip\_x$
    $grip\_y \leftarrow center\_y - y\_bound + grip\_y$
    $grip\_x, grip\_y \leftarrow to\_world\_coordinates(grip\_x, grip\_y, optical\_center\_x, optical\_center\_y, focal\_length\_x, focal\_length\_y)$
end if
return $grip\_x, grip\_y, grip\_angle, grip\_width$

4.5 YOLO training

Under training YOLOv8 utilizes Multi-Scale Object Detection by a feature pyramid network to detect objects of different sizes and scales within an image. In addition Mosaic is used, which is an improvement to the CutMix data augmentation. The Mosaic technique takes 4 images, resizes them, stitch them together, and then taking a random cutout of the stitched images to get the final Mosaic image. These techniques helps to generalize the final model and be able to predict and localize objects better wherever they are within the frame and at whatever size.

Some key YOLO training configurations that is set Table 4.2:

Parameter	Value
Task	Segment
Model	yolov8x-seg.pt
Epochs	100
Batch size	16
Image size	640
Optimizer	SGD
Initial learning rate (‘lr0‘)	0.01
Final learning rate (‘lrf‘)	0.01
Momentum	0.937
Weight decay	0.0005
Dropout	0.0
Intersection over Union (‘iou‘)	0.7
Mixed precision training (‘amp‘)	true
Data augmentation (‘augment‘)	false

Table 4.2: Key Configurations of the YOLOv8 model

4.6 Perception Pipeline

The perception pipeline involves several steps that utilize ROS2 nodes, tracking algorithms, and advanced computer vision techniques to achieve accurate real-time perception for the robot:

1. A ROS2 *node* is initiated with the object tracking, detection and instance segmentation. A separate thread is initiated to publish the results in parallel.

2. From the resulting detections, only those that meet specific criteria are selected; located within the center 20% of the image width to minimize field of view distortion. This process involves limiting the field of view and saving the indices of the detections that meet the criteria.

3. The chosen detections are then transformed into *Detection* objects that contain necessary information such as *bounding box* coordinates, *confidence score*, and *class id*.

4. The selected detection results are also stored in a *Dictionary*, which includes additional data such as *masks* and *segmentation*.

5. The *Detection* objects and the *Dictionary* are then pushed into a First In, First Out (FIFO) *Queue*, ensuring that data is processed in the order it was received.

6. In the publishing thread, the object *tracker* is updated with the latest detections, and these are matched with existing *tracks* to maintain continuity of object identities over time.

7. A *set* is updated with the unique *tracker ids* for detections.

8. Using the *masks* of objects with unique *tracker ids*, the *optimal grip points* for the robotic manipulator are calculated.

9. Finally, the *pose* of each uniquely tracked object is published for use in downstream processes.

The publishing function running in the ROS2 node handling all the detection data in a parallel thread, is described by the following pseudocode Algorithm 2:

Algorithm 2 Object ID Tracking - Publish Object Pose With Unique IDs

Require: $queue, object\_tracker, publisher\_, names\_dict,$
$focal\_length\_x, focal\_length\_y, optical\_center\_x, optical\_center\_y$

while True **do**
  $detections, object\_pos, time\_stamp \leftarrow queue.get()$
  $tracks \leftarrow object\_tracker.update(detections2boxes(detections))$
  $tracker\_id \leftarrow match\_detections\_with\_tracks(detections, tracks)$

  Filter out *detections* without $tracker\_id$ and update *detections*
  Filter out $object\_pos$ without $tracker\_id$ and update $object\_pos$

  for each $detection\_id$ not in $check\_id$ **do**
    Add $(detection, object\_pos)$ to $publish\_queue$
    Add $detection\_id$ to $check\_id$
  end for

  $mask\_data \leftarrow publish\_queue.get()$
  $grip\_args \leftarrow$ **optimal_grip**$(mask\_data, focal\_length\_x,$
  $focal\_length\_y, optical\_center\_x, optical\_center\_y)$

  $cls\_id \leftarrow mask\_data[5]$
  $grip\_x, grip\_y, angle\_rad \leftarrow grip\_args$

  $msg.header.stamp \leftarrow time\_stamp$
  $msg.header.frame\_id \leftarrow names\_dict[cls\_id]$
  $msg.pose.position.x \leftarrow grip\_x$
  $msg.pose.position.y \leftarrow grip\_y$
  $msg.pose.position.z \leftarrow 0.10$

  $angle\_deg \leftarrow rad\_to\_deg(angle\_rad)$
  $quat\_x, quat\_y, quat\_z, quat\_w \leftarrow deg\_to\_quaternions(x = 180, y = 0, z = angle\_deg)$

  $msg.pose.orientation.x \leftarrow quat\_x$
  $msg.pose.orientation.y \leftarrow quat\_y$
  $msg.pose.orientation.z \leftarrow quat\_z$
  $msg.pose.orientation.w \leftarrow quat\_w$

  Publish *msg* using $publisher\_$
end while

4.7 Evaluating dataset

In the domain of computer vision, particularly object detection, the concept of a label correlogram proves invaluable for understanding the relationships and dependencies within the data. The label correlogram, a form of scatterplot matrix or pairplot, provides a visual representation of the relationships between the variables defining each object's bounding box in the dataset, namely the x-coordinate, y-coordinate, width (w), and height (h) – a space commonly referred to as the 'xywh' space.

The correlogram comprises a series of 2D histograms where each plot displays the relationship between two different variables. These variables pertain to the properties of the bounding boxes that encapsulate the detected objects within the images. Specifically, the 'x' and 'y' denote the coordinates of the upper-left corner of the bounding box, while 'w' and 'h' denote the width and height of the bounding box, respectively.

The analysis of a label correlogram in 'xywh' space provides several valuable insights. For instance, it can reveal whether there is a correlation between the width and height of the bounding boxes across the dataset. This could indicate a pattern in the sizes of objects being detected and, in turn, could influence the design of the detection algorithm. Similarly, the relationship between 'x' and 'y' coordinates might suggest spatial trends in object placement within the images, which could be related to the orientation of objects or inherent bias in the image acquisition process.

Furthermore, the correlogram can elucidate potential correlations between the position and size of the objects. For instance, if a correlation is observed between 'x' and 'w' or 'y' and 'h', it could suggest a tendency for larger objects to appear more frequently in certain regions of the image.

4.8 Flowchart of Perception Pipeline

The flowchart of the perception pipeline is visualized in Section 4.8. It breaks down the process of the developed vision system. Where the input is given from an RGB camera and the output is the grip point sent to the motion control systems.

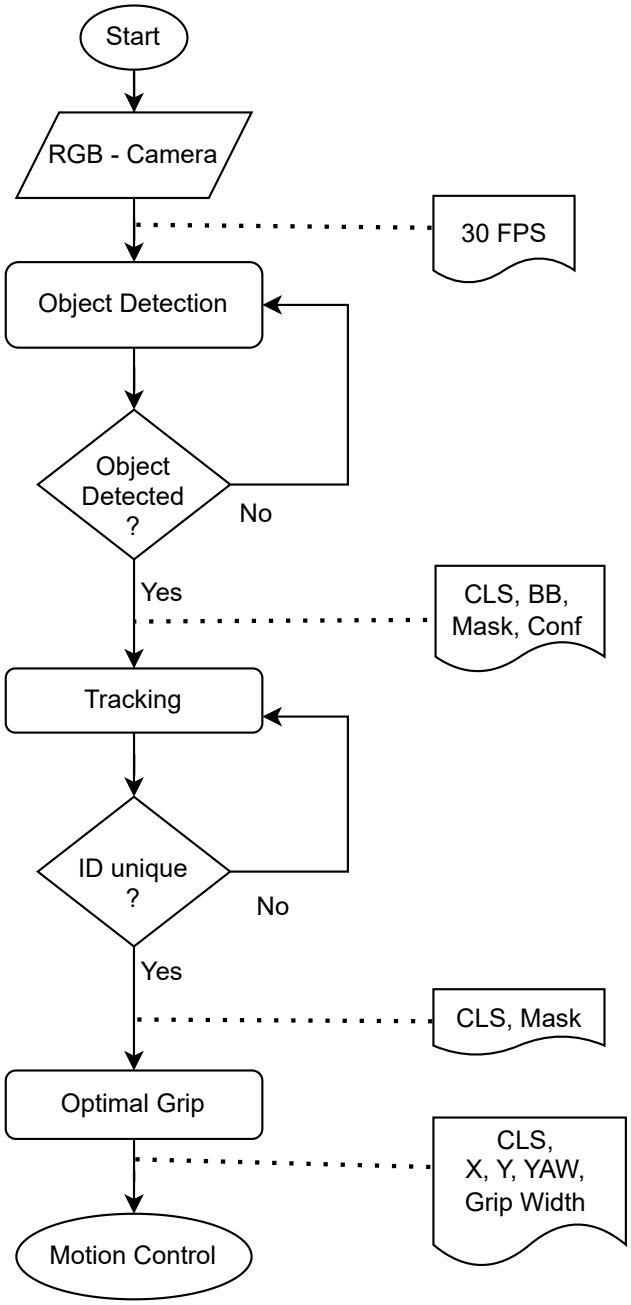


Figure 4.2: Flowchart of the perception pipeline

Chapter 5

Autonomous Sorting Facility

This chapter elaborates on the performed construction of an autonomous sorting facility, encompassing both physical and digital production. The focus is on creating 3D models, implementing inverse kinematics via Moveit2, and developing high-level motion control for a cobot interaction, based on real-time object detection. This chapter also includes description of how obstacles encountered implementing the physical small scale sorting facility has been overcome, in particular, obstacles related to the remote communication. For example, a reverse engineering of a gripper communicating via Modbus was performed. This obstacle was significant to overcome to reach the results of this project, and it is a good illustration of the complexities and insights gained from solving such interdisciplinary challenges. Lastly, the design of a modular node system using ROS is discussed, highlighting its crucial role in bridging physical and digital components within the autonomous sorting facility.

5.1 Collaborative Robot

The robotic manipulator used in this project is a UR10e from Universal Robot. It is a six-axis robot certified to work around humans due to having safety features to prevent and minimize the risk of human injury. The cobot has the ability to lift 12,5 kg, a reach of 1,3 meters, and a weight of 33.5kg. Additional information can be found in Appendix A. Joint working range and maximum velocity are critical to take into account when developing the motion control system, both with inverse kinematics and with reinforcement learning, especially when the movement is not repetitive and the goal pose is dependent on objects at the unknown position. The robot is built up with 6 joints, which are referred to by the names in Table 5.1. The base, also called shoulder pan joint in robotics, is the origo when giving reference point according to the robot, and wrist 3 is referred to as the end-effector if the gripper is not attached.

Table 5.1: UR10e hardware joint limitations Sheet

Joint names	**Working range**	**Maximum Speed**
Base	± 360°	± 120°/s
Shoulder	± 360°	± 120°/s
Elbow	± 360°	± 180°/s
Wrist 1	± 360°	± 180°/s
Wrist 2	± 360°	± 180°/s
Wrist 3	± 360°	± 180°/s

5.2 3D Modelling

A full prototype had to be built both physically and digitally. Given equipment and parts needed to be built and assembled into a complete facility for demonstration and testing. Since the perception only detects objects at the beginning of the conveyor belt, the assembly needs to be accurate and stable. Parts need to be produced and the model needs to be remodeled to the correct format so it can be implemented in the digital environments.

5.2.1 Design of Facility

The facility is designed for having all the main equipment connected to each other, strategically positioned. An analysis of both camera and robot placement has been made. The placement of the camera is explained in Section 4.3. The efficiency of the pick-and-place operation is highly dependent on the placement of the robot in relation to the conveyor belt. One of the critical factors to consider when positioning a robot is the avoidance of singularities. Singularities occur when a robot loses one or more degrees of freedom, which can result in unpredictable behavior and reduced precision.

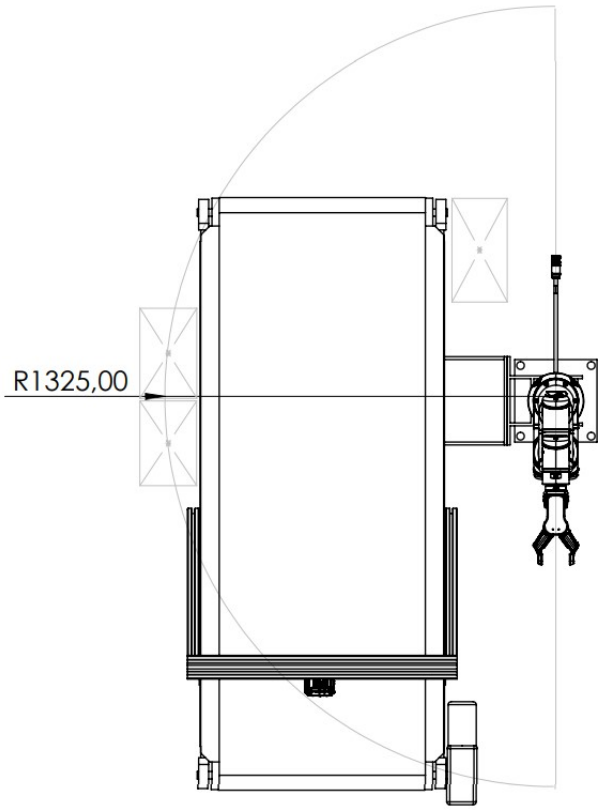


Figure 5.1: Working Range

Section 5.2.1 shows how a UR can conflict with singularities in certain planes. By optimizing the placement of the robot at the conveyor belt, taking into account singularities will maximize the workspace [11]. For the experimental setup, there is positioned three waste bins around the conveyor belt for sorting the objects, by placing the objects based on their respective class. The robot needs to cover the total surface from the camera mount and reach both sides of the conveyor belt for sorting objects. The measured work area is visualized in Figure 5.1. Every part used in this project is 3D modeled, either the models have been downloaded from the producer when available or they are modeled with the digital 3D CAD software SOLIDWORKS from scratch.

5.2.2 Physical Construction

After designing the facility, the construction process started. The camera mount is produced in aluminum profiles. The aluminum profiles are 3D modeled and analyzed on how to be constructed and ordered. Bill of Materials (BOM) is listed in Appendix Table B.1 together with production drawings. Aluminum profiles are chosen for the reason of their mechanical properties, adaptability to configuration changes and to be rebuilt for other projects. The attachment between the robot and conveyor belt is machined and welded based on the production drawings in Appendix C.

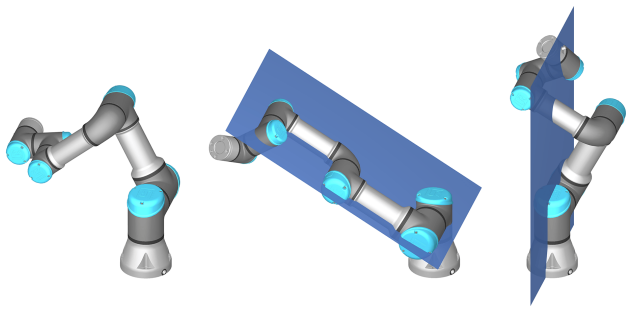


Figure 5.2: Singularities [11]

5.2.3 3D models for simulation

In the first step, a model is set up in SOLIDWORKS and then imported in stereolithography (STL) format. The configuration is adjusted based on the desired local coordinate system. Following this, the STL files are utilized to create Unified Robot Description Format (URDF) and Universal Scene Description (USD) models. The URDF models are used to build a digital twin environment that simulates the IK motion control system, which is visualized in the RViz. Similarly, a digital twin environment for the RL motion control system is constructed using USD models, visualized in Isaac Sim.

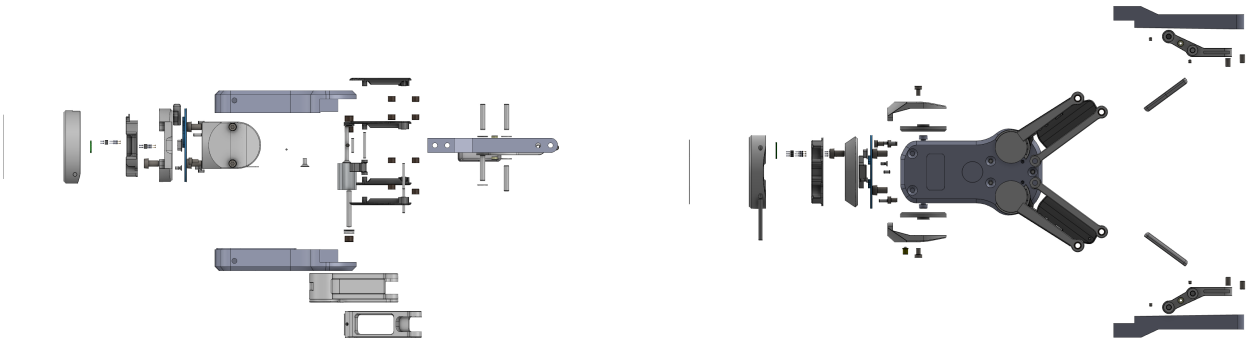
The Unified Robot Description Format (URDF), a widely-used XML format in robotics, is used to describe the robot's physical structure, including its joints and connections. These mesh files provide detailed descriptions of each joint. The type of each joint, whether fixed or revolute, is chosen based on the required motion between links. The term 'translation' refers to the distances (in the x, y, and z coordinates) from the origin of the parent link to the origin of the child link. These distances are specified in meters. The orientation of the child link relative to the parent link is defined using quaternions or Euler angles.

A gripper from OnRobot is used in this thesis for picking objects. It is an RG6, with a quick changer attached to the robot and the gripper as a device mount. The specifications are shown in Table 5.2.

Table 5.2: Specifications of the OnRobot RG6 Gripper

Specification	**Value**
Payload	6 kg
Max. Stroke	160 mm
Min. Force	25 N
Max. Force	120 N
Gripper Mass	1.25 kg
Number of Fingers	2
Energy Source	Electrical
Protection (IP)	IP54

A 3D model of the gripper was downloaded and disassembled digitally. This process facilitated the reassembly with the optimal joint configuration to correctly identify movable parts. An exploded view, presented in Figure 5.3, illustrates the assembly structure of the gripper. To accommodate objects up to 14 cm high, extenders were designed and 3D printed. These extenders, informed by measurements from the 3D model, enhance the reach of the gripper.



(a) Exploded View from the Side

(b) Exploded View from Above

Figure 5.3: OnRobot RG6 with Quick Changer and Extenders

5.2.4 Assembly

The completed prototype ended up at a total size of 2 meters in length, 1.4 meters wide, and 2.6 meters high Section 5.2.4.

Figure 5.4: 3D Model of Assembly

The physical construction can be seen in Section 5.2.4. It corresponds with the size measurement of the 3D model above. It has a solid construction, which is critical when the robot has to pick objects outside the FOV of applied perception.

Figure 5.5: Physical Setup at Mechatronic Innovation Lab

5.3 Motion of Control

This section elaborates on how the motion control system is set up with IK through MoveIt2. A detailed explanation of how an interface can be made for continuously receiving objects to pick with the vision system.

5.3.1 Polyscope

PolyScope is the operating system that powers Universal Robotics' e-Series, including the UR10e. It provides a graphical user interface (GUI) for robot programming and control. The PolyScope is only used to enable external control from a PC hosting the UR ROS driver and to set up safety limitations shown in table 5.3. The external control is enabled through activating a PolyScope extension (URCAP) UR GitHub page.

Table 5.3: UR10e Limitations

Specification	**Value**
Power	1000 W
Momentum	1300 mm
Stopping Time	1000 ms
Stopping Distance	2000 mm
Tool speed	5000 mm/s
Tool Force	250.0 N
Elbow speed	5000 mm/s
Elbow force	250.0 N

5.3.2 UR Driver for Joint Control

The ROS driver for UR provides an interface at a high-level for commanding the angular joint position of the robot. The driver supplies multiple ROS nodes that communicate with the UR robot controller via TCP/IP. The nodes publish and subscribe to ROS topics in order to control the robot's joints and receive sensor feedback. The driver controls the robot on low-level where it actuates each joint based on the given reference values. As most robot suppliers are starting to make drivers for their robots for remote control via ROS, a standardized system is followed for making the robot adaptive to manipulation of additional ROS libraries such as trajectory manipulation.

5.3.3 Motion Control with MoveIt2

The integration of MoveIt2 with the UR ROS driver provides a foundation for effective motion planning and execution. This integration facilitates the establishment of a communication interface between the UR ROS driver and the MoveIt2 framework, allowing for dynamic interaction and data exchange. In this framework, ROS nodes are set up by MoveIt2 to communicate with the UR ROS driver. These nodes publish and subscribe to ROS topics pertinent to motion planning and execution. The MoveIt2 framework utilizes an IK solver to ascertain joint configurations for achieving a desired end-effector pose. This calculation is crucial as it links the higher-level planning stage with the specific, low-level joint motions required to realize the intended pose. Since MoveIt2 is a trajectory calculator, a top-level controller with an interface is needed for handling continuous data from the vision system.

5.3.4 Path planning with OMPL

OMPL provides a variety of sampling-based motion planning algorithms. Sampling-based algorithms work by exploring the robot's configuration space, where each point represents a possible pose of the robot. These algorithms are particularly effective in high-dimensional spaces, and unlike grid-based algorithms, they don't rely on discretizing the space, which means they can better handle real-world complexity [14]. Different path planning algorithms is utilized for testing, but the RRTConnect planner is used in the experimental setup. RRT-Connect is a variant of the Rapidly-exploring Random Tree (RRT) algorithm, which is widely used for robot motion planning due to its effectiveness and comprehensiveness. This is implemented by selecting RRT in RViz, after running both UR driver and MoveIt2.

5.4 Interfacing Real-Time Object Detection

The `pyMoveIt2` is a basic Python interface for MoveIt2. This interface has been built upon to employ a ROS node that continuously handles new target pose data. The implementation has been configured for the UR10e. Chapter 4 details the output of the vision system, which is further subscribed to a node for processing and transmitting information to the implemented motion control interface. The motion control interface node receives position and pose estimations, queues the data, and transmits them through MoveIt2. The vision system provides coordinates in meters for translation and radians for yaw (ψ), representing the rotation of the wrist 3 joint. The interface converts position data into the desired pose in quaternions, representing the robotic arm's rotation in three-dimensional space. The equation that describes the quaternion conversion is defined in Equation (5.1):

$$q = w + xi + yj + zk \tag{5.1}$$

where w, x, y, and z are the components of the quaternion, and i, j, and k are the fundamental quaternion units. The quaternion, denoted as Q or q, is a four-dimensional vector consisting of (q_w, q_x, q_y, q_z) or equivalently (w, x, y, z). The mathematical equations used to convert the coordinates are Equations (5.2) to (5.5):

$$qw(\phi,\theta,\psi) = \cos\left(\frac{\phi}{2}\right)\cdot\cos\left(\frac{\theta}{2}\right)\cdot\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\cdot\sin\left(\frac{\theta}{2}\right)\cdot\sin\left(\frac{\psi}{2}\right) \tag{5.2}$$

$$qx(\phi,\theta,\psi) = \sin\left(\frac{\phi}{2}\right)\cdot\cos\left(\frac{\theta}{2}\right)\cdot\sin\left(\frac{\psi}{2}\right) - \cos\left(\frac{\phi}{2}\right)\cdot\sin\left(\frac{\theta}{2}\right)\cdot\cos\left(\frac{\psi}{2}\right) \tag{5.3}$$

$$qy(\phi,\theta,\psi) = \cos\left(\frac{\phi}{2}\right)\cdot\sin\left(\frac{\theta}{2}\right)\cdot\sin\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\cdot\cos\left(\frac{\theta}{2}\right)\cdot\cos\left(\frac{\psi}{2}\right) \tag{5.4}$$

$$qz(\phi,\theta,\psi) = \cos\left(\frac{\phi}{2}\right)\cdot\cos\left(\frac{\theta}{2}\right)\cdot\sin\left(\frac{\psi}{2}\right) - \sin\left(\frac{\phi}{2}\right)\cdot\sin\left(\frac{\theta}{2}\right)\cdot\cos\left(\frac{\psi}{2}\right) \tag{5.5}$$

The primary mechanism for the pick-and-place operation involves a callback, which initiates the process of moving the end-effector to the object at the *object_position*. It then guides the robot arm to *pick_position*, thereafter to *release_position*, before returning to the *home_position*. During the interval between the *object_position* and the *pick_position*,

the gripper is set to close. The control of the gripper is managed by a separate node, which is further elaborated in the following Section 5.5. These positions are displayed in Table 5.4.

Table 5.4: End-effector Position

Position Name	**X Coordinate**	**Y Coordinate**	**Z Coordinate**
Object Position	goal_x	goal_y	goal_z
Pick Position	goal_x	goal_y	goal_z - 0.12
Lift Position	goal_x	goal_y	goal_z + 0.02
Home Position	0.75	0.00	goal_z + 0.05

The vision system is passing the class name of the detected object. In Table 5.5, an overview is presented with each object used in the experimental setup. The coordinates of *release_position* based on class are listed in the table.

Table 5.5: Place coordinates for object type, with origo in robot base.

Label	**X-Coordinate**	**Y-Coordinate**	**Z-Coordinate**
PP-5 Lid	1.2	0.2	0.2
PP-5 Blue Box	0.13	-0.57	0.2
PP-5 Filter Tip Box	1.2	-0.20	0.2
PP Vitrolife	1.2	-0.20	0.2

Before the node could be running continuously and update the input data, a setup of mutual exclusion is implemented for handling the pose estimation from the vision system. Trajectory planning is executed via a FIFO queue data structure, which ensures each received goal pose is processed sequentially. The Python interface for MoveIt2 requires disabling the restrictions for callbacks to execute them in any way it sees fit. The callbacks are then processed in parallel with each other, and the same callbacks will run in parallel. The thread lock mechanism and a thread-safe queue facilitate the sequential execution of goal pose callbacks, preventing concurrent movement instructions that might compromise the robot's operation.

The corresponding traveling direction to the object is calculated and subtracted from the object's y-coordinate, which is the traveling direction, effectively adjusting the target position to account for the conveyor belt's movement. Ultimately, the entire process leverages MoveIt2's framework to command the robot to execute autonomous pick-and-place operations based on real-time positional data from the vision system. This is a complex orchestration of efficient control mechanisms.

5.5 Gripper with Remote Control

This section provides a comprehensive analysis of how an OnRobot gripper is reverse-engineered for accessing remote control via a PC to actuate the interaction of the motion control system developed in this project. This had to be done due to a lack of detailed information on how the gripper can be controlled directly without additional equipment that was not available. In the initial phase, current and voltage were measured, and interpreted while the gripper was activated by the robot, therefore logged and evaluated. To replicate the communication process, a protocol was established, and software was made to facilitate communication with the node system. This served to optimize the interaction between the system's various components, allowing for efficient and modular operation. An electrical

circuit was designed and implemented into an embedded system accessing the remote control.

5.5.1 Reverse-Engineering of Onrobot RG6

The gripper is connected to the robot with an 8-pin SAC cable. In PolyScope, the tool equipment configuration is set to a single 24V as output. Figure 5.6 shows the wiring out of wrist 3 and to the tool mount. It is more than one option in rows 1, 2, and 6, the reason for setting up other programs, and utilizing additional equipment needing these as input. For this case; RS485+, RS485- and GND are hijacked for sending modbus rtu communication to the gripper.

Nr	Port
1	RS485+ or AI2
2	RS485- or AI3
3	DI
4	DI
5	24V
6	24V or DO
7	DO
8	GND

Figure 5.6: Signal Description

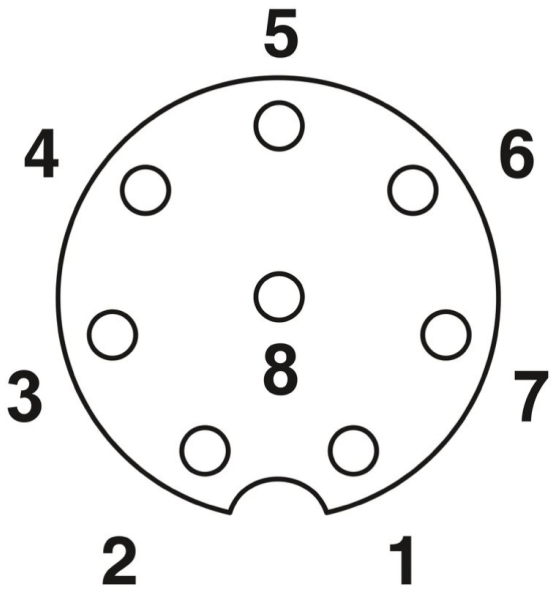


Figure 5.7: 8-Pin SAC cable

5.5.2 Reverse-Engineering of Modbus RTU Communication

The focus of this work was the control of the OnRobot RG6 gripper, achieved through reverse-engineered Modbus RTU communication. The process included reading and writing operations to register addresses, and a CRC (Cyclic Redundancy Check) checksum check implementation, to ensure data integrity [22]. A visualization of the Modbus Protocol description is presented in figure Figure 5.8.

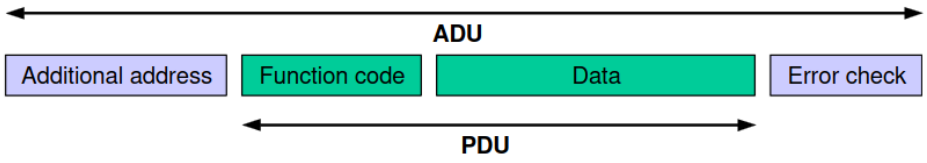


Figure 5.8: Protocol Data Unit and Application Data Unit

To integrate the gripper to the motion system, a ROS package was created, specifically designed to publish status messages while listening to a topic for controlling the gripper. Limited documentation from OnRobot regarding remote gripper control necessitated this reverse-engineering approach. It was observed that OnRobot supports the use of their proprietary Compute Box, an additional device sold separately, for directing all communication messages to the appropriate device [20]. Each device, including the Compute Box and the

gripper, is characterized by unique device addresses that direct communication.

Interestingly, the Compute Box enables a Modbus TCP communication interface for itself and connected OnRobot devices [22]. However, details regarding how the Compute Box facilitates remote communication remain proprietary and are not publicly available. While the Universal Robot's Polyscope interface can support Modbus TCP communication, it cannot directly access OnRobot devices using the device addresses documented for the Compute Box interface.

To overcome these constraints, the cables from the gripper to the Robot were split and an older datasheet was used to map the connected pins [19], as shown in Figure 5.7 on page 43. An updated data pin sheet had to be created due to discrepancies between the data pin sheet and the gripper version. This was achieved by conducting voltage and current measurements. Subsequently, the communication cables were connected to the PC via an RS485 to USB converter, setting up a serial monitor. This configuration enabled the monitoring of Modbus RTU communication and the reading of values directly from the Fieldbus [1].

Configuring the correct baud rate, byte size, parity, and stop bits to reflect the configurations for the gripper's microcontroller is vital for successful Modbus communication. The baud rate represents a value with a unit of bits per second, while the byte size can range from 5 to 9 bits. Parity—defined as Even, Odd, or None—provides a minimal error check for every byte in a message, and Stop Bits (1 or 2 bits) indicate the end of the data packet by transmitting from low to high voltage.

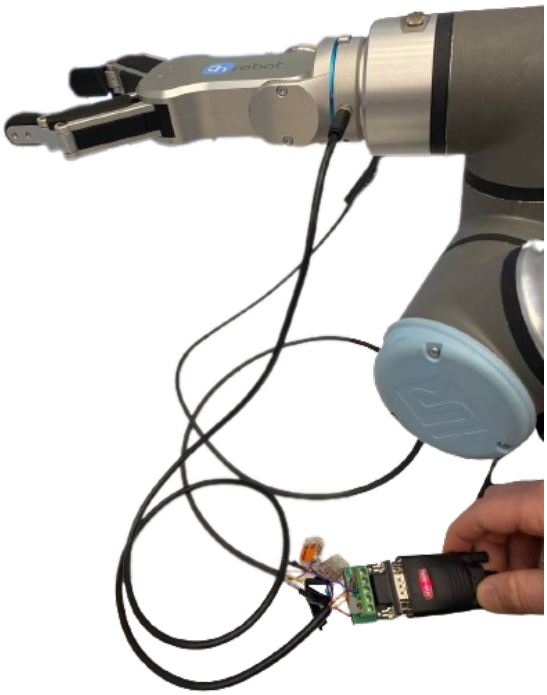


Figure 5.9: communication reading

The scarcity of publicly available documentation regarding these communication configurations presented a challenge. A solution was developed by setting up serial communication based on the Modbus Application Protocol [22]. Minimal necessary documentation provided by OnRobot's Compute Box for Modbus TCP communication was beneficial to this process. It detailed device addresses, register addresses, allowed values, and function codes for the register addresses [21]. Further insight was drawn from the Modbus Application Protocol documentation, specifying message formation for a particular function code, data integrity checks, CRC checksum inclusion in all messages, and the use of big-Endian byte order for addresses and data items [1].

5.5.3 Embedded System for Remote Control of Gripper

After being able to control the gripper through the USB converter directly, an embedded system was designed and implemented to run the remote control interface. A SAC-8P

cable links the robotic system and the gripper, providing the route for the exchange of control signals. To divert this communication, the RS485 cables within the SAC-8P were strategically tapped and then interfaced with a Jetson Nano D.1a. The Jetson Nano is a single board computer, used for hosting the developed program package for actuating the gripper, connected to the same network as the computers. The Jetson Nano is then integrated into the overall node network, allowing it to act as an intermediary between the robot and gripper, thereby gaining the ability to dictate their interactions. The power source for the Jetson Nano is taken from the wrist 3 via a LM2596 buck converter D.1b. Given the system's power requirements, the buck converter steps down the voltage from 24 volts to a more suitable 5 volts, ensuring that the Jetson Nano operates optimally without any risk of power-related damage. The finished electrical diagram for the embedded system is shown in Figure 5.10.

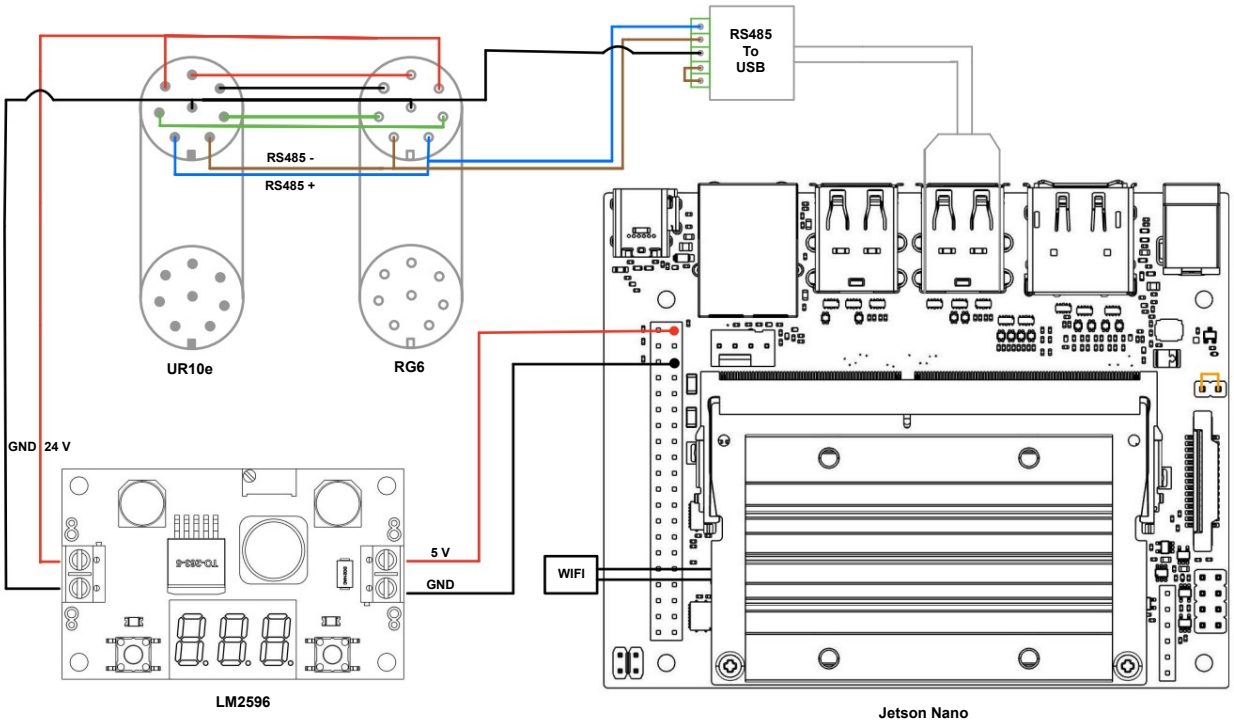
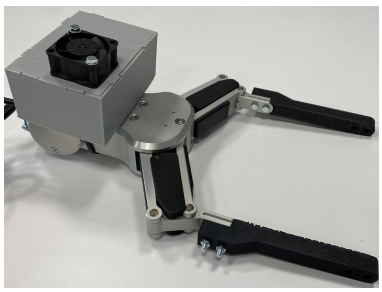
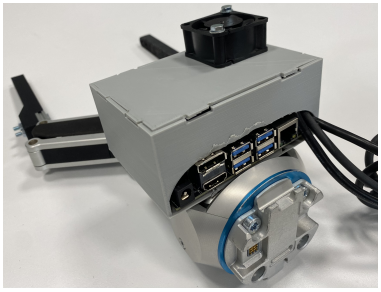


Figure 5.10: Embedded System for Gripper Control

A casing for these components wired together is 3D printed in two separate parts shown in Figure 5.11. Additionally, a fan is mounted on top to cool down the electrical components.



(a)



(b)

Figure 5.11: Control Box on Gripper

5.6 Simulation in Digital Environment

In the development phase, the UR driver and the physical robot were compared with a simple static pick-and-place operation. The accuracy was working well, so then, most of the testing was done by a twin environment setup. The UR driver has a finished model setup for RViz, and the rest of the environment was built. The UR10e robot is composed of several linked joints, which are defined in the Unified Robot Description Format (URDF) file associated with the driver. In ROS, each joint's position and orientation are defined by transformation frames or TFs. These TFs allow ROS to understand the position of each joint and link in the robot relative to each other. By reading the joint and having a digital clone in simulation, a ROS function called TF listener is used for knowing the space coordinates of each joint without the need for kinematic calculations. The coordinates of each link and configuration are visualized in Figure 5.12.

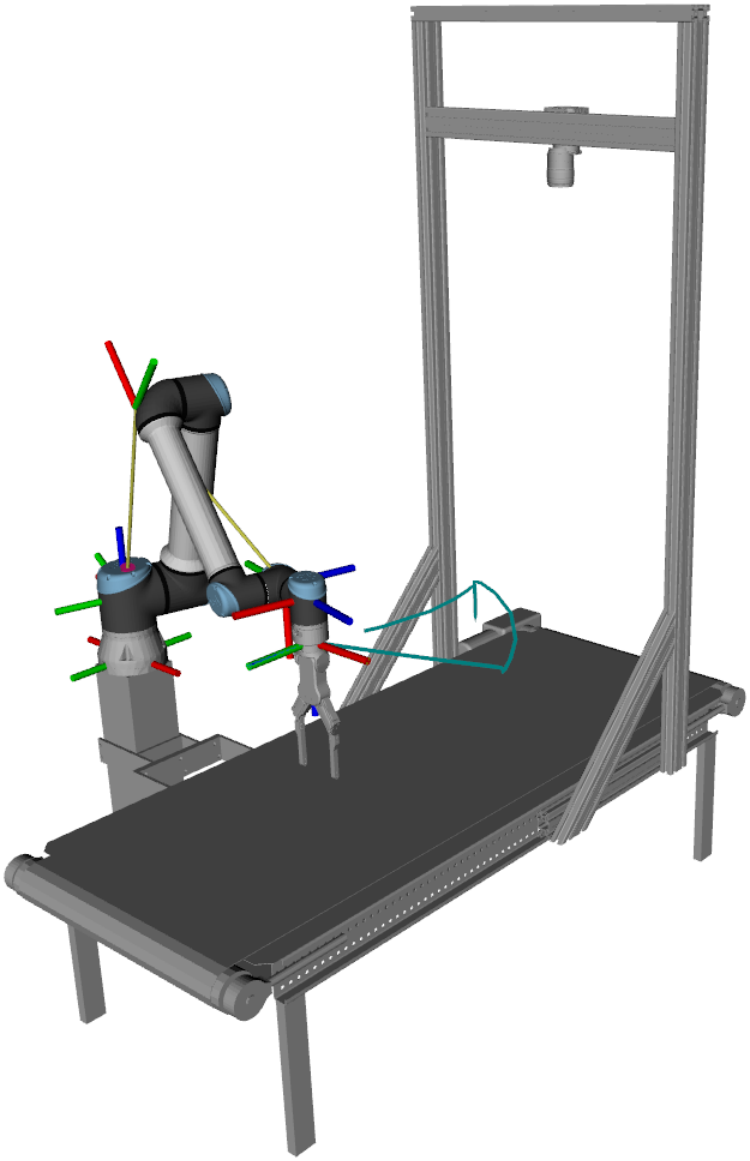


Figure 5.12: Tool-point path and TF to robot in RViz

After setting up the interface to the vision system, a node for publishing the objects position to the simulated environment was made as a marker. RViz is not a physics simulator, the node was setup by using ROS internal clock for making it move 10.5 centimeters per seconds. For developing the pick-and-place motion system, only a green marker is spawned based on

the physical target position as a rectangular, where the short side presenting optimal rotation to end-effector, and shown in Appendix E.

5.7 Nodes and communication

ROS has different function such as rqt graphing. This tool visualises how the hole system, with activates nodes and communication flow.

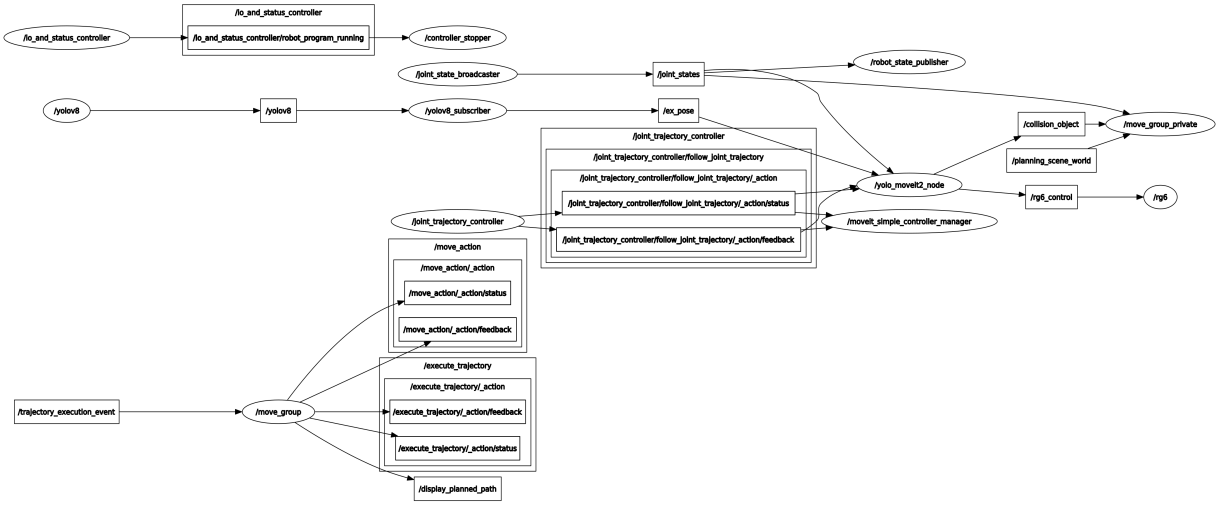


Figure 5.13: Node System with IK motion Control

Chapter 6

Reinforcement Learning in Simulation for Controlling a Physical Robot

This chapter describes the use of Reinforcement Learning (RL) to optimize a pick-and-place operation in an industrial setting where the physical environment operates without perception. Instead, a fully observable digital twin controls the motion of the physical robot. The RL policy, which guides the robot's actions, is only trained in the simulation environment, and the objects are randomly oriented during this training process to expose the policy to a wide range of possible scenarios.

In the sim-to-real phase, the objects in the digital twin environment spawn based on the perception pipeline, described in Chapter 4 and Section 5.4. This process uses the information to mimic the state of the real-world environment in the digital twin. The sim-to-real approach replicates the joint movements in simulation to the physical robot, mirroring the actions determined by the RL policy in the real world. This differs from other approaches that may directly deploy an RL model in the physical environment after training in a simulation.

6.1 Building a Digital Twin in Isaac Sim

Building a digital twin in Isaac Sim involves creating a virtual replica of a physical system for simulation and prediction. The process includes:

1. **3D Modeling**: A 3D model of the physical system is created using a CAD (Computer-Aided Design) tool such as SolidWorks, AutoCAD, or other 3D modeling software. The model should replicate the physical properties and characteristics of the system. The Isaac Sim integration with Omniverse, the 3D modeling can also be done here.

2. **Conversion to USD**: Isaac Sim uses the Universal Scene Description (USD) format for its 3D models. Once the 3D model is created, it needs to be converted into the USD format.

3. **Import into Isaac Sim**: The USD model can be imported into Isaac Sim, where users can manipulate the model, adjust its properties, and set up the environment for simulation.

4. **Physics and Simulation**: Isaac Sim uses NVIDIA's PhysX engine for physics simulation, which allows for realistic interactions between objects in the environment.

5. **Reinforcement Learning Training**: With the digital twin set up in Isaac Sim, reinforcement learning algorithms can be used to train agents to perform tasks within

this environment. The trained models can then be used for prediction and control in the real-world system.

6.2 Gym Environment

The development and deployment of RL techniques for real-world applications have been substantially facilitated by the implementation of simulated environments. Specifically, the RL Gym Environment, constructed around the 'digital twin' concept in Isaac Sim, offers a platform for RL models to interact with an environment that is reflective of a state-of-the-art physics simulator. The design and structure of the gym environment are purposefully aligned with real-world physics and object placements, enabling the RL models to be exclusively trained in simulation and their performance subsequently evaluated without necessitating training on the real-world robot.

Robotic Operating System (ROS) is adopted as the communication protocol for the RL gym environment to interface with the real-world robot, ensuring seamless data interchange. An in-depth discussion about this ROS-based sim-to-real communication can be found in the sim-to-real subsection of this work.

The gym environment allows the RL agent to engage with each robot joint by sending angular position commands. This enables a fine-grained control over the robot's actions within the simulated environment. During the sim-to-real operation, the RL model continuously communicates the current angular positions of the simulated model to the real-world robot, ensuring that the physical system accurately replicates the movements of the virtual counterpart. However, it is important to note that due to this communication process, the simulated joint positions can be slightly ahead of the real-world model.

The RL Gym Environment, thus, presents a well-structured and realistic platform for training RL models. It aids in circumventing potential risks and costs associated with direct real-world training while ensuring an accurate reflection of physical laws and dynamics in the virtual learning space.

The training of a RL model typically needs to be trained for many hours or days for a RL task with this kind of complexity. The implementation of parallel processing of cloned environments with unique environment properties based on environment state makes it possible to train a RL model for a complex task within a couple of hours. The gym environment is implemented to be run visually or headlessly, where the visual training runs are used to evaluate the model based on simulated behaviour. A visual run in this project has been decided to default at 512 environments as seen in Figure 6.1, to not cap the main memory. Headlessly the number of environments is not bottlenecked by the main memory, and have been decided to run with 4096 parallel environments. Headlessly the bottleneck for adding more environments is trying to handle too many parallel processes at once, and by adding more environments it is needed to allocate more memory to the process. The current gpu configurations for the PhysX engine is as seen in Table 6.1.

6.3 Reinforcement Learning Implementation

In this section, we present the implementation of the RL algorithm for controlling the robotic arm in the Isaac Sim environment. The goal of the robot is to perform three tasks sequentially, switching between tasks upon successful completion. We utilize PyTorch for tensor operations and define a reward function to guide the agent's learning.

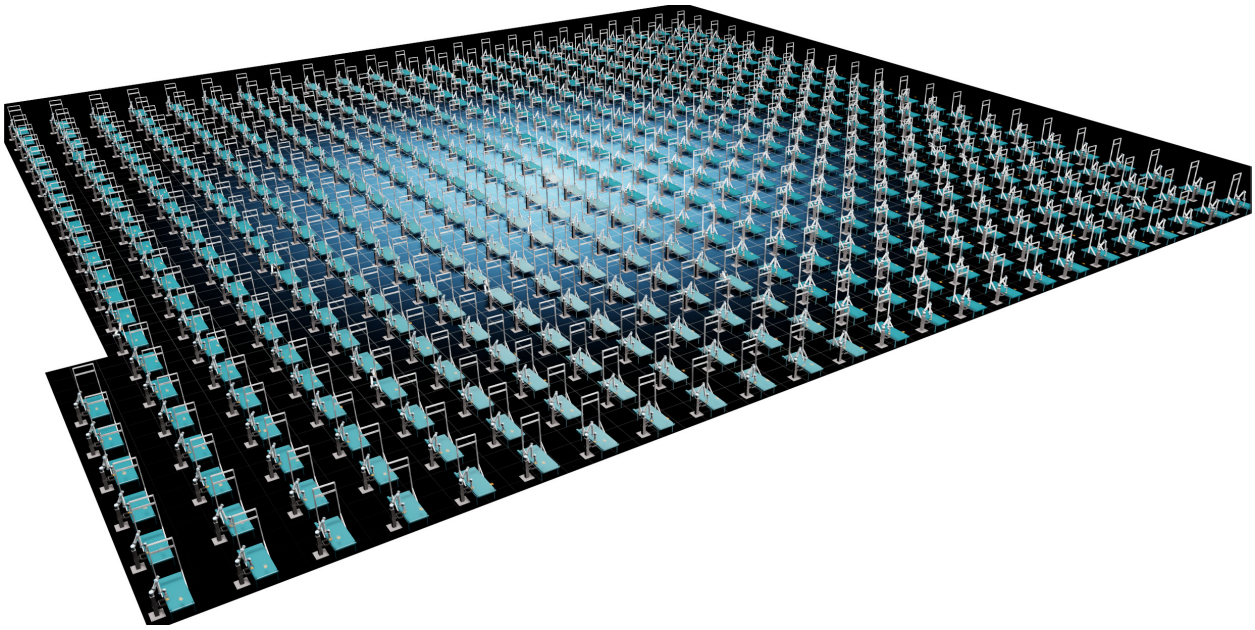


Figure 6.1: All environments

PhysX GPU Buffer Configurations	Value
gpu_max_rigid_contact_count	524288
gpu_max_rigid_patch_count	33554432
gpu_found_lost_pairs_capacity	19771
gpu_found_lost_aggregate_pairs_capacity	524288
gpu_total_aggregate_pairs_capacity	1048576
gpu_max_soft_body_contacts	1048576
gpu_max_particle_contacts	1048576
gpu_heap_capacity	33554432
gpu_temp_buffer_capacity	16777216
gpu_max_num_partitions	8

Table 6.1: PhysX GPU Buffer Configurations

6.3.1 Reinforcement Learning Observation and Output

The RL inputs an observation buffer for i_{th} environments. The parameters passed as the observation state is the 6 joint positions, 6 joint velocities, target speed, target pose defined by position and orientation, orientation distance from target alignment.

6.3.2 Computing Environment Observations for Reinforcement Learning

The efficacy of a RL (RL) model is heavily contingent upon the breadth and precision of the observations it receives from the environment. In our specific RL implementation, we leverage a holistic observation approach, considering a myriad of factors that contribute to the real-time state of the system. These observations form a multi-dimensional input array for the RL model, providing pertinent information about the current state of the environment, such as the state of the robot arm and the characteristics of the goal it is trying to achieve. The following is the observation space:

- **Robotic Arm State:** The state of the robot arm is primarily determined by the positions and velocities of its degrees of freedom (DoFs). The arm's DoF positions are normalized and stored in the observation buffer to retain spatial consistency across different episodes and environments, which substantially aids the learning process of the RL model.

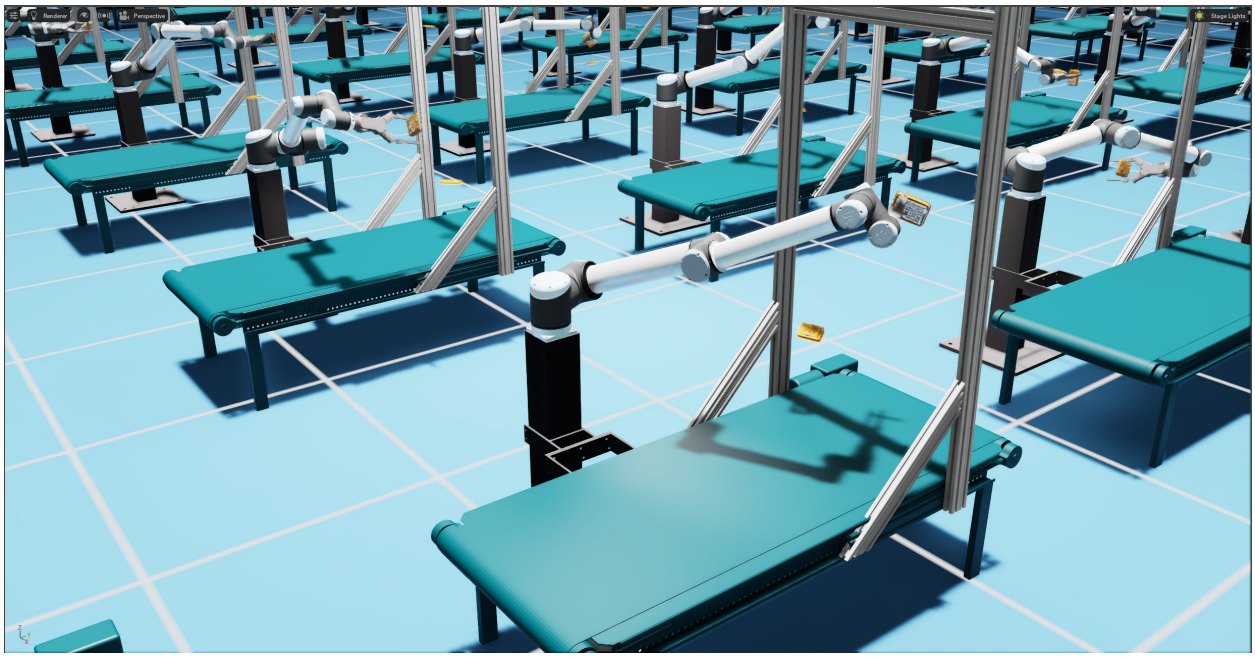


Figure 6.2: training

- **Target Velocity:** The current speed of the target position is also a critical aspect of the state of the system. This information assists the RL model in predicting the motion of the target and planning its actions accordingly.

- **Goal Position:** The goal position represents the desired destination of the arm's end effector. By providing this to the RL model, the agent can compute the necessary movements to reach the desired state.

- **Goal and Object Orientation:** The desired rotation of the arm is encapsulated within the goal rotation quaternion. In addition to the goal rotation, the quaternion representing the required rotation to align the object with the goal is also included. This data facilitates the RL model in understanding the required rotational adjustments to accomplish the task.

- **Recent Actions:** The most recent actions taken by the agent also form a part of the observations. This historical information enables the RL model to assess the consequences of its actions, aiding in the iterative improvement of its policy.

This observation method serves as a robust input strategy for the RL model, providing a holistic snapshot of the environment's current state. It combines diverse elements, such as the robot's state, target velocity, goal characteristics, and historical actions, creating a potent informational basis that empowers the RL model to learn effectively and devise optimal policies.

6.3.3 Reward Function and Task Control

The reward function, compute_arm_reward, is designed to balance multiple objectives, such as minimizing the distance and orientation difference between the object and the target, regularizing actions, and penalizing failures. It takes into account various parameters such as the distance and rotation rewards, action penalty, success tolerance, reach goal bonus, fall distance, fall penalty, and maximum consecutive successes. The function computes rewards for each environment and updates the successes count based on the goal and orientation criteria.

When the agent reaches the success criteria for a task, the environment switches to the next task in the sequence. This is accomplished using the tensors task0_resets, task1_resets, and task2_resets, which are responsible for resetting the tasks based on the task IDs calculated from the current successes. The different tasks can be organised into different phases of the main goal, pick and place of objects into collection bags. The different phases can be described as follows:

1. **Hovering Phase**: This phase is characterized by the agent successfully positioning itself above the target object. The commencement of this phase can be mathematically represented as:

   Let D_g denote the Euclidean distance ($goal\_dist$) between the agent and the target object, and D_r represent the difference (rot_dist) in their respective orientations. A tolerance level T ($success\_tolerance$) is predefined. For an environment i, if $|D_{g_i}| \leq T$ and $|D_{r_i}| \leq T$, a binary switch $goal\_resets_i$ is activated (set to 1). Subsequently, $successes_i$ is incremented by $goal\_resets_i$.

   Next, let $task\_ids_i$ be the modulus of $successes_i$ after division by 3. If $goal\_resets_i$ is active and $task\_ids_i$ equals 0, another binary switch $task0\_resets_i$ is activated. The activation of $task0\_resets_i$ signifies the successful commencement of the Hovering Phase for the i-th environment. The phase is visualized in Figure 6.3.

2. **Pickup Phase**: This phase involves the agent moving downwards to pick up the object. The mathematical representation of this phase's commencement is similar to the Hovering Phase, albeit with a different condition. If $goal\_resets_i$ is active and $task\_ids_i$ equals 1, the binary switch $task1\_resets_i$ is activated, marking the start of the Pickup Phase for the i-th environment. The phase is visualized in Figure 6.4 on page 54.

3. **Collection Phase**: During this phase, the agent, having successfully picked up the object, begins moving towards the collection bag. The commencement of this phase is defined by the activation of $task2\_resets_i$, which occurs if $goal\_resets_i$ is active and $task\_ids_i$ equals 2. The phase is visualized in Figure 6.5 on page 55.

In this manner, the $task\_resets$ tensors function as binary switches that control the transitions between tasks in each parallel environment. The use of $torch.where()$ function and bitwise logical operations allow efficient and parallel computation of these conditions across all environments. This streamlined task management system enhances the RL process, enabling precise control over the learning stages and facilitating the efficient execution of complex manipulative actions in parallel environments.

In the RL Model, the reward function plays a crucial role in guiding the agent's learning process. The reward function is designed to encourage the agent to achieve the desired goal while maintaining efficiency and stability during the task. In this specific implementation, the reward function is composed of three main components: position distance, orientation alignment, and action regularization.

The position distance component evaluates the Euclidean distance between the object's current position and the target position. This distance, denoted as $goal\_dist$, is calculated using the $L2$ norm:

$$goal\_dist = \|object\_pos - target\_pos\|_2$$

The orientation alignment component assesses the similarity in orientation between the object's current rotation and the target rotation. To compute this, the difference between the

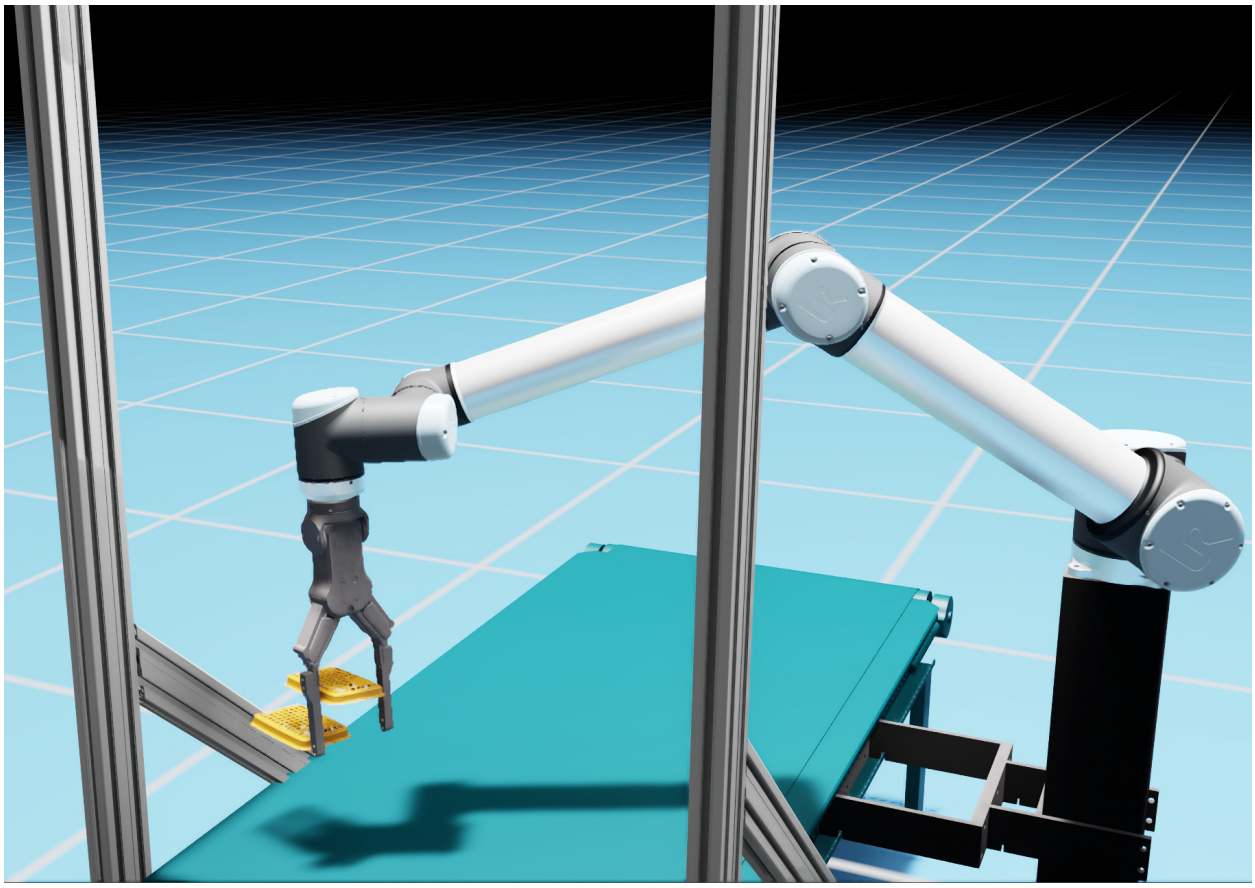


Figure 6.3: **Hovering Phase:** alignment of end effector pose and target pose represented by the physical objects in simulation.

two quaternions is first determined by multiplying the object's quaternion by the conjugate of the target's quaternion, as shown in the code:

$$quat\_diff = quat\_mul(object\_rot, quat\_conjugate(target\_rot))$$

The rotation distance, rot_dist, is then calculated using the arc sine function on the clamped $L2$ norm of the imaginary part of the quaternion difference:

$$rot\_dist = 2 \cdot \arcsin\left(\min\left(\|quat\_diff_{[1:4]}\|_2, 1\right)\right)$$

Next, the position distance reward $dist\_rew$ and the orientation alignment reward rot_rew are calculated by scaling the distance and rotation components with their respective reward scales:

$$dist\_rew = goal\_dist \cdot dist\_reward\_scale$$

$$rot\_rew = \frac{1}{|rot\_dist| + rot\_eps} \cdot rot\_reward\_scale$$

The action regularization component penalizes the agent for taking many actions to promote smooth and efficient movements. This penalty is computed as the sum of the squared action values:

$$action\_penalty = \sum(actions^2)$$

The total reward is then calculated as the sum of the three components, with the action penalty scaled by its respective penalty scale:

$$reward = dist\_rew + action\_penalty \cdot action\_penalty\_scale + rot\_rew$$

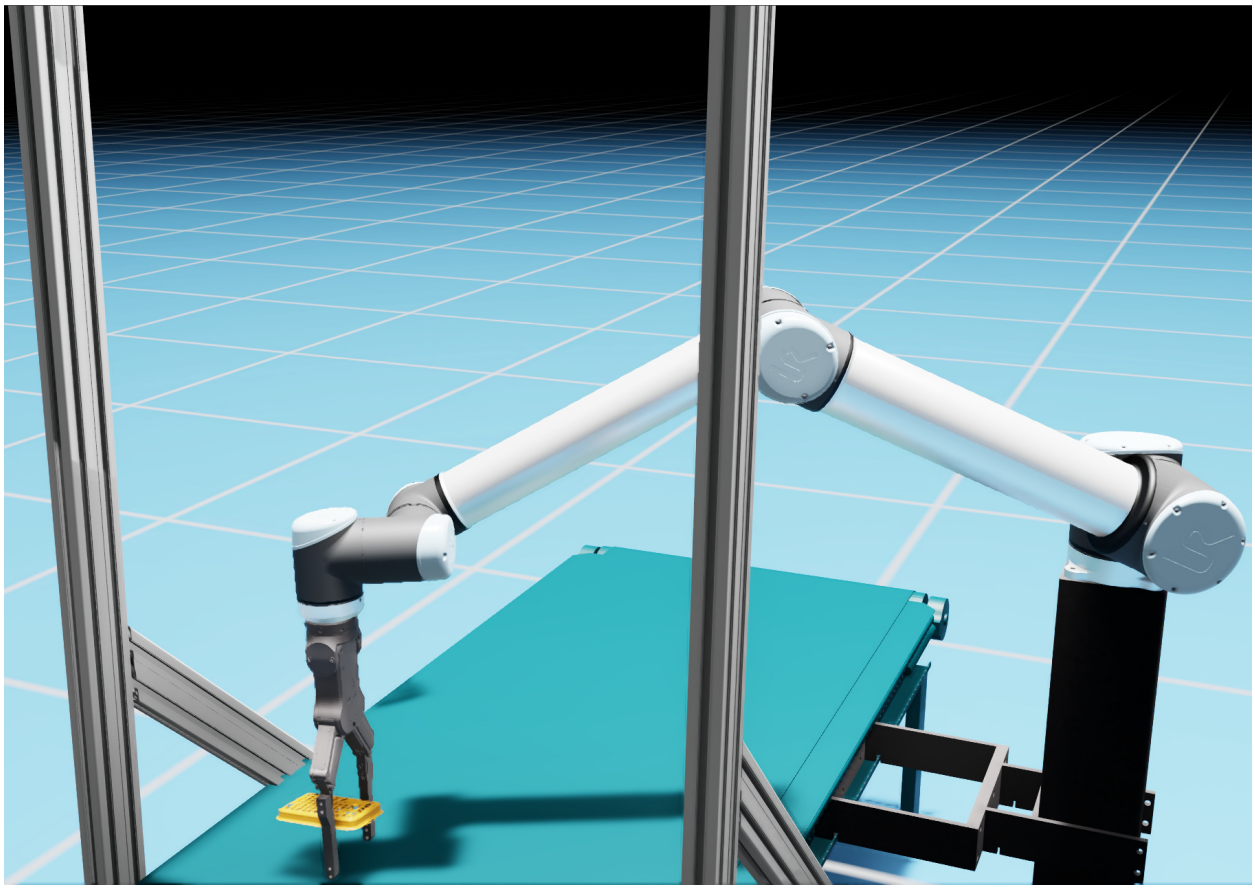


Figure 6.4: **Pickup Phase:** successfully aligned with target pose in the pickup phase.

To encourage the agent to reach the goal with the correct orientation, a success bonus is added to the reward if the position and rotation distances are within a specified tolerance:

$$reward = \begin{cases} reward + reach\_goal\_bonus, & \text{if } |goal\_dist| \leq success\_tolerance \text{ and} \\ & \quad |rot\_dist| \leq success\_tolerance \\ reward, & \text{otherwise} \end{cases}$$

By combining these components, the reward function effectively guides the agent to reach the desired goal position and orientation while promoting efficient and stable actions.

6.3.4 Sim2Real Deployment

The RealWorldUR10 class serves as the interface between the simulated environment in Isaac Sim and the physical robot using ROS. The class subscribes to topics that provide information on the robot's joint angles and publishes commands to control the robot. It also receives pose estimation data from the YOLO-based object detection system, which is used to update the goal position and orientation.

The *send_joint_pos* method converts the joint angles from the simulation environment to the real-world robot, ensuring that the angles are within the specified limits. It then updates the target joint positions accordingly.

The travel method compensates for the object's movement in the real world by adjusting the position based on the time elapsed since the pose estimation.

The *pub_task* method is an asynchronous method that regularly publishes joint trajectory

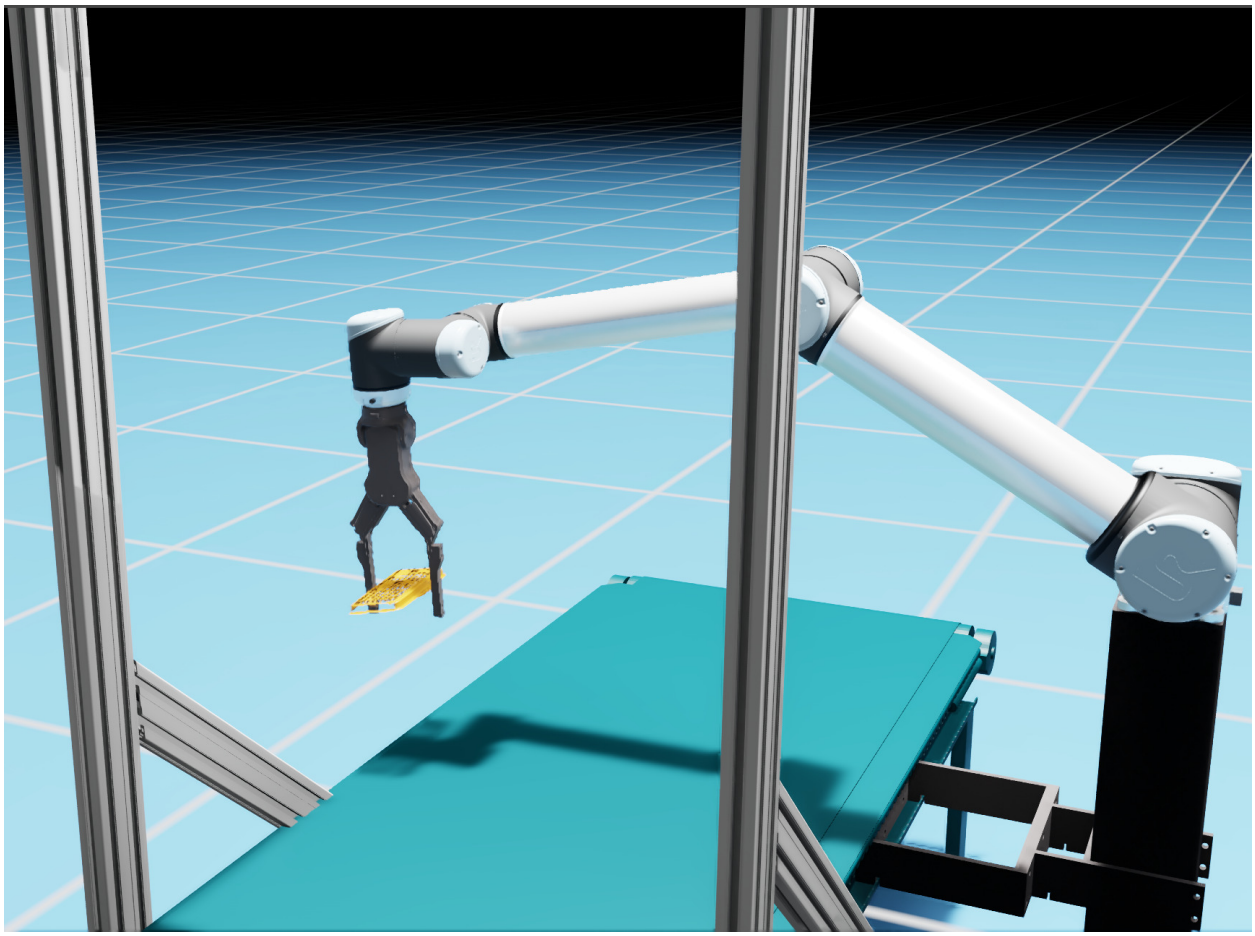


Figure 6.5: **Collection Phase:** successfully aligned with target pose in the collection phase.

commands to control the robot in the real world, taking into account the current and target joint positions.

6.4 Controlling a Real-World UR10 Robot Arm Based on Simulated Movements

The overall control architecture involves a ROS node that interfaces between the simulated and real-world UR10s. The control strategy is based on two primary components: a moving average filter and a trajectory duration control mechanism.

6.4.1 Weighted Moving Average Filter

In our system, the weighted moving average filter is utilized to smooth the target joint angles sent to the real-world UR10. The joint angles obtained from the simulation are passed through the weighted moving average filter before being sent to the real-world UR10. The filter is defined by the following equation:

$$cmd = pos \cdot (1 - \alpha) + target\_pos \cdot \alpha \tag{6.1}$$

where cmd is the filtered command sent to the real-world UR10, pos is the current joint position of the real-world UR10, $target\_pos$ is the target joint position obtained from the simulation, and α is the smoothing factor that determines the degree of smoothing.

This filter effectively creates a trade-off between the current position and the target position. When α is close to 0, the command is more influenced by the current position, creating a

lag behind the target position but smoothing out rapid changes. When α is close to 1, the command closely follows the target position, but is also more sensitive to rapid changes in the target position.

6.4.2 Trajectory Duration Control

The trajectory duration control mechanism is used to control the speed of the movement of the real-world UR10. The duration of each joint movement is calculated based on the distance to be moved and a maximum velocity limit, as defined by the following equation:

$$duration = \frac{|cmd - pos|}{max\_vel} \tag{6.2}$$

where *cmd* is the filtered command from the moving average filter, *pos* is the current joint position, and max_vel is the maximum velocity limit. The calculated *duration* is then compared with a minimum duration threshold, and the larger of the two is used as the *duration* for the joint movement. The minimum duration threshold is set to 0.1 seconds based on the publishing frequency in the ROS node at $10hz$. This mechanism ensures that the real-world UR10 moves at a controlled speed, avoiding abrupt movements.

6.5 Training the Proximal Policy Optimization Algorithm

In this study, the implementation of the PPO algorithm is guided by an intricate collection of hyperparameters and settings. The hyperparameters have been set based on documented performance for other Reinforcement Learning tasks that have inspired the RL implementation. The UR10 Reacher indie project [4] and examples given by NVIDIA Omniverse Isaac Sim Gym Envs Section 2.7 [18], along with the RL games for setting up PPO [3].

6.5.1 Algorithmic Settings

Several algorithmic settings have been set these include:

- **Algorithm and Model:** The $ppo\_continuous$ algorithm is chosen for this task, and the model is set as $continuous\_ppo\_logstd$. This pairing indicates the use of an Actor-Critic model with a continuous action space, which is fitting for the continuous state and action space in the given problem, and a logarithm standard deviation.
- **Network Structure:** The chosen network architecture is a Multi-Layer Perceptron (MLP) composed of layers with 256, 128, and 64 units respectively. Exponential Linear Unit (ELU) is used as the activation function.
- **Environment Settings:** A custom reward shaper with a scale value of 0.01 is employed, and 'gamma' and 'tau' parameters, significant in reinforcement learning algorithms, are set at 0.99 and 0.95 respectively.
- **Learning Parameters:** An adaptive learning rate schedule is chosen with a learning rate set at 5e-3. The maximum number of training epochs is set at 1500, with a $kl\_threshold$ of 0.008.
- **Training Settings:** A large minibatch size of 32768 is used to enhance robust learning. For each minibatch, the network undergoes 5 mini-epochs. Parameters such as $entropy\_coef$ is set to 0, $grad\_norm$ is set to 1, and e_clip is set to 0.2 during the learning phase.

Table 6.2 and Table 6.3 was the network and hyperparameter configurations set for training.

Network Parameters	Value
algo: name	ppo_continuous
model: name	continuous_ppo_logstd
network: name	actor_critic
network: separate	False
mlp: units	[256, 128, 64]
mlp: activation	elu
mlp: d2rl	False
mu_activation	None
sigma_activation	None

Table 6.2: Network Configuration Parameters

Hyperparameter	Value
mixed_precision	False
normalize_input	True
normalize_value	True
value_bootstrap	True
num_actors	4096
reward_shaper: scale_value	0.01
normalize_advantage	True
gamma	0.99
tau	0.95
learning_rate	5e-3
lr_schedule	adaptive
schedule_type	standard
kl_threshold	0.008
score_to_win	100000
max_epochs	1500
save_best_after	100
save_frequency	200
grad_norm	1.0
entropy_coef	0.0
truncate_grads	True
e_clip	0.2
horizon_length	64
minibatch_size	32768
mini_epochs	5
critic_coef	4
clip_value	True
seq_len	4
bounds_loss_coef	0.0001

Table 6.3: Hyperparameter Configuration for PPO

Chapter 7

Experiments and Results

Assessment of the final solution's effectiveness was carried out through the collection and analysis of data from computations, simulations, and physical experiments. The evaluation focused on the level of consistency and accuracy the PPO algorithm could provide in controlling a UR10e. This was done by comparing the angular joint displacement between the simulated joints and their physical counterparts. The performance of the object detection system and the reinforcement model were also examined.

7.1 Instance Segmentation Training Results

This section elucidates the training results of the AI model employed for instance segmentation.

7.1.1 Prediction vs Ground Truths

The Figure 7.2 on page 59 depicts a confusion matrix, offering insights into the prediction versus ground truth results. The outcomes are commendable for all classes, barring the PP-5 SarstedtFilter Tip Box (SFTB). The classes, apart from SFTB, exhibit a prediction accuracy range of 89% to 100% for the ultimate trained model. Only 3% of the PP Vitrolife class is erroneously predicted as background, while a larger portion, 8% of the PP-5 SFTB class, shares the same fate. These results signify the final model's commendable ability to accurately predict the classes, thereby reinforcing its reliability as a perception tool for autonomous pick-and-place operations. The prediction error is minimal, thereby making it academically insignificant for both case studies.

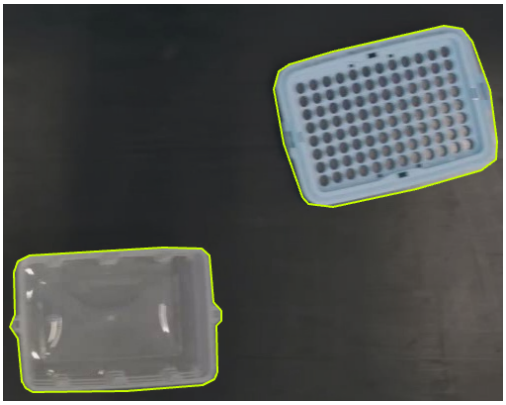


Figure 7.1: The major difference of an object within a class that affects performance for the particular class

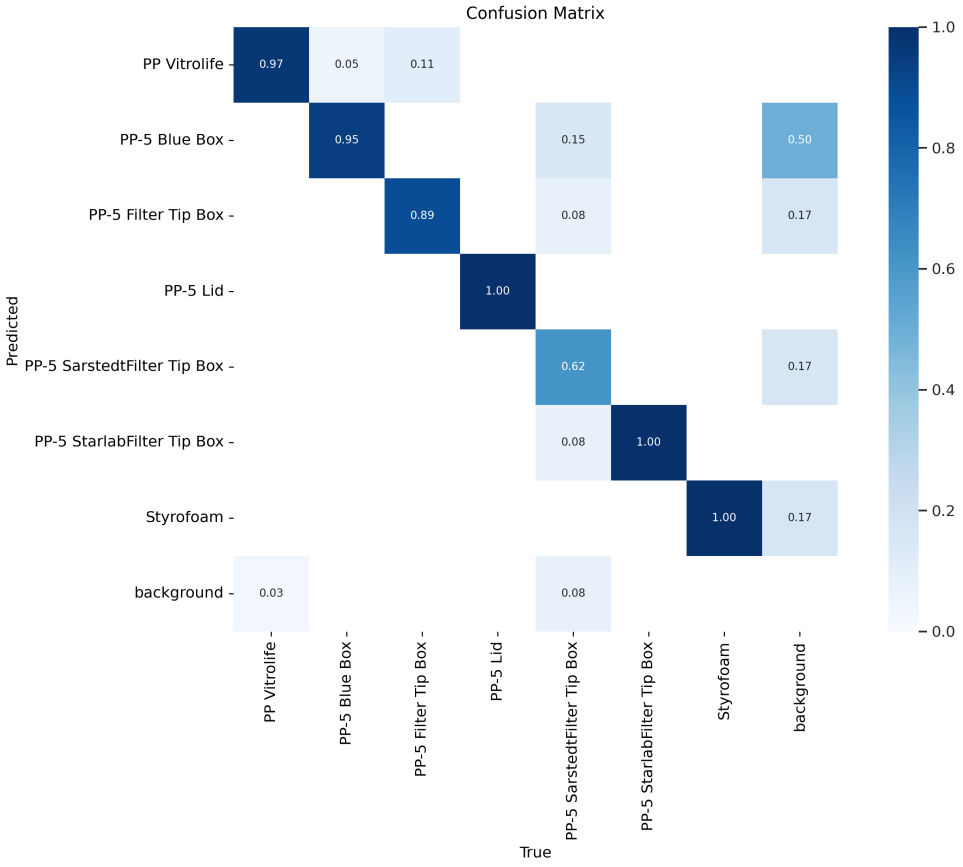


Figure 7.2: Confusion Matrix of Predictions vs. Ground Truths

7.1.2 Object Detection Metrics in Training

The following figures showcase the object detection metrics during training. The custom dataset is manipulated with augmentation before training for a more generalized model. This does affect the metrics results, but the model still achieves great results. Will look into F1 scores for both mask and bounding box predictions in Figure 7.3 and Figure 7.4. The last figure represents the training epochs and the progressive results on loss and all object detection metrics. Showcasing training converging Figure 7.5.

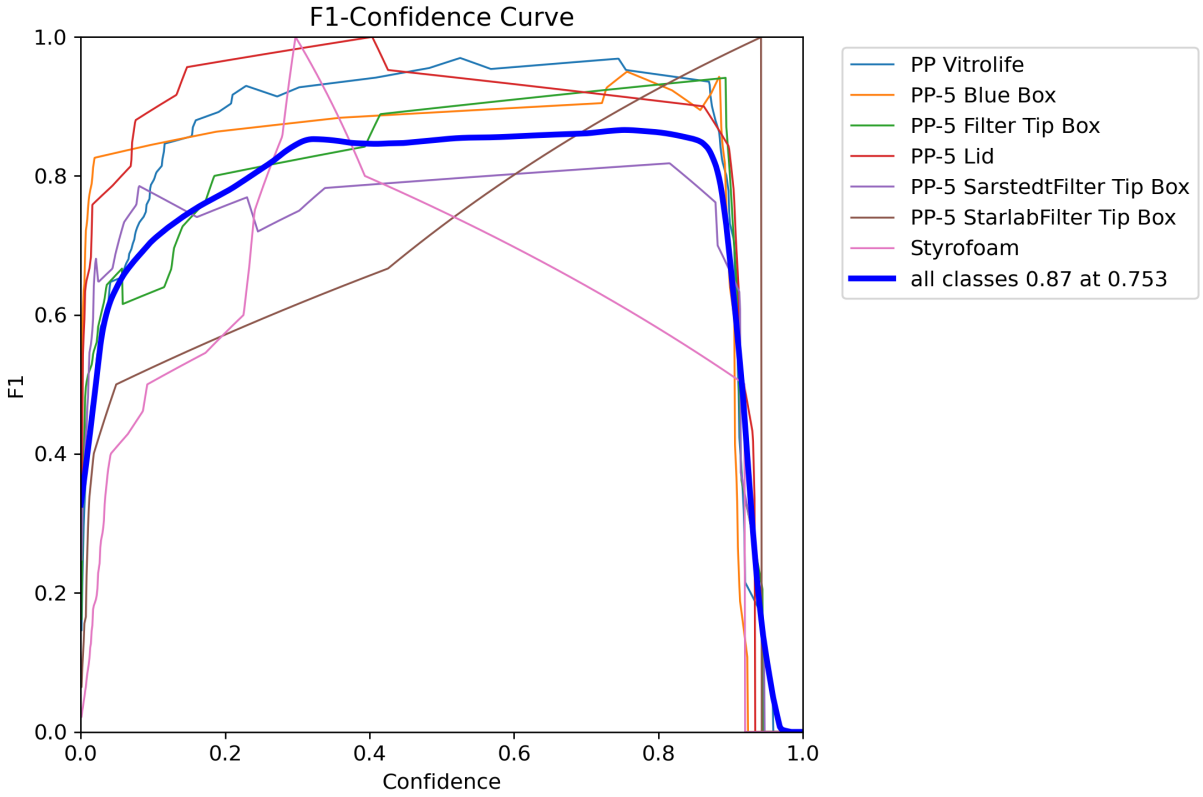


Figure 7.3: The F1 score for bounding box predictions. It achieves a great score except for Styrofoam, which is under-represented in the dataset.

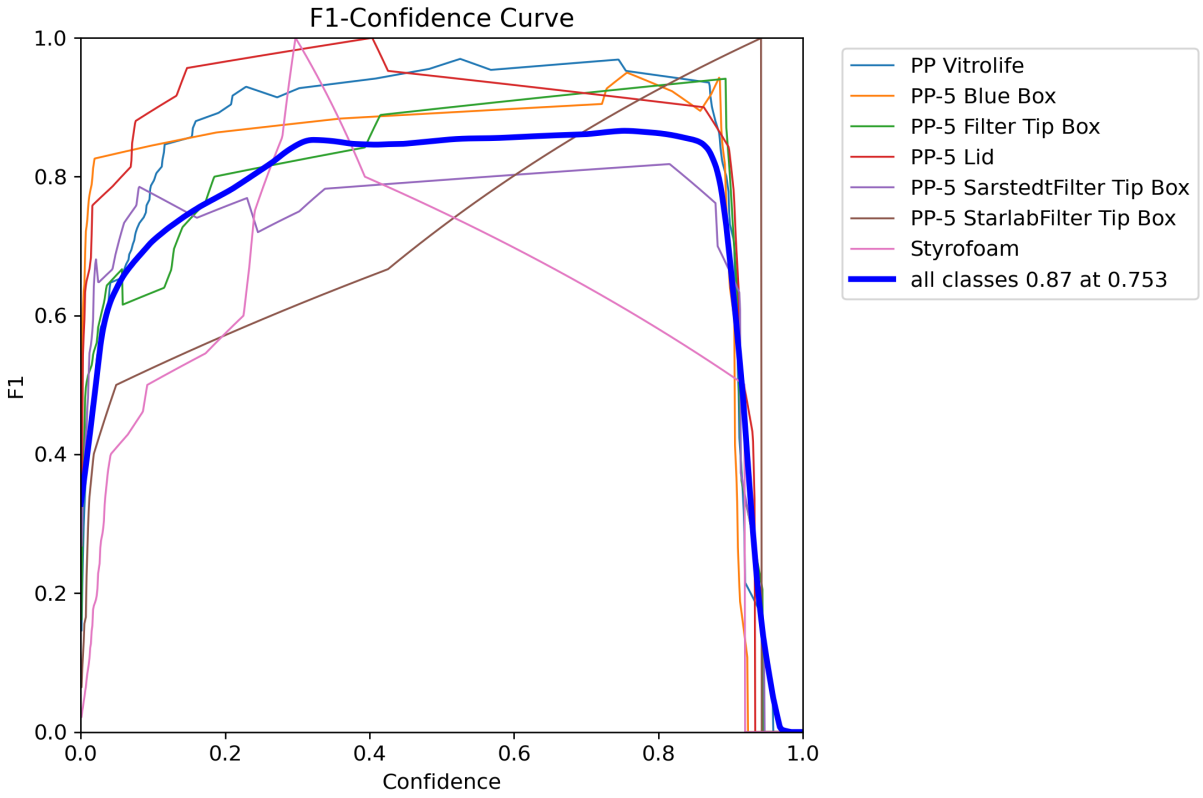


Figure 7.4: The F1 score for mask predictions.

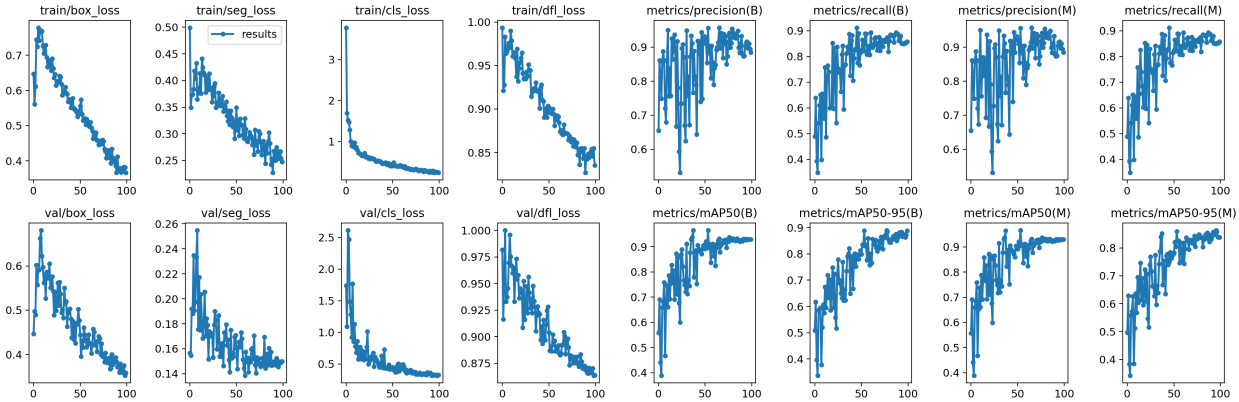


Figure 7.5: Training Results with Loss Plotting and Object Detection Metrics

7.2 Experimental Setup for Pick-And-Place

The results of the pick-and-place operation conducted on a physical test case are presented in the figures below. Different objects with different starting positions were used for testing. Wrist 3 was plotted through RViz with a TF listener, with the base as origin. For a more clear visualization in the presented plots, a dead time was added between each drop of the object for a full cycle of the whole process. The experimental setup was done over 40 seconds period.

7.2.1 IK Motion Control

Cycle time: 7,5 [s].
Home position in plot (x, y and z): 0.755, 0.84 and 0.2 [m].

The first Section 7.2.1 plots with the IK setup for controlling the physical robot. The system was able to sort objects with 42 centimeters of space in-between for being able to sort objects in continuous time. Table 7.1.

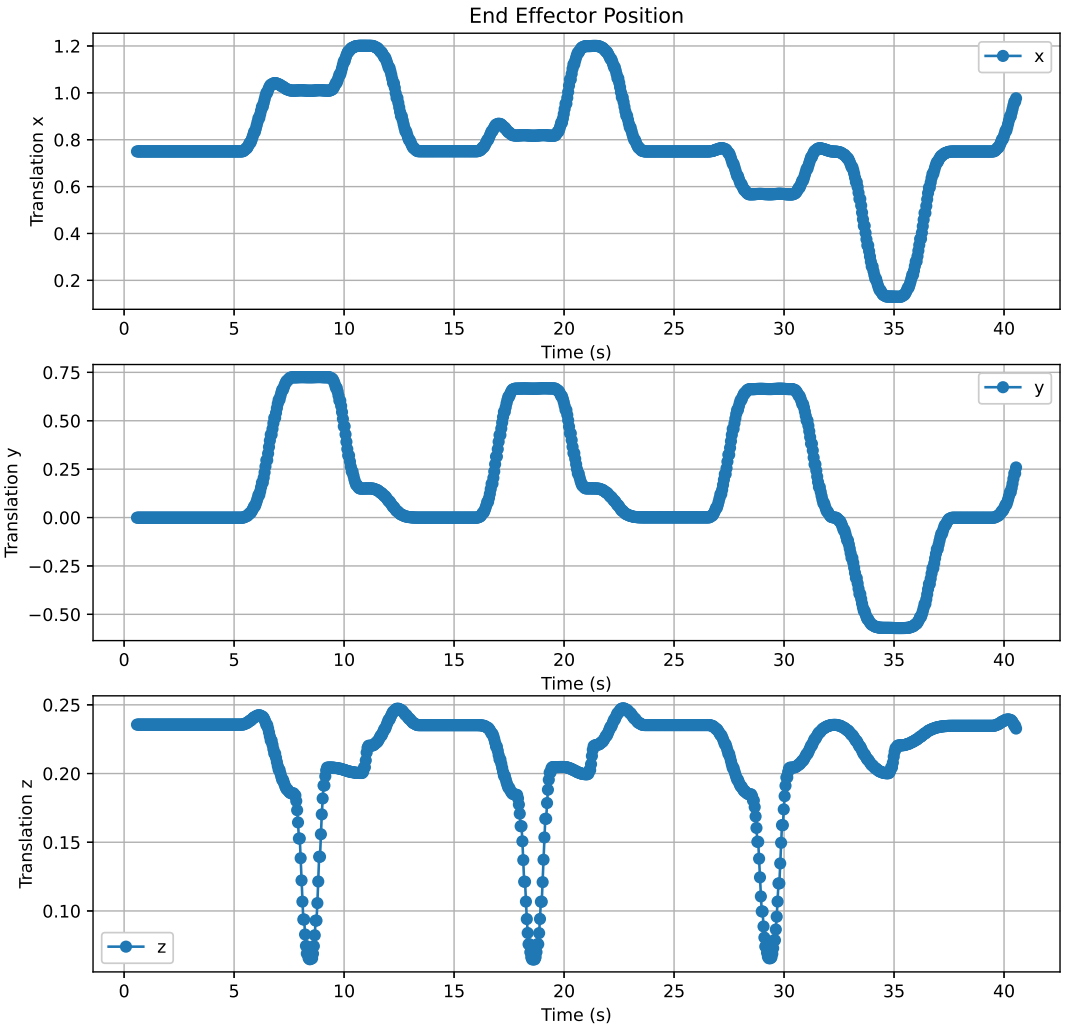


Figure 7.6: Tool-point path in physical space, with IK

Table 7.1: End-effector pose and time with IK.

	Object 1	**Object 2**	**Object 3**
Plastic Type	PP	PT	PE
Home Time [s]	0-5	6-13	14-27
Grasp Position (x, y, z) [m]	0.755, 0.84, 0.2	0.755, 0.84, 0.2	0.755, 0.84, 0.2
Grasp Time [s]	8	18	29
Place Position (x, y, z) [m]	1.2, 0.2, 0.2	1.2, 0.2, 0.2	0.13, -0.57, 0.2
Place Time [s]	11	17	35

7.2.2 RL Motion Control

Cycle time: 2.5 [s].

The RL motion control doesn't have a home position and only utilizes a standby position based on the endpoint of the last operation.

The Section 7.2.2 is the plot with the RL model as motion control for the physical robot. Table 7.1 lists the achieved pose and time stamp.

Table 7.2: End-Effector pose and time with RL

	Object 1	**Object 2**	**Object 3**	**Object 4**	**Object 5**
Plastic Type	PT	PP	PP	PP	PT
Grasp Position (x, y, z) [m]	1.2, 0.65, -0.05	1.2, 0.6, 0.13	1.2, 0.5, 0.13	1.2, 0.65, 0.1	1.2, 0.6, 0.05
Grasp Time [s]	6	12	17.5	23.5	28.5
Place Position (x, y, z) [m]	1.2, -0.15, 0.2	1.2, 0.15, 0.2	1.2, 0.1, 0.2	1.2, 0.19, 0.1	1.2, -0.2, 0.05
Place and Standby Time [s]	7-11	13-16	18-22	24-26	30-33

The last Section 7.2.2 is plotted simultaneously with the simulated model in Isaac Sim, just to observe the accuracy of desired and actual end-effector position. Due to the plots being generated through the tf listener in ROS, the plottings have a different start time but achieve the same trajectory. The joints of both models are compared in the next section.

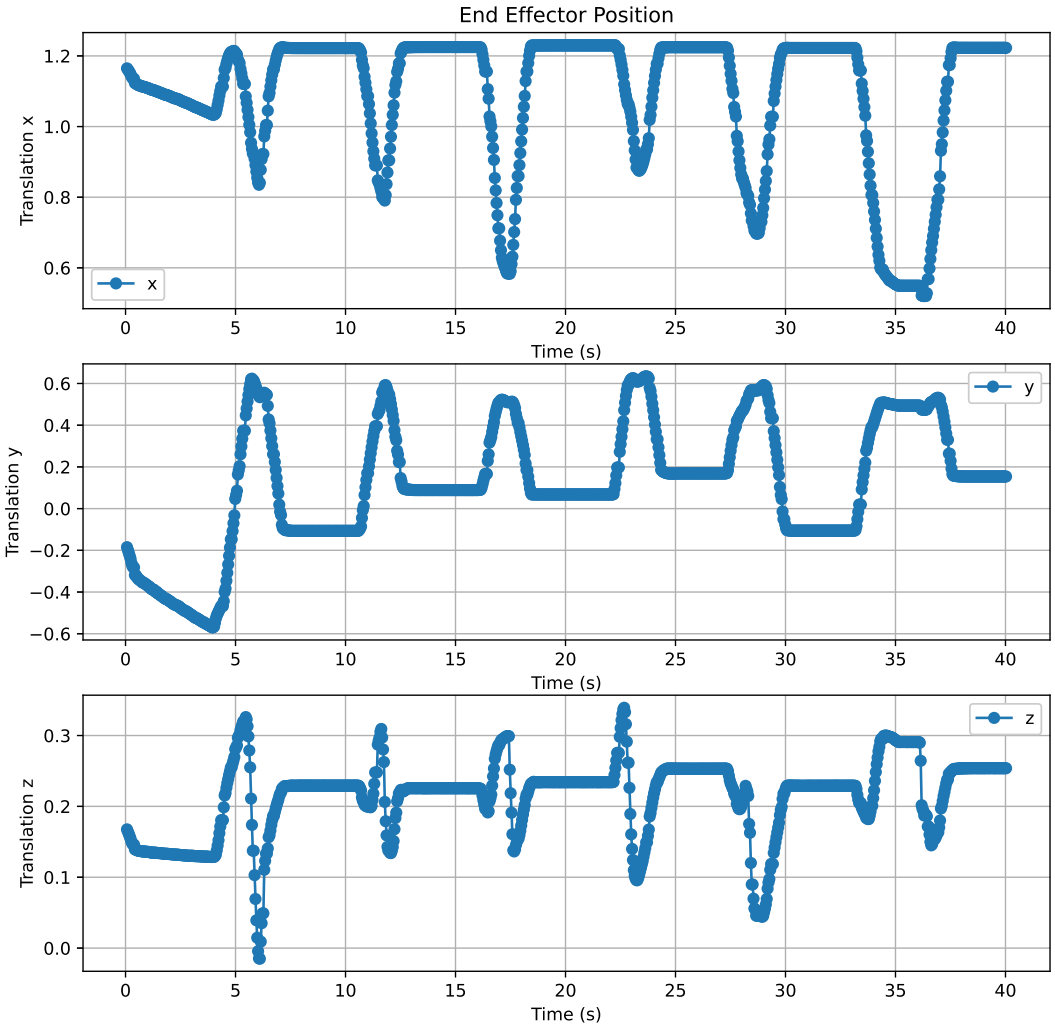


Figure 7.7: Tool-point path in physical space, with RL

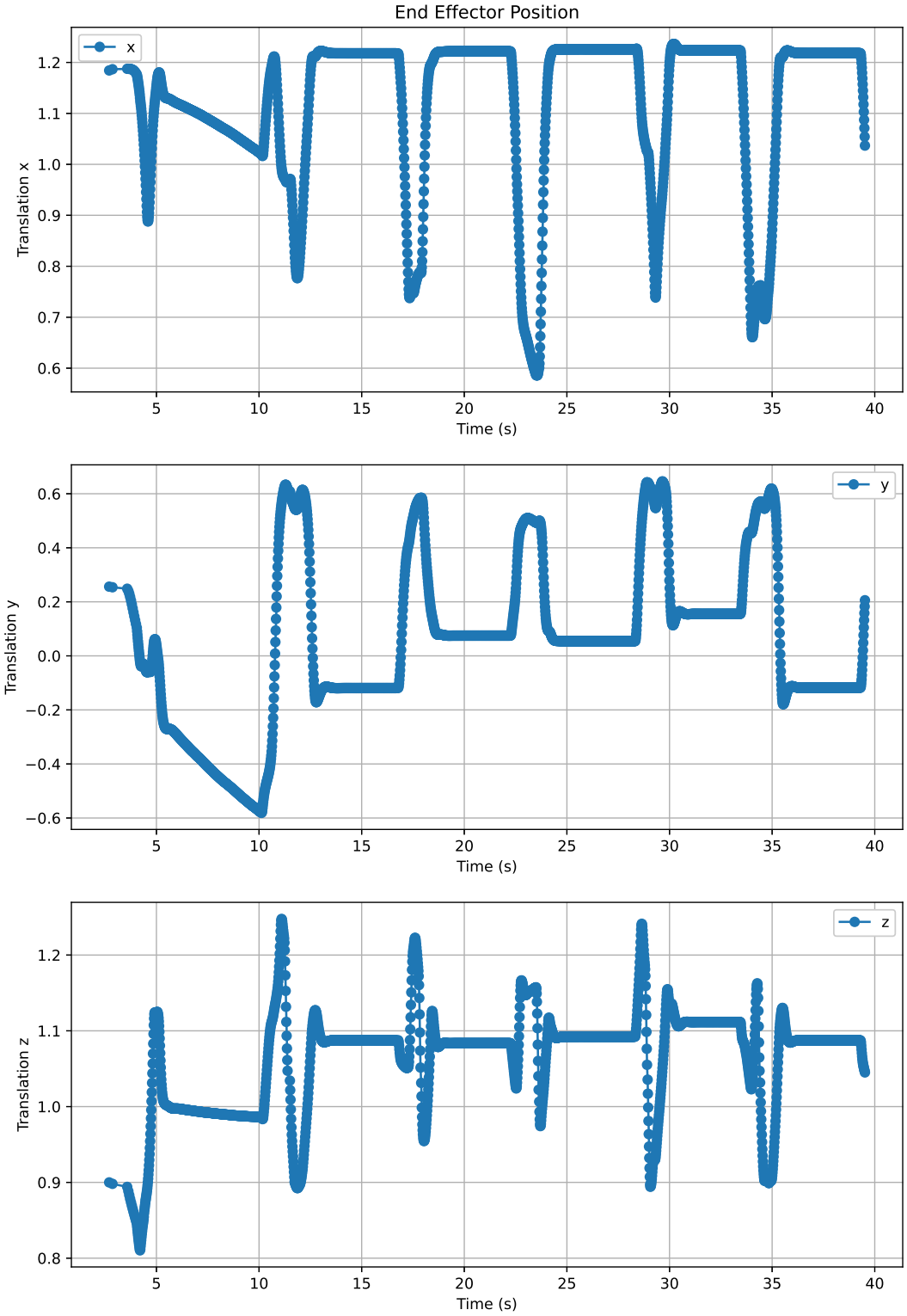


Figure 7.8: Tool-point path in digital space, with RL

7.2.3 Joint Angular Displacement Comparison

The joint placement displacement is plotted for comparing the deviation between the simulated and real model. Each joint is plotted in the time scope of 65 seconds. The results are generated by plotting the joint feedback state from the UR driver, and compared to the joint state from the digital model in Isaac Sim. The real robot is presented in the blue graph, and the simulated is presented in the orange graph.

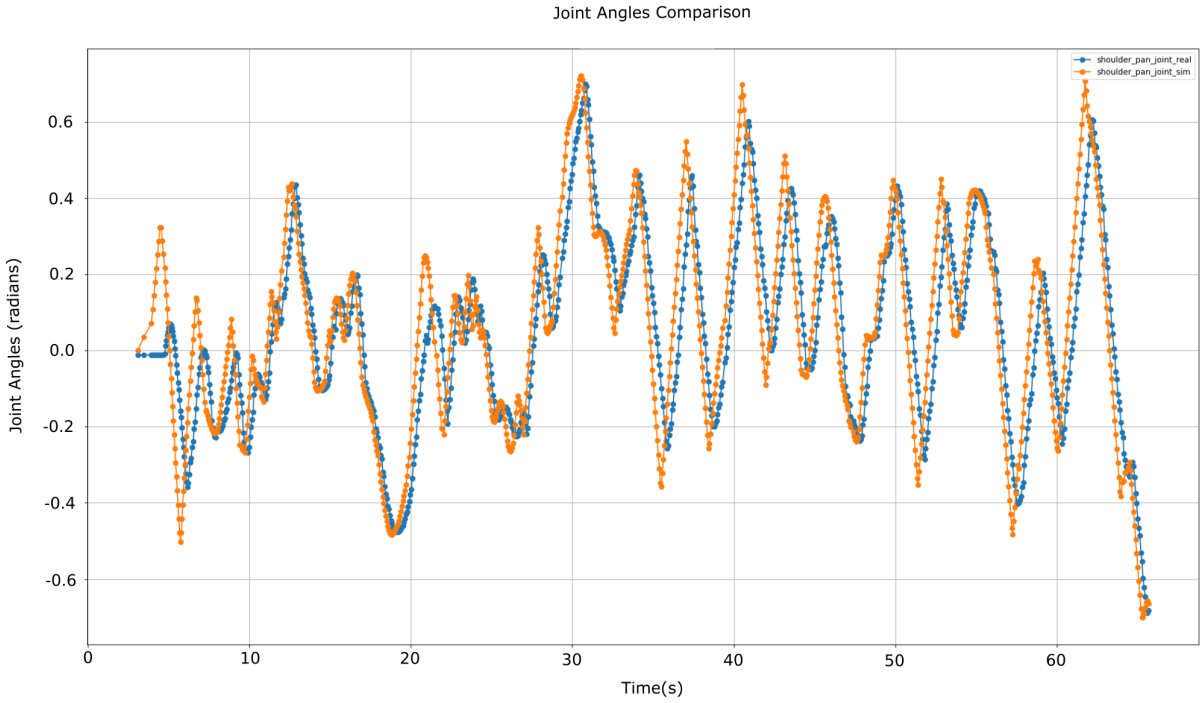


Figure 7.9: Joint: Base

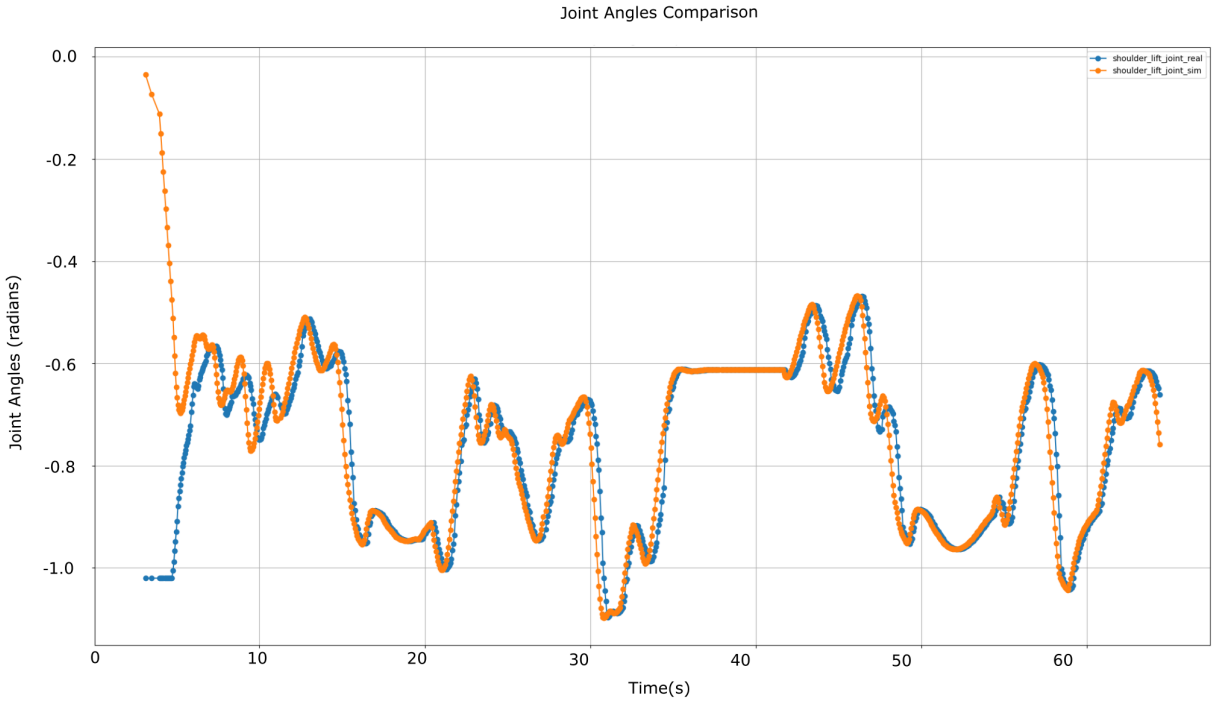


Figure 7.10: Joint: Shoulder

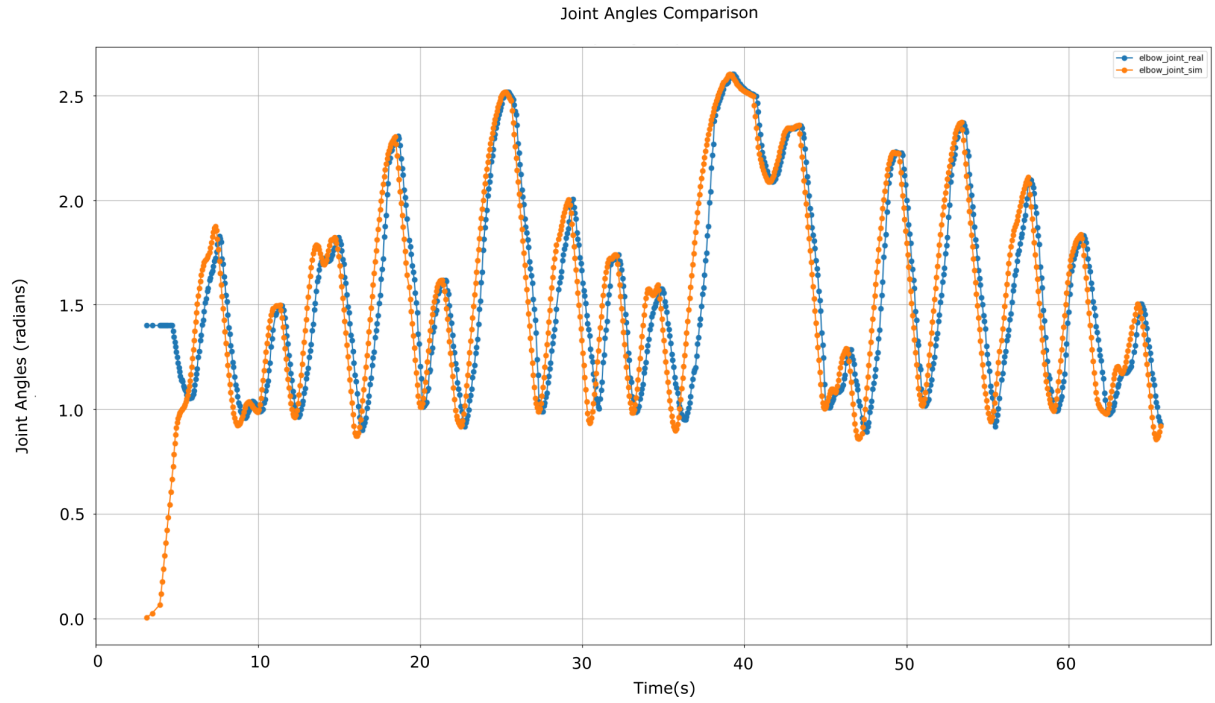


Figure 7.11: Joint: Elbow

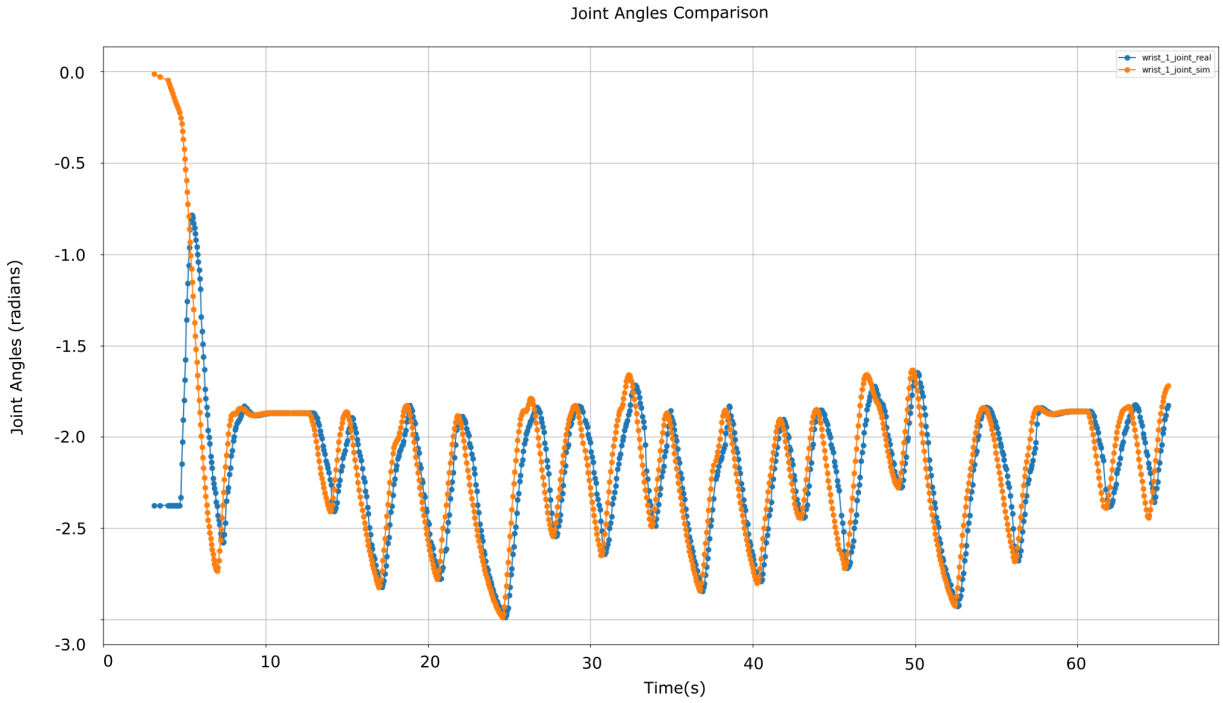


Figure 7.12: Joint: Wrist 1

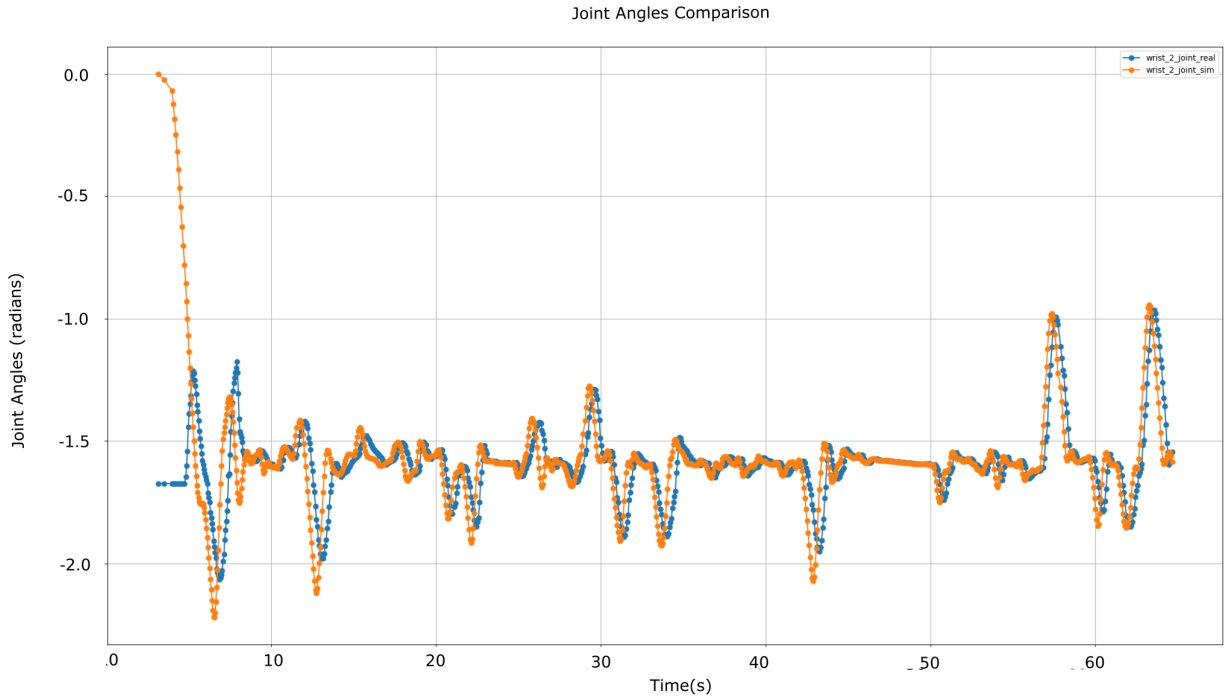


Figure 7.13: Joint: Wrist 2

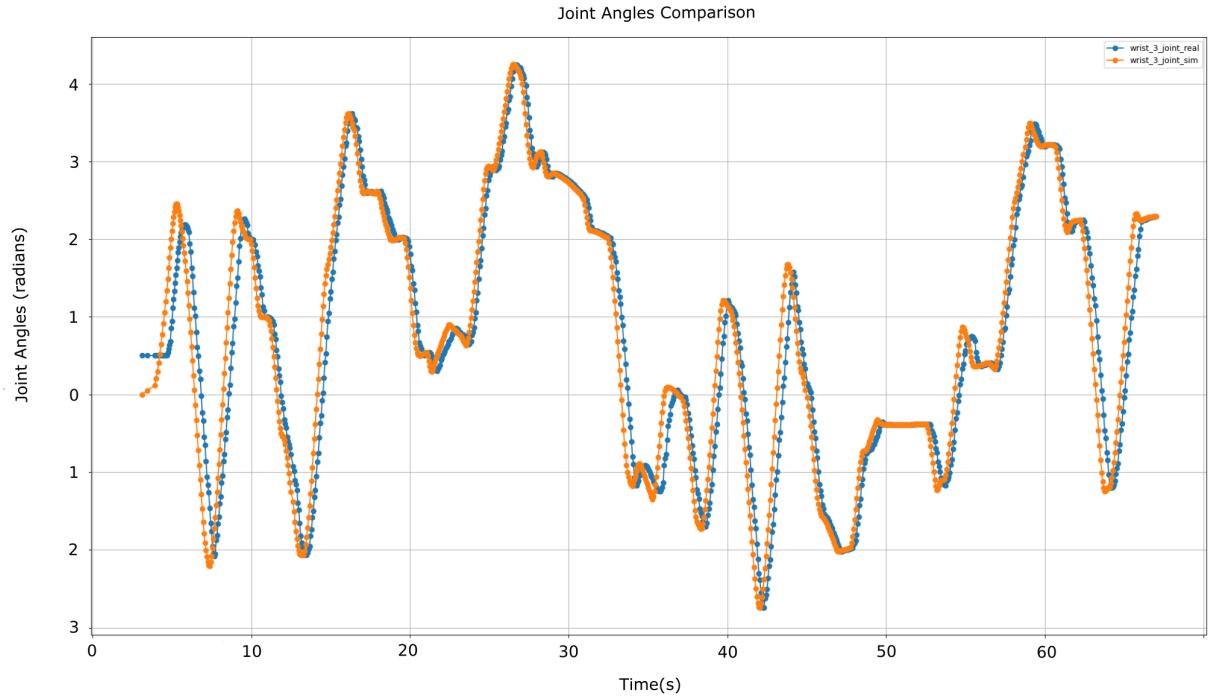


Figure 7.14: Joint: Wrist 3

7.3 Reinforcement Learning Training

The metric for Consecutive Successes per Frame is plateaued at a value of 12, after 200 million frames, indicating stable performance and model convergence.

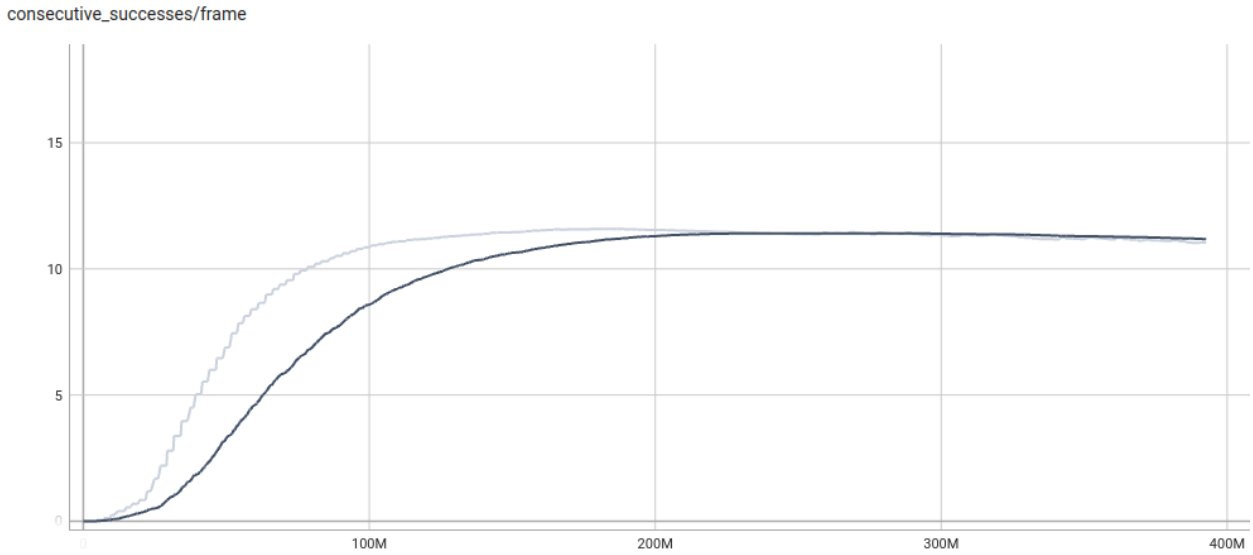


Figure 7.15: Consecutive Successes per Frame

For the Rewards per Episode, the graph demonstrates a flattening trend at approximately 700 on the x-axis and nearly 9000 on the y-axis, after which a minor decrement in values is observed, indicating a potential saturation of the RL model's learning capacity.

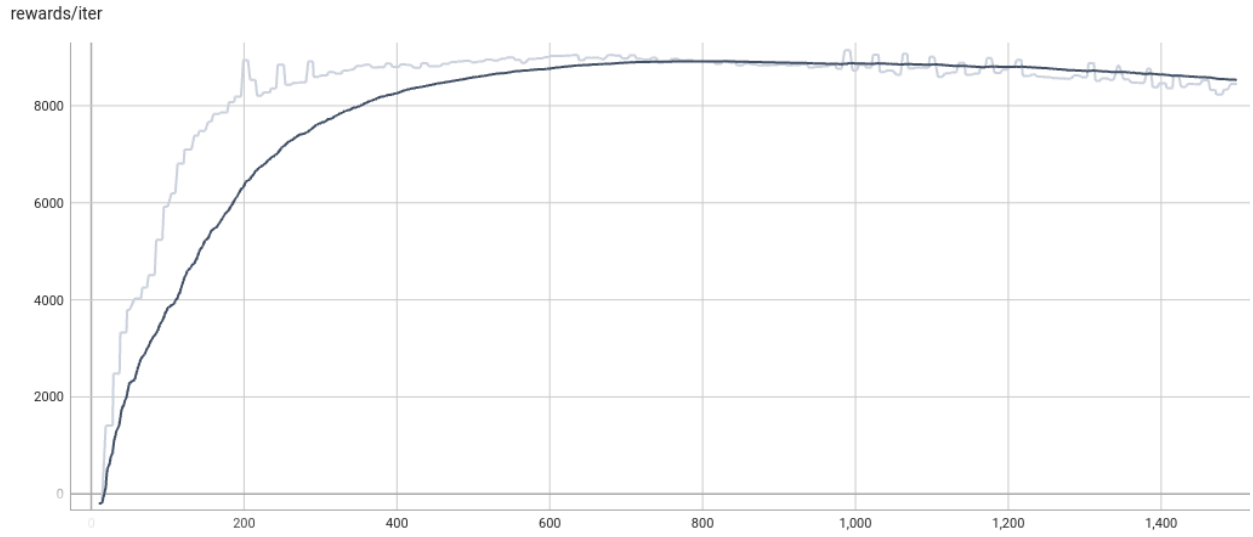


Figure 7.16: Rewards per Episode

Chapter 8

Discussions

8.1 Problem Statement Summary

The conducted research has focused on the development of an enhanced control system for a robotic arm in a pick-and-place setting. The work particularly emphasizes the application of ML techniques, including instance segmentation and reinforcement learning, to improve the manipulator's performance in motion control. The research has been driven by the problem statements presented in Chapter 1.3. Below the main findings of the conducted research related to these problem statements are summarized:

- Instance segmentation data derived from a single RGB camera can be utilized effectively in real-world scenarios by developing an optimal grip point algorithm. This algorithm processes the segmentation data to calculate optimal grasp angle and width, thus facilitating precise object recognition and tracking.

- The application of reinforcement learning in motion control can offer advantages such as adaptability and continuous learning, which are particularly beneficial for handling objects on a conveyor belt system. The conducted research and system implementation also verify that reinforcement learning, particularly Proximal Policy Optimization (PPO), can adapt to the velocity and direction of moving objects, thus improving efficiency and adaptability over inverse kinematics.

- A Proximal Policy Optimization (PPO) algorithm can provide consistent and precise control of a UR10e robotic arm by enhancing the velocity, accuracy, and adaptability of the robotic arm's movements. The consistency and predictability of this control can be optimized by designing a reward function that takes into account distance and orientation from the target object, thereby influencing the movement of the robot arm.

- Reward engineering can enhance modularity in training robots by administering task control independently for each environment in a simulation. This enables the robot to learn specific tasks effectively, thereby improving overall performance. Specific techniques, such as incremental task complexity and curriculum learning, can be used to design reward systems that support modular robot learning.

- The redundancy control of a UR10 robot arm can be improved by implementing a reinforcement learning model trained in a digital twin simulation environment. This environment mirrors the real-world operational context, allowing the robot arm to adapt to various scenarios and improve overall performance. Potential limitations, such as the gap between simulation and real-world performance, can be addressed by iteratively refining the simulation model based on real-world feedback.

8.2 Leveraging Object Detection in Waste Management

In the context of the modern waste management industry, machine learning holds potential to supersede numerous conventional sensors, thereby redefining standardized perceptions. Predominantly, the project delineated in Section 1.2 underscores the indispensability of a real-time object detection algorithm capable of delivering precise and timely information about detected objects. The YOLO8 algorithm employed in this project has successfully delivered on this requirement. A key attribute sought after in the project is scalability, given the waste industry's ever-evolving landscape that presents new objects to manage in diverse environments. This necessitates an algorithm that can accurately identify, locate, and manipulate objects of varying shapes, sizes, and states of motion, utilizing only the input from an RGB camera. Hence, the object detection mechanism emerges as a critical factor in project success.

During the testing phase, the algorithm exhibited instances of false positives, detecting objects in frames where none were present. The experimental setup encountered varying lighting conditions, necessitating the use of augmentation techniques to mitigate these effects. Instead of enclosing the RGB camera FOV, these techniques allowed for a broader operational range for the manipulator, effectively increasing its versatility. Additionally, camera calibration methods have been incorporated to address perspective distortions. These modifications have enabled the construction of a scalable pipeline, allowing the system to adeptly handle the dynamic nature of the waste management industry. The custom tracking algorithms for object tracking is crucial, as the algorithm was unable to track objects. During this project, the developers of yolov8 updated an almost identical solution by implementing BYTETrack, at the end of this project for tracking objects. The gripping algorithm for determining optimal grip point, did also give a general solution for where the gripper should grip objects, without needing any info on the shape to the specific object. The system was able to control the UR10e robot and OnRobot RG6 gripper in real time, enabling a functional pick-and-place system. However, improvements can be made to handling inactive pixels within objects and testing the optimal grip points algorithm.

Predictions vs Ground Truths

To uncover the reasons behind the incorrect predictions of the PP-5 SFTB class, the dataset labeling was subjected to a thorough analysis. Initially, the final dataset labeling distribution was evaluated using the bar chart as shown in Figure J.1. A significant imbalance in the distribution of classes can be observed when comparing some of the classes. However, this imbalance does not directly account for the inaccurate predictions of the PP-5 SFTB class, given its adequate representation in the dataset. Moreover, the label distribution implies that even classes that are not well represented can still be accurately classified. The scatterplots in the same Figure J.1 highlight that instances are predominantly spread across the images, which is suggestive of generalized training. Object localization and orientation significantly impacts the trained model and how well it performs during testing for new cases.

The PP-5 SFTB class's poor performance prompted a further evaluation of the dataset. The class is represented by a box with and without a lid, as displayed in Figure 7.1. It is observed that when the object appears markedly different with and without the lid, it significantly influences the training result. This observation points to the limited accuracy of the Computer Vision AI model for instance segmentation in predicting a class with substantial variations in color, texture, and shape, particularly in the context of a small dataset.

Label Correlogram Results

In Figure I.1, label correlogram results are investigated for the x and y-axis as described in Section 4.7, and the scatterplot matrix indicates that it has managed to balance the placements of objects within the frame. Done through physical placements of objects in the images, manipulation of the images through augmentations by rotation, and flipping in the creation of the dataset. Though, is also enabled through randomized augmentation features done during training.

The first notable observation pertains to the correlogram for the 'x' and 'y' axes, which represent the spatial placement of the bounding boxes within the image frames. The scatterplot matrix suggests a balanced distribution of objects throughout the image frame. This balanced distribution was achieved through a deliberate strategy of physical placement of objects within the images during data collection, manipulation of images through augmentations such as rotation and flipping during the dataset creation, and random augmentations applied during the training phase.

The x and y axis vs. the width and height axis of the correlogram results shows that the predicted bounding boxes are mostly between 0.1-0.9 in the x and y-axis, but still a significant amount at 0.0-0.1 and 0.9-1.0. By analyzing the plots, the height and width, are mostly centralized at about 0.1 of the total height or width of the image. This does not necessarily mean that height and width are correlated, but the most common values are either 0.1 for height or width.

The last label correlogram looking into the width and height axis indicates a noticeable correlation between height and width, meaning the most common shape is close to squared. The matrix plot also shows density at a 0.2 to 0.1 aspect ratio. Since the density is somewhat similar for height vs. width and width vs. height from 0.2 to 0.1, it is likely that it is a standard shape caused by the rotation of images for the dataset.

In conclusion, the label correlogram results provide profound insights into the performance and capabilities of our trained YOLOv8 model. It highlights the model's capacity to identify and localize objects of varying sizes across different locations within the image frame, a testament to the effectiveness of the employed training techniques and data augmentations.

8.3 Evaluation of Pick-And-Place operation

The completed autonomous sorting facility is evaluated based on logged data and observations. The motion control systems can be activated conveniently through a modular node system, ensuring that the input and output interfaces of each control system are uniform. The IK motion control system was capable of continually and redundantly sorting objects, achieving an cycle time of 7.5 seconds. This motion control system would be beneficial for picking out abnormalities, which was one of the use cases. The RL motion control system was able to attain the goal position swiftly and precisely, with a cycle time of 2.5 seconds. It should be noted that the RL model did not implement the joint movement and actuation of the gripper. After reverse engineering the gripper, the cycle time was found to be approximately 2 seconds. Since the goal was to train the RL model to move as fast as possible to manage moving objects, the implementation of the gripper was not implemented. The vision system requires a amount of input for the integration of new objects; a minimum of 10 images of the object is sufficient, on a general level, without tested large amounts of data. The facility has the capability to pick and place objects using a gripper with an adjustable width, ranging from 5 to 16 cm. However, it cannot manage smaller objects due to the limitations of the gripper control node actuation mechanism, which can only operate from fully open to closed with a set force of 120N.

Experimental Setup

The end effector position in X, Y, and Z is plotted to study accuracy, velocity, and path planning for the robot. Three plot is conducted, and evaluated. Both motion control systems are tested with the same prerequisites, where the coordinates for grip position sent from the vision system. Gripper was neglected in the RL test, so the test case is more of an observation. For the experimental setup, objects were placed at the start of the conveyor belt with three different placements at each drop to the conveyor belt. Object positioned across at equal intervals with 1/3 of the total width of the conveyor belt. The end effector path in 3D space is plotted by the reading of wrist 3 links in RViz is shown in Appendix E. The evaluation was based on the same setup, but in each translation axis in accordance with time. All plots are with the base of the robot as origo. Since the vision system sends the data at the same time, the object is picked in the same length direction of the conveyor belt, which relates to the y axis, but the some differences based on where to grip the object. The x axis is visualizing where the objects was dropped down on the conveyor belt and where it was placed. The objects was sorted to each of their respective end point, based on where the plastic should be sorted by class. The period of one cycle operation is at an average of 8 seconds, with some variance correlating to where the objects is positioned at the conveyor belt and if the gripper needs to be rotated. An analysis of an RL model with a focus on the joint angular displacement was taken. Each joint underwent individual assessment, and performance was charted over a 60-second period, at the maximum velocity to randomly coordinates, without the robot going into safety mode. Comparisons were drawn between simulation results and real-world measurements for each joint. Outputs from the model, namely simulated joint angles, were plotted against actual joint angles sourced from a real-world robotic arm via the ROS trajectory state interface.

8.4 Gripper

This embedded system could be more refinement. For instance, a Raspberry Pico, an even smaller and more compact microcontroller, could have been utilized for a similar function. The Pico can execute the required tasks while also reducing the footprint of the modification, making the overall system less invasive and more streamlined.

8.5 Reinforcement Learning

Proximal Policy Optimization (PPO), an influential algorithm for policy optimization, has been instrumental in the advancement of this research. It offers an adept resolution to the exploration-exploitation trade-off, a central challenge in reinforcement learning. In the context of constructing a physics simulation environment that mirrors real-world dynamics and continuously evaluating a Reinforcement Learning model in that environment, PPO's advantageous attributes become apparent. Its relative lightweight and rapid training capabilities, combined with its demonstrated proficiency in robotic tasks, made it an optimal choice for this project.

However, it is crucial to consider other potential algorithms to enhance the performance further. For instance, exploring the Soft Actor-Critic (SAC) model might yield superior results. SAC is a model-free algorithm that efficiently balances exploration and exploitation by maximizing entropy alongside the expected return. This quality could potentially enhance the performance and robustness of the system.

Nevertheless, the focal point of this research has been the establishment of a digital twin that accurately approximates real-world physics to control the UR10 manipulator effectively. This goal entails the intricate task of matching the virtual model's dynamics with the physical robot's behavior, facilitating reliable training and testing of reinforcement learning models. In this regard, the successful implementation of PPO serves as a testament to the viability of the chosen approach.

The chosen strategy also underscores the crucial role of simulation in reinforcement learning for robotics. Simulations offer a safe, flexible, and cost-effective platform for testing and improving algorithms before deployment on physical systems. Yet, bridging the gap between simulation and reality, the so-called 'reality gap', remains a significant challenge in this field. This research contributes to addressing this challenge by demonstrating the successful application of a simulated model to control a physical robot. However, continued work is needed to refine these methods and enhance the realism and applicability of such simulation models.

The configuration of a reinforcement learning algorithm's hyperparameters plays a pivotal role in its overall performance and effectiveness. A vital aspect of this configuration is the number of environments - the parallel instances of the task under study. This factor affects the learning process, including the selection of other hyperparameters such as the mini-epochs, learning rate, and entropy coefficient.

The concept of mini-epochs is integral to the Proximal Policy Optimization (PPO) algorithm. The number of mini-epochs determines the extent of learning from each set of experiences. When increasing the number of environments, a proportional increase in the number of mini-epochs might be beneficial. This adjustment could facilitate more comprehensive learning from the expanded data set. However, an essential caveat to consider is the computational cost, which would likely rise with an increase in mini-epochs. A conservative initial increase, perhaps to ten mini-epochs, is a prudent starting point, and subsequent adjustments can be made depending on observed performance outcomes.

The learning rate is a paramount hyperparameter in optimization algorithms, and its setting often depends on the specific task and the reward scale. A static learning rate within the range of 1e-3 to 1e-5 is commonly used in reinforcement learning tasks. However, this range isn't sacrosanct, and some tasks might require deviation. Larger learning rates could expedite learning but could also destabilize the process. Conversely, smaller learning rates could stabilize learning, albeit at a slower pace.

The entropy coefficient, another hyperparameter, encourages exploration in the learning process by introducing an entropy bonus to the policy's objective function. A higher entropy coefficient engenders more exploration, while a lower coefficient leads to greater exploitation. For a static entropy coefficient, starting with a small positive value, such as 0.01, is advisable, and subsequent adjustments can be made based on performance.

It is worth noting that hyperparameter adjustment is an iterative process that requires diligent monitoring and fine-tuning. It is often beneficial to run multiple trials to garner a more reliable estimate of the algorithm's performance. This approach allows for the consideration of the inherent variability in reinforcement learning outcomes. Ultimately, the balance between exploration and exploitation, learning speed and stability, and computational efficiency and thoroughness must be meticulously maintained for successful reinforcement learning implementation.

8.5.1 Results and Discussion Reinforcement Learning

The reinforcement learning algorithm, combined with the reward function and Sim2Real implementation, enables the agent to learn the control policies required for performing the three tasks sequentially. The agent adapts to the changing tasks and learns to minimize the distance and orientation differences between the object and the target while regularizing its actions.

By leveraging the Isaac Sim environment and ROS, the learning process can be seamlessly transferred from the simulated environment to the real world, demonstrating the potential of reinforcement learning for controlling robotic systems in various applications.

8.6 Scope of Project

The conclusion of this research project illuminates numerous opportunities for further exploration and refinement. It is evident that the integration of advanced robotics and machine learning techniques yields promising results in industrial settings, particularly for pick-and-place operations. However, the complexity and of such applications necessitate continuous innovation and adaptation.

Chapter 9

Conclusions

This research has extensively explored the use of machine learning techniques to enhance the control system of a robotic arm for pick-and-place operations, with promising results.

One of the primary achievements was the successful application of instance segmentation data derived from a single RGB camera for object perception. An optimal grip point algorithm was developed that processed this segmentation data to calculate the optimal grasp angle and width. The application facilitated accurate object recognition and tracking.

Reinforcement learning, particularly the Proximal Policy Optimization (PPO) algorithm, emerged as a pivotal element in the system's motion control. It demonstrated notable adaptability and continuous learning capacity. The PPO algorithm was able to adapt to the velocity and direction of moving objects, thus enhancing both the efficiency and adaptability of the robotic arm's movements. The PPO algorithm also allowed consistent and precise control of a UR10e robotic arm by effectively enhancing its velocity, accuracy, and adaptability. The RL motion control achieved a cycle time for the end-effector placements of 2.5 seconds. The gripper for a full cycle of open and close took 2 seconds, whereas future work would be implementing a gripper with the RL solution.

The concept of reward engineering was explored to enhance modularity in robotic training. Techniques like incremental task complexity and curriculum learning were employed to design effective reward systems, thereby improving overall performance. Furthermore, the dynamic control and speed of the UR10 robot arm were improved by implementing a reinforcement learning model trained in a digital twin simulation environment that accurately mirrors real-world operations. This led to improved adaptability to various scenarios and overall performance enhancement.

However, the study also identified a potential limitation concerning the gap between simulation and real-world performance. It is recommended that future efforts be directed towards iteratively refining the simulation model based on real-world feedback to bridge this gap for the RL solution. The motion control system using trajectory calculations via MoveIt2 performed adequately, managing to sort an average of one object per seven seconds, including pick and place.

In conclusion, the contributions of this research extend beyond the theoretical domain and have practical implications for real-world industrial automation. The results provide a solid foundation for future studies aiming to further optimize pick-and-place operations. The enhancements in machine learning applications, control systems, and simulation environments present significant steps forward in the realm of industrial robotics. The study's findings highlight the untapped potential and versatility of machine learning and reinforcement learning in this field, paving the way for further advancements.

Bibliography

[1] Lammert Bies. *RS485 serial information*. 2021. URL: https://www.lammertbies.nl/comm/info/rs-485.

[2] Alexey Bochkovskiy. "YOLOv4: Optimal Speed and Accuracy of Object Detection." In: *arXiv preprint arXiv:2004.10934* (2020). URL: https://arxiv.org/pdf/2004.10934.pdf.

[3] Denys88. *rl_games*. Reinforcement Learning for games implemented in PyTorch. 2023. URL: https://github.com/Denys88/rl_games/tree/master.

[4] j3soon. *OmniIsaacGymEnvs-UR10Reacher*. UR10 Reacher Reinforcement Learning Sim2Real Environment for Omniverse Isaac Gym/Sim. 2023. URL: https://github.com/j3soon/OmniIsaacGymEnvs-UR10Reacher.

[5] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *YOLO by Ultralytics*. Version 8.0.0. Jan. 2023. URL: https://github.com/ultralytics/ultralytics.

[6] Andrew Lobbezoo. *Reinforcement Learning for Pick and Place Operations in Robotics: A Survey*. 2021. URL: https://www.mdpi.com/2218-6581/10/3/105.

[7] Lynred. *How infrared detectors are bringing superior precision to waste sorting processes*. 2021. URL: https://lynred.com/blog/how-infrared-detectors-are-bringing-superior-precision-waste-sorting-processes.

[8] Chengqi Lyu and Zhang. "RTMDet: An Empirical Study of Designing Real-Time Object Detectors." In: *arXiv preprint arXiv:2212.07784* (2022).

[9] MATHLAB. *Inverse kinematics (IK) algorithm design with MATLAB and Simulink*. 2020. URL: https://www.mathworks.com/discovery/inverse-kinematics.html.

[10] MathWorks. *What Is Camera Calibration?* 2023. URL: https://www.mathworks.com/help/vision/ug/camera-calibration.html.

[11] Mecademic. *Modbus*. URL: https://www.mecademic.com/en/what-are-singularities-in-a-six-axis-robot-arm.

[12] FAIR Meta AI Research. "Segment Anything." In: *arXiv preprint arXiv:2304.02643* (2023). URL: https://arxiv.org/pdf/2304.02643.pdf.

[13] Modbus.org. *Modbus*. 2023. URL: https://modbus.org/.

[14] Mark Moll and Ioan A. Sucan. *The Open Motion Planning Library*. [Online; accessed 25.04.2023]. 2021. URL: https://ompl.kavrakilab.org/.

[15] MoveIt. *Planners Available in MoveIt*. 2023. URL: https://moveit.ros.org/documentation/planners/.

[16] NVIDIA. *Asset Converter Extension*. 2023. URL: https://docs.omniverse.nvidia.com/prod_extensions/prod_extensions/ext_asset-converter.html.

[17] NVIDIA. *Universal Scene Description (USD)*. URL: https://www.nvidia.com/en-us/omniverse/usd/.

[18] NVIDIA Omniverse. *OmniIsaacGymEnvs*. NVIDIA Omniverse Isaac Gym Environments for Reinforcement Learning. 2023. URL: https://github.com/NVIDIA-Omniverse/OmniIsaacGymEnvs.

[19] OnRobot. *Datasheet RG6*. URL: https://onrobot.com/sites/default/files/documents/Datasheet_RG6_v1.1_EN.pdf.

[20] OnRobot. *OnRobot Compute Box Description*. URL: https://onrobot.com/sites/default/files/documents/onrobot-compute-box-description_e10_en.pdf.

[21] OnRobot. *User Manual for TECHMAN OMRON TM*. URL: https://onrobot.com/sites/default/files/documents/User_Manual_for_TECHMAN_OMRON_TM_v1.05_EN_0.pdf.

[22] Modbus Organization. *Modbus Application Protocol Specification V1.1b*. URL: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf.

[23] Joseph Redmon. "YOLOv3: An Incremental Improvement." In: *arXiv preprint arXiv:1804.02767v1* (2018). URL: https://arxiv.org/pdf/1804.02767v1.pdf.

[24] Joseph Redmon. "You Only Look Once: Unified, Real-Time Object Detection." In: *arXiv preprint arXiv:1506.02640* (2016). URL: https://arxiv.org/pdf/1506.02640.pdf.

[25] ros-industrial. *ROS-Industrial Universal Robots support*. 2023. URL: https://github.com/ros-industrial/universal_robot.

[26] Gavriel State. *Introducing Isaac Gym: RL for Robotics*. 2021. URL: https://developer.nvidia.com/blog/introducing-isaac-gym-rl-for-robotics/.

[27] Juan Terven. "A Comprehensive Review of YOLO: From YOLOv1 and Beyond." In: *arXiv preprint arXiv:2304.00501* (2023). URL: https://arxiv.org/pdf/2304.00501.pdf.

[28] Ultralytics. *YOLOv8 model architecture*. URL: https://user-images.githubusercontent.com/27466624/239739723-57391d0f-1848-4388-9f30-88c2fb79233f.jpg.

[29] Viswanatha v, Chandana R K, and Ramachandra Ac. "Real Time Object Detection System with YOLO and CNN Models: A Review." In: *Xi'an Jianzhu Keji Daxue Xuebao/Journal of Xi'an University of Architecture and Technology* XIV (July 2022). DOI: 10.37896/JXAT14.07/315415.

[30] Chien-Yao Wang. "CSPNet: A New Backbone that can Enhance Learning Capability of CNN." In: *arXiv preprint arXiv:1911.11929* (2019). URL: https://arxiv.org/pdf/1911.11929.pdf.

[31] Jifeng Dai Wenhai Wang. "InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions." In: *arXiv preprint arXiv:2211.05778* (2023). URL: https://arxiv.org/pdf/2211.05778.pdf.

[32] wikipedia. *Modbus*. 2023. URL: https://en.wikipedia.org/wiki/Modbus#References.

[33] Takuya Azumi Yuya Maruyama Shinpei Kato. *ROS1/ROS2 architecture for DDS approach to ROS*. 2016. URL: https://www.researchgate.net/figure/ROS1-ROS2-architecture-for-DDS-approach-to-ROS-We-clarify-the-performance-of-the-data_fig1_309128426.

[34] Andy Zeng et al. "Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020. URL: http://arc.cs.princeton.edu.

[35] Xingyuan Zhou et al. "ByteTrack: Multi-Object Tracking by Associating Every Detection Box." In: *arXiv preprint arXiv:2110.06864* (2021). URL: https://arxiv.org/pdf/2110.06864.pdf.

Appendix A

Datasheet UR10e

UR10e

Specification

Payload	12.5 kg (27.5 lbs)	
Reach	1300 mm (51.2 in)	
Degrees of freedom	6 rotating joints	
Programming	12 inch touchscreen with polyscope graphical user interface	

Performance

Power, Consumption, Maximum Average	615 W	
Power, Consumption, Typical with moderate settings (approximate)	350 W	
Safety	17 configurable safety functions	
Certifications	EN ISO 13849-1, PLd Category 3, and EN ISO 10218-1	
Force Sensing, Tool Flange	Force, x-y-z	Torque, x-y-z
Range	100.0 N	10.0 Nm
Precision	5.0 N	0.2 Nm
Accuracy	5.5 N	0.5 Nm

Movement

Pose Repeatability per ISO 9283	± 0.05 mm	
Axis movement	Working range	Maximum speed
Base	± 360°	± 120°/s
Shoulder	± 360°	± 120°/s
Elbow	± 360°	± 180°/s
Wrist 1	± 360°	± 180°/s
Wrist 2	± 360°	± 180°/s
Wrist 3	± 360°	± 180°/s
Typical TCP speed	1 m/s (39.4 in/s)	

Features

IP classification	IP54
ISO 14644-1 Class Cleanroom	5
Noise	Less than 65 dB(A)
Robot mounting	Any orientation
I/O ports	
Digital in	2
Digital out	2
Analog in	2
Tool I/O Power Supply Voltage	12/24 V
Tool I/O Power Supply	2 A (Dual pin) 1 A (Single pin)

Physical

Footprint	Ø 190 mm
Materials	Aluminium, Plastic, Steel
Tool (end-effector) connector type	M8 | M8 8-pin
Cable length robot arm	6 m (236 in) cable included. 12 m (472 in) and high-flex options available.
Weight including cable	33.5 kg (73.9 lbs)
Operating temperature range	0-50°C
Humidity	90%RH (non-condensing)

Control Box

Features

IP classification	IP44
ISO 14644-1 Class Cleanroom	6
Operating temperature range	0-50°C
Humidity	90%RH (non-condensing)
I/O ports	
Digital in	16
Digital out	16
Analog in	2
Analog out	2
Quadrature Digital Inputs	4
I/O Power Supply	24V 2A
Communication	500 Hz Control frequency Modbus TCP PROFINET Ethernet/IP USB 2.0, USB 3.0
Power source	100-240VAC, 47-440Hz

Physical

Control box size (W x H x D)	460 mm x 449 mm x 254 mm (18.2 in x 17.6 in x 10 in)
Weight	12 kg (26.5 lbs)
Materials	Powder Coated Steel

The control box is also available in an OEM version.

Teach Pendant

Features

IP classification	IP54
Humidity	90%RH (non-condensing)
Display resolution	1280 x 800 pixels

Physical

Materials	Plastic, PP
Weight	1.6 kg (3.5 lbs) including 1m of TP cable
Cable length	4.5 m (177.17 in)

The teach pendant is also available in a 3PE option.

Figure A.1: UR10e datasheet

Appendix B

Camera Mount in Aluminium Profiles

All aluminum profiles are produced from Bosch Rexroth (BR) and bought though RS components.

Part Name	Quantity	Part No.	Price (each)	Total Price
BR Aluminium Profile Strut	5	3842993130	NOK 1 112,47	NOK 5 562,35
BR M8 Angle Bracket (40mm)	4	3842529383	TBD	TBD
BR M8 Angle Bracket (80mm)	7	3842530360	NOK 197,58	NOK 1 383,06
BR Angular Cover Cap (80mm)	7	3842548860	NOK 59,413	NOK 416,891
BR Joint Clamp (40mm)	10	3842532364	TBD	TBD

Table B.1: Bill of Materials

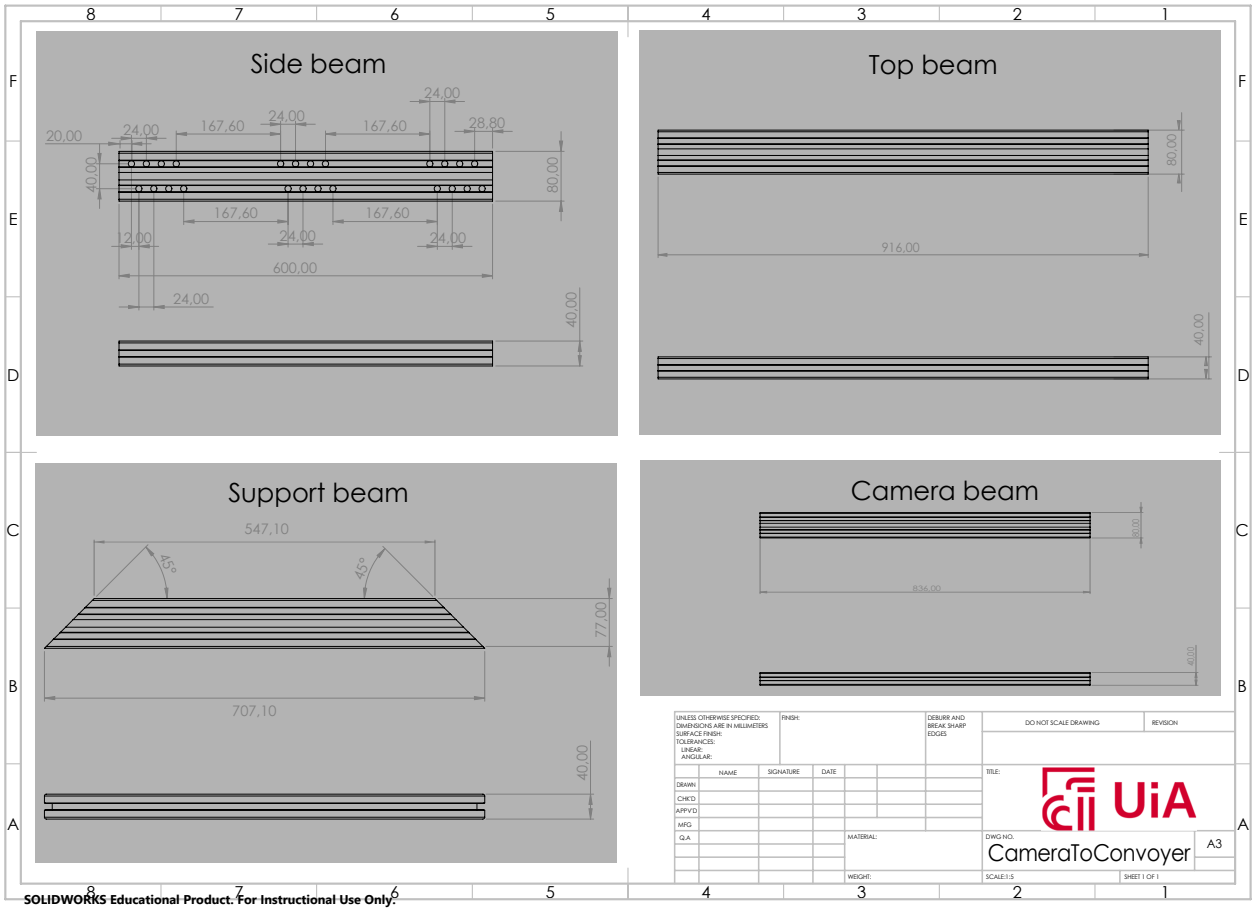


Figure B.1: Production Drawings of Camera Mount

Appendix C

Machining

In collaboration with the mechatronics department, 9 steel plates was machined and welded to make this facility mounted together. Figure Appendix C is a machined bracket mounting the camera to the aluminum railing. Rest of the figures Appendix C is welded together for making the robot stuck in position.

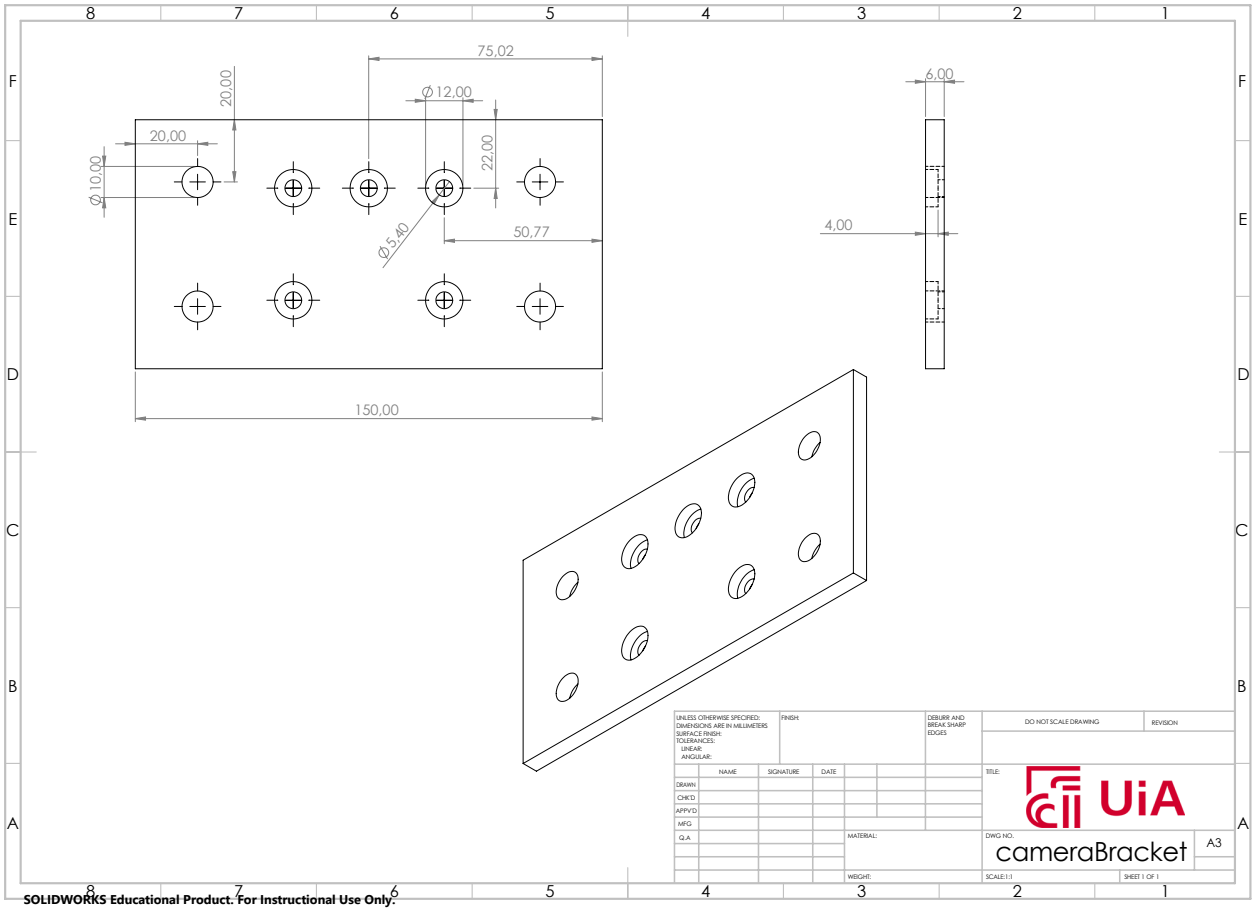


Figure C.1: Attachment From Camera to Mount

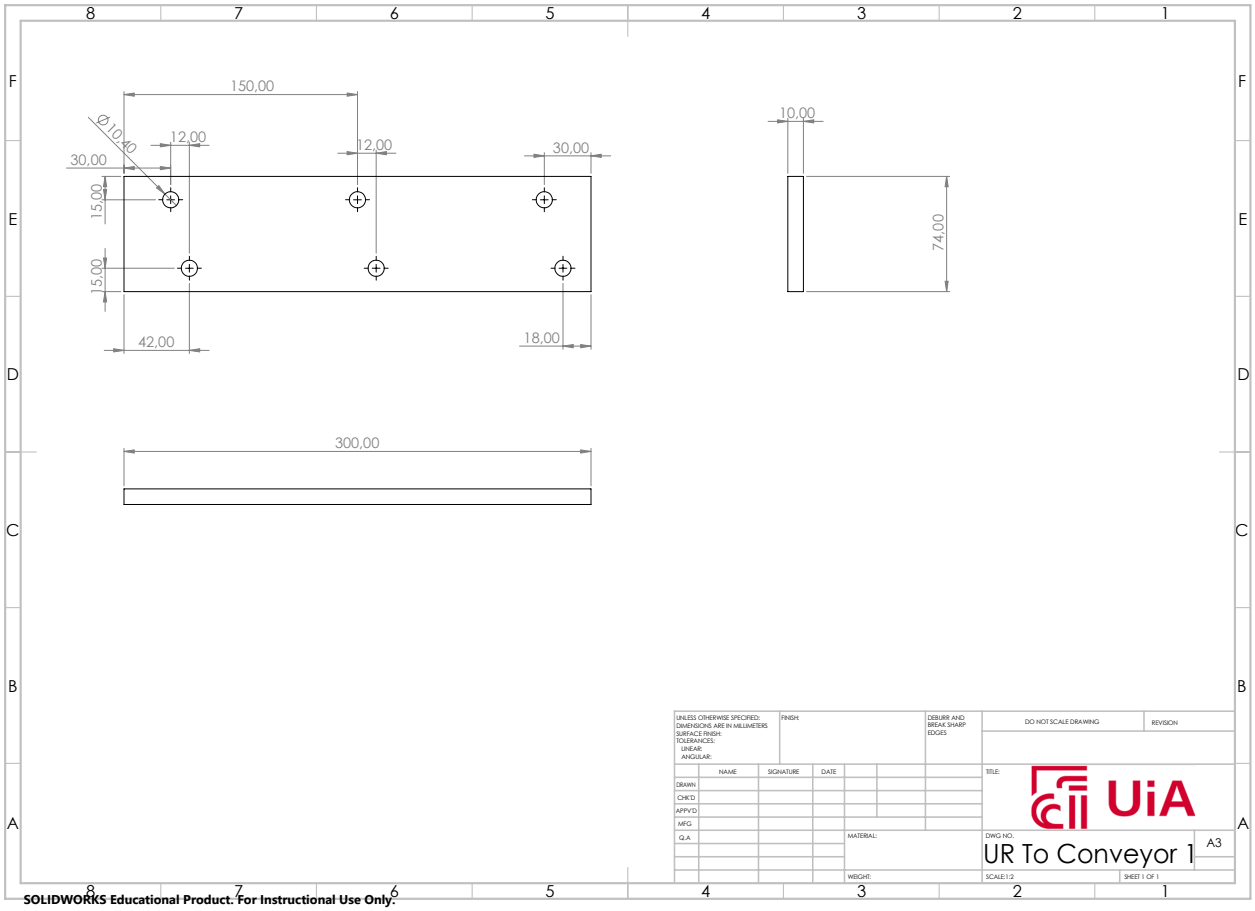


Figure C.2: Attachment From UR to Conveyor Part 2

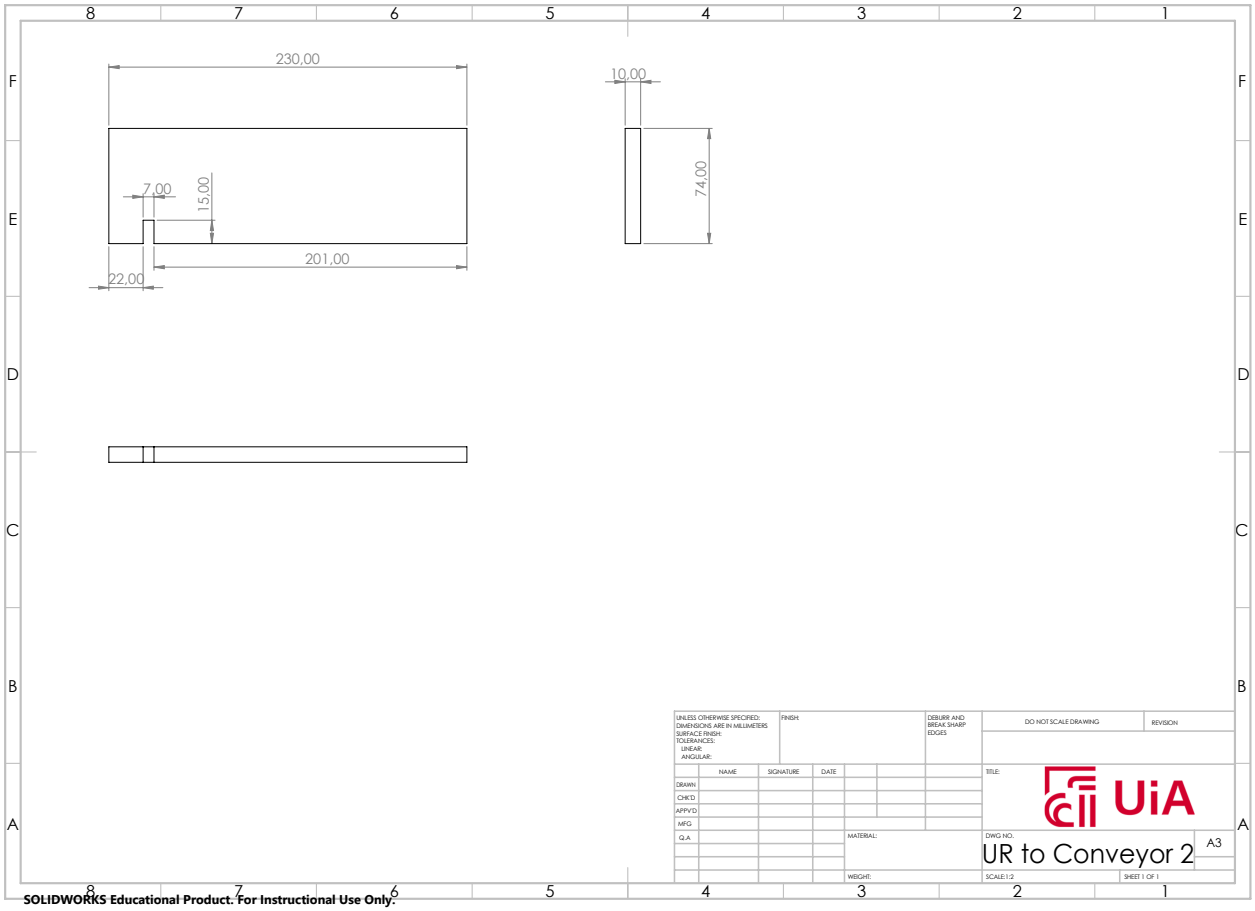


Figure C.3: Attachment From UR to Conveyor Part 1

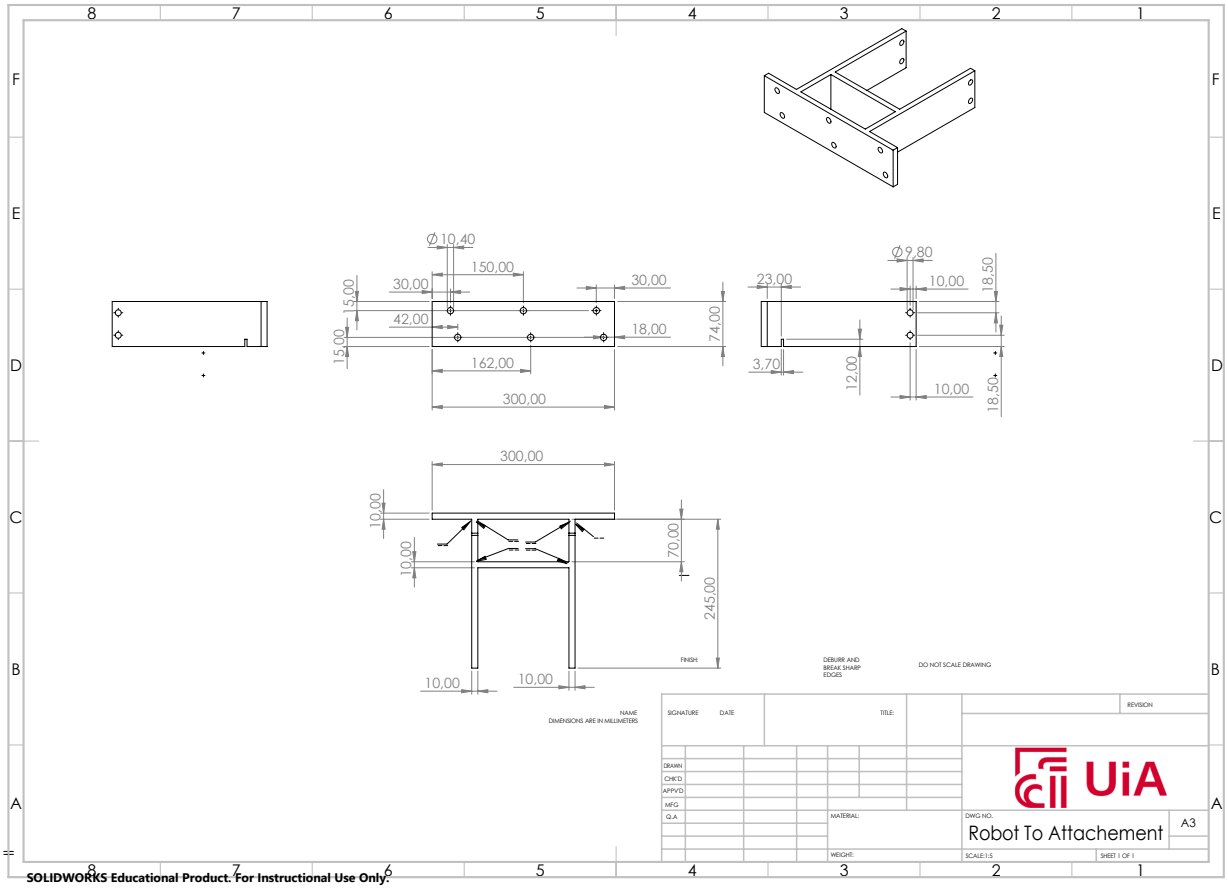
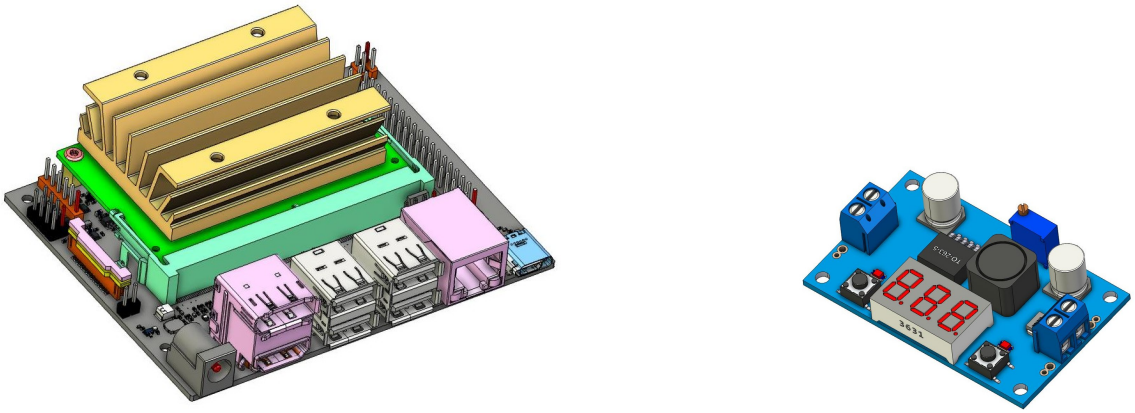


Figure C.4: Attachment to Robot Mount

Appendix D

Electrical components

The main components for making the remotely controllable gripper is consisting, of a single board computer Figure D.1a and a step-down convert Figure D.1b.



(a) Jetson Nano (b) LM2596

Figure D.1: Main Components for Controlling the Gripper Remote

Appendix E

Pick-And-Place in RViz

For testing the developed software without the robot, RViz is used as a simulator. objects are spawned based on coordinates from the vision system as a marker, represented in the figure below.

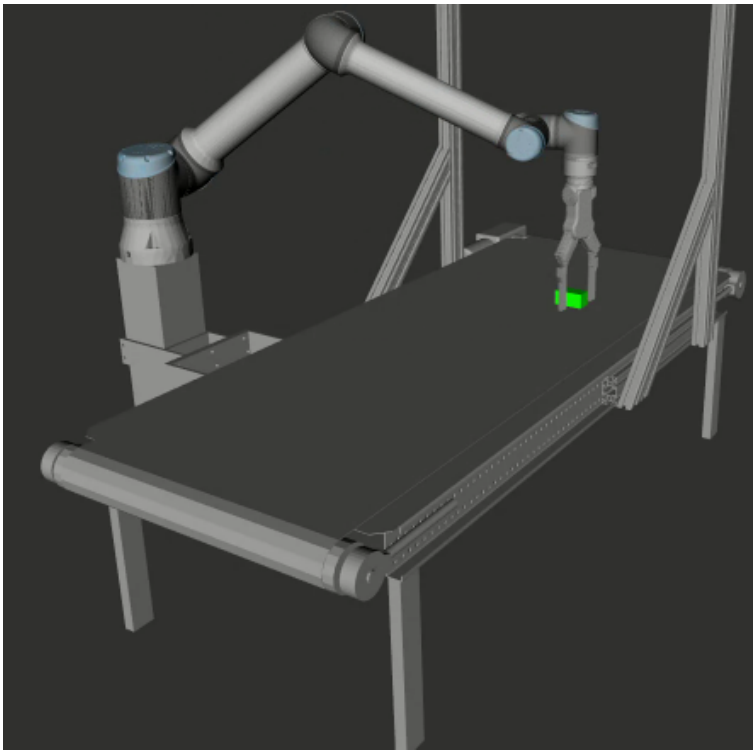


Figure E.1: Digital Twin in Rviz Picking Object

Appendix F

Process Line

This project is going to be implemented in several process plants. Both for sorting out abnormalities, but also a part for a full scale recycling station as point number 8 on the overview picture below Appendix F.

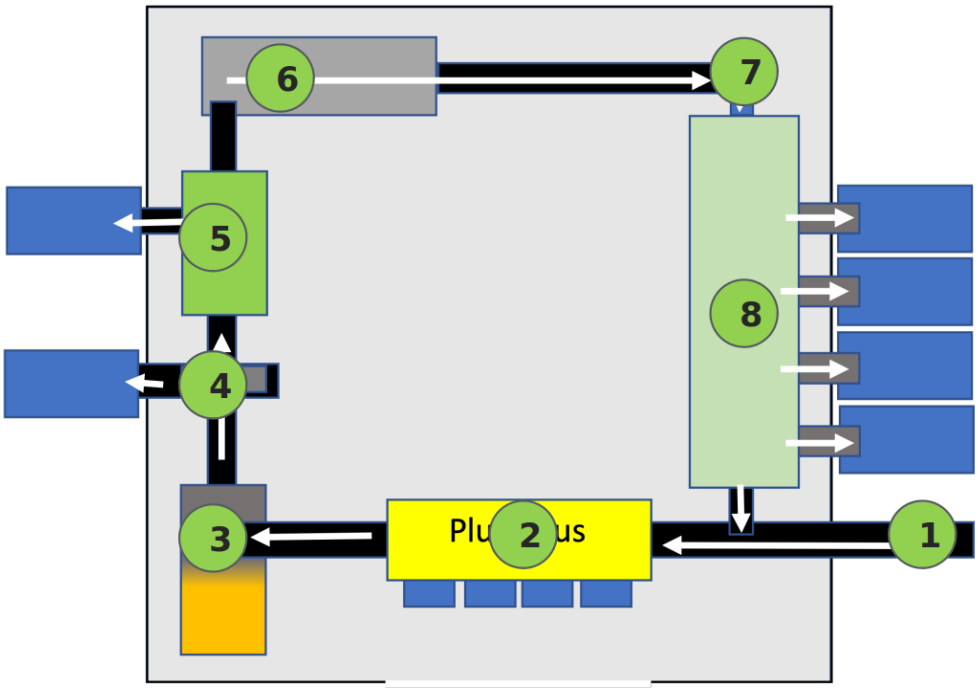


Figure F.1: Sorting Facility Top-Down View

Task Nr	Process
1	Waste Input
2	Manual Selection
3	Waste Grinder
4	Magnet Picks
5	Drum Sieve
6	Waste Magazine
7	Vibrator
8	Pick-And-Place

Figure F.2: Process Tasks

Appendix G

YOLOv8 argument configurations

```
task: segment
mode: train
model: yolov8x-seg.pt
data: /content/dataset/data.yaml
epochs: 100
patience: 50
batch: 16
imgsz: 640
save: true
save_period: -1
cache: false
device: null
workers: 8
project: null
name: null
exist_ok: false
pretrained: false
optimizer: SGD
verbose: true
seed: 0
deterministic: true
single_cls: false
image_weights: false
rect: false
cos_lr: false
close_mosaic: 0
resume: false
amp: true
overlap_mask: true
mask_ratio: 4
dropout: 0.0
val: true
split: val
save_json: false
save_hybrid: false
conf: null
iou: 0.7
max_det: 300
half: false
dnn: false
```

```
plots: true
source: null
show: false
save_txt: false
save_conf: false
save_crop: false
show_labels: true
show_conf: true
vid_stride: 1
line_thickness: 3
visualize: false
augment: false
agnostic_nms: false
classes: null
retina_masks: false
boxes: true
format: torchscript
keras: false
optimize: false
int8: false
dynamic: false
simplify: false
opset: null
workspace: 4
nms: false
lr0: 0.01
lrf: 0.01
momentum: 0.937
weight_decay: 0.0005
warmup_epochs: 3.0
warmup_momentum: 0.8
warmup_bias_lr: 0.1
box: 7.5
cls: 0.5
dfl: 1.5
pose: 12.0
kobj: 1.0
label_smoothing: 0.0
nbs: 64
hsv_h: 0.015
hsv_s: 0.7
hsv_v: 0.4
degrees: 0.0
translate: 0.1
scale: 0.5
shear: 0.0
perspective: 0.0
flipud: 0.0
fliplr: 0.5
mosaic: 1.0
mixup: 0.0
copy_paste: 0.0
cfg: null
```

```
v5loader: false
tracker: botsort.yaml
save_dir: runs/segment/
```

Appendix H

PPO Configurations

```yaml
params:
  seed: ${...seed}
  algo:
    name: ppo_continuous

  model:
    name: continuous_ppo_logstd

  network:
    name: actor_critic
    separate: False

    space:
      continuous:
        mu_activation: None
        sigma_activation: None
        mu_init:
          name: default
        sigma_init:
          name: const_initializer
          val: 0
        fixed_sigma: True

    mlp:
      units: [256, 128, 64]
      activation: elu
      d2rl: False

      initializer:
        name: default
      regularizer:
        name: None
```

Listing 1

```yaml
config:
  name: ${resolve_default:UR10PickAndPlace,${....experiment}}
  full_experiment_name: ${.name}
  device: ${....rl_device}
  device_name: ${....rl_device}
  env_name: rlgpu
  multi_gpu: False
  ppo: True
  mixed_precision: False
  normalize_input: True
  normalize_value: True
  value_bootstrap: True
  num_actors: ${....task.env.numEnvs}
  reward_shaper:
    scale_value: 0.01
  normalize_advantage: True
  gamma: 0.99
  tau: 0.95
  learning_rate: 5e-3
  lr_schedule: adaptive
  schedule_type: standard
  kl_threshold: 0.008
  score_to_win: 100000
  max_epochs: ${resolve_default:1500,${....max_iterations}}
  save_best_after: 100
  save_frequency: 200
  print_stats: True
  grad_norm: 1.0
  entropy_coef: 0.0
  truncate_grads: True
  e_clip: 0.2
  horizon_length: 64
  minibatch_size: 32768
  mini_epochs: 5
  critic_coef: 4
  clip_value: True
  seq_len: 4
  bounds_loss_coef: 0.0001

  player:
    deterministic: True
    games_num: 100000
    print_stats: True
```

Appendix I

Labels Correllogram

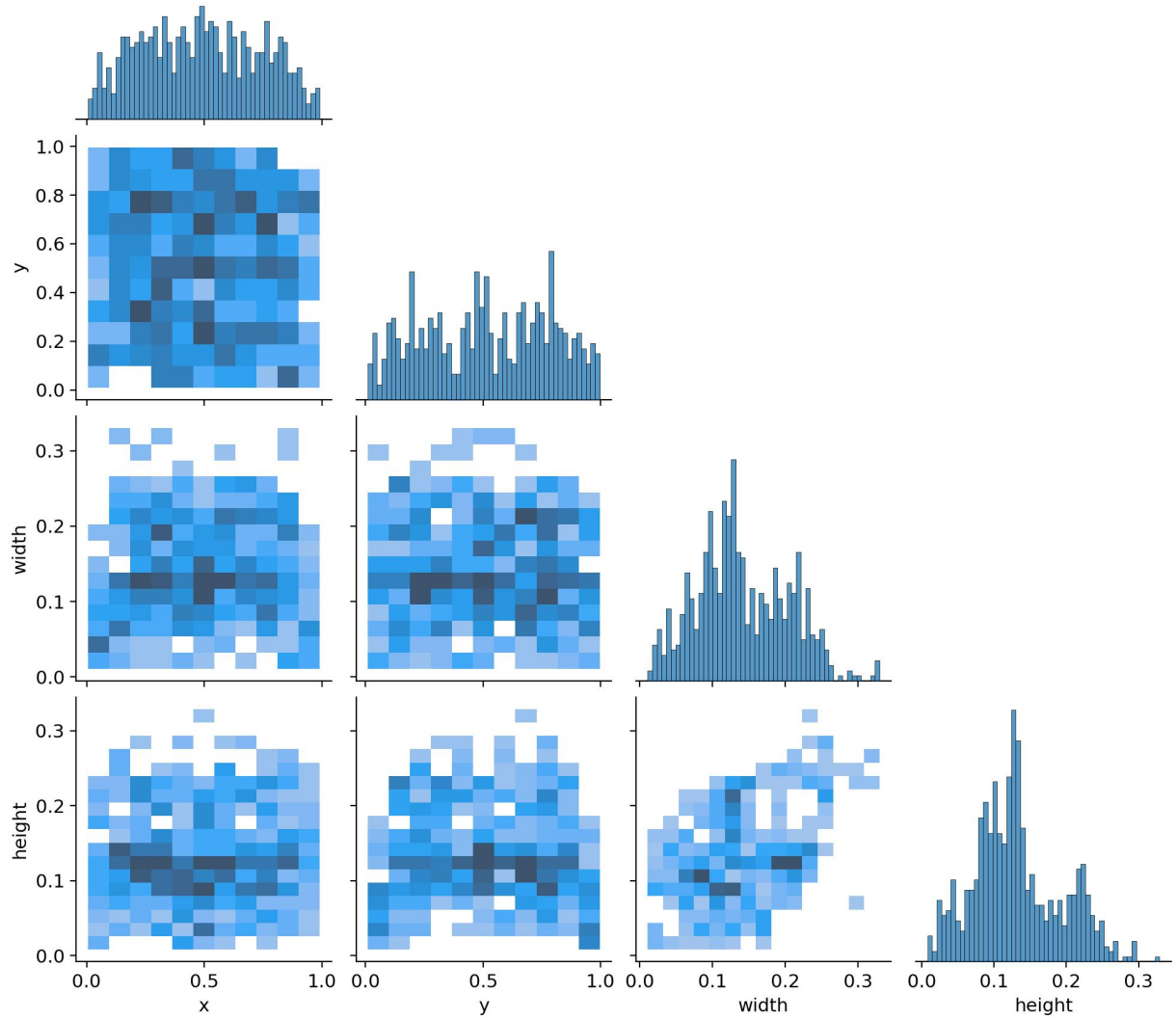


Figure I.1: Caption

Appendix J

Labels Dataset Distribution

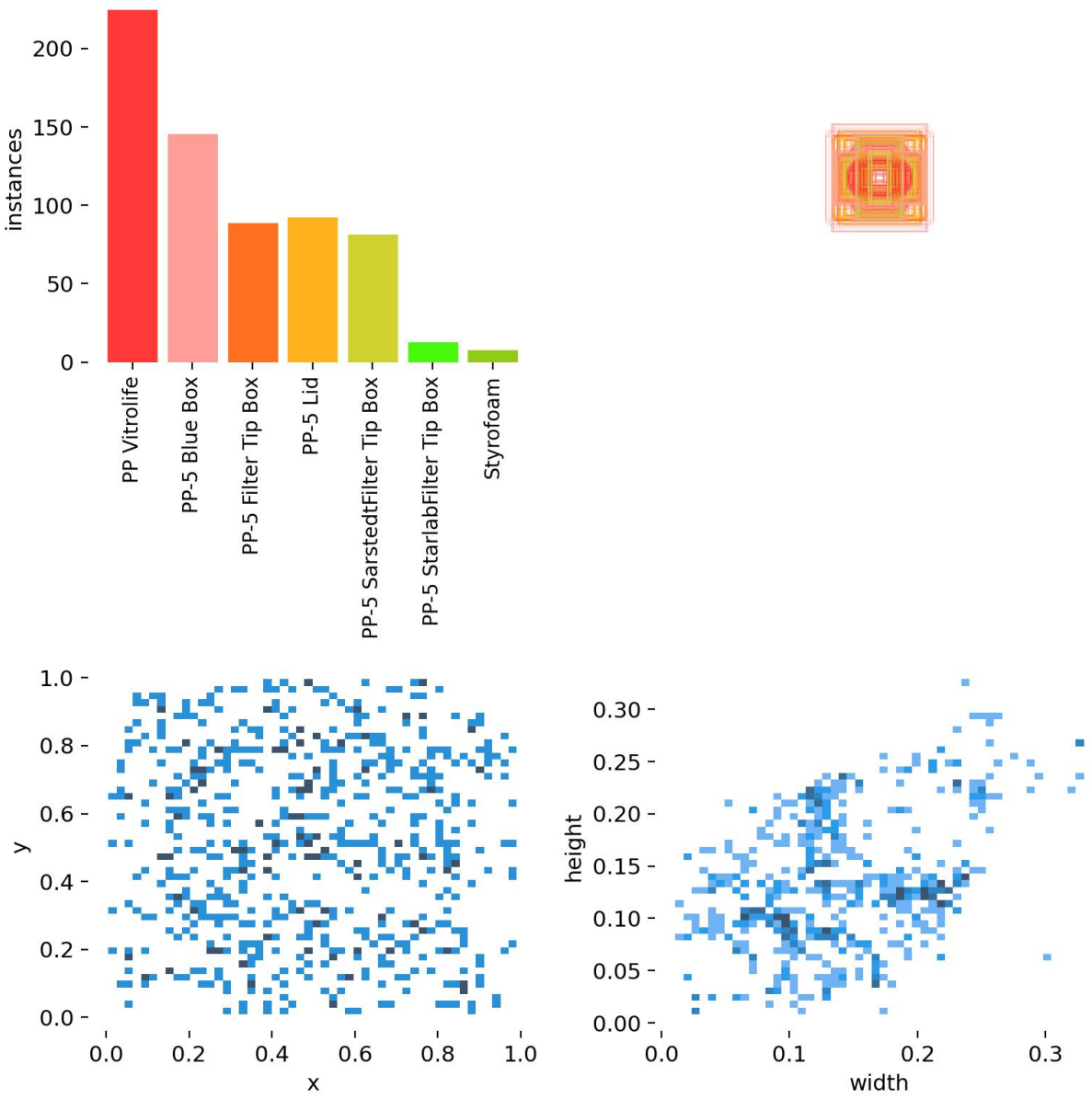


Figure J.1: Dataset Distribution of Labels, showcasing positioning and dimension attributes