MASTER'S THESIS 2023

Intelligent Robotic Systems for Quality Control

Marcus Nagy, Martin Lyrå

ISSN 1650-2884
2023-19

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

2023-19

Intelligent Robotic Systems for Quality Control

Smarta Robotsystem för Kvalitetskontroll

Marcus Nagy, Martin Lyrå

Intelligent Robotic Systems for Quality Control

(A Study on a Pick Skill for a Collaborative Robot)

Marcus Nagy
ma8828na-s@student.lu.se

Martin Lyrå
ma4542ro-s@student.lu.se

June 12, 2023

Master's thesis work carried out at Tetra Pak International S.A..

Supervisor: Volker Krüger, volker.krueger@cs.lth.se

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

This thesis investigates the application of a picking skill for intelligent robotic systems with the intention of quality control. Picking an object is a skill crucial for quality control yet non-trivial in robotics, requiring both sensing and actuation. Using ArUco markers, we established a baseline for accurate pose estimation.

This study emphasizes the necessity of a robust hand-eye calibration for precision in vision-based tasks.

The proposed picking skill, leveraging fiducial markers like adequately sized ArUco, demonstrated substantial efficacy and potential for enhanced reliability and versatility with suggested improvements. This allowed successful picking of variously sized and oriented containers.

Keywords: MSc, Intelligent Robotics, Skill, ROS, SkiROS, ArUco, Picking

Acknowledgements

We would first and foremost like to thank our supervisor Volker Krueger for the continuous support and guidance throughout the thesis work.

Secondly we would like to thank Jacek Malec for taking on the task of being the examiner.

Finally we would like to thank all the people working at *Robotics and Semantic Systems* LTH for all the help, especially Marcus Klang for *git* and technical aid, Matthias Mayr for supplying with implementation of the actuation part, Simon Kristoffersson Lind for continuous help with the use of *SkiROS*. Not to mention the co-authored implementation of ArUco marker detection and camera calibration between us and the other thesis project during a hackathon consisting of Josefin Gustafsson and Pontus Rosqvist.

From Tetra Pak, we extend our heartfelt thanks to Daniel Cederström. His support throughout our thesis work and his crucial role as a mediator between us and Tetra Pak are deeply appreciated.

Contents

Chapter 1
Introduction

Tetra Pak delivers a wide variety of solutions for both production and packaging of liquids into containers of different volumes. The material for the containers as such are delivered on rolls which are later cut, folded and glued together, as well as printed on.

The containers are designed to meet the needs and expectations of modern consumers while also addressing environmental concerns. They can be conveniently compressed and folded, making them easy to recycle and requiring less space. To maintain these standards, the containers undergo regular quality control assessments. Factors such as proper folding, print quality, and appropriate glue usage are evaluated to ensure that the containers are symmetrical, easy to open, and disposable.

The machinery needs occasional oversight, service checks, and upkeep to guarantee quality standards. At present, all procedures from container quality control to machinery supervision are performed by humans. However, there's a growing interest in utilizing automation and intelligent systems like robots for these tasks.

The aim of this thesis is to enhance quality control in container inspection by incorporating intelligent robotics. Our initial focus is on designing a reliable robotic pick skill, which will be essential for selecting individual sample containers for subsequent quality control analysis.

1.1 Intelligent Robotics for Quality Control

There has been an acceleration of converting traditional industrial processes into more safe and efficient environments as part of the *Industry 4.0* paradigm [19]. This has resulted in the sudden acceleration for industries that want to fill in the absence of automation and the use of intelligent systems.

The quality control of products often relies on manual labor, involving human intervention that can be susceptible to errors [33, 23].

Automation has proven to be a viable option for quality control and process inspections over the past two decades [12]. Not only can it reduce errors prone to human intervention, but it can also decrease labor costs [12]. Furthermore, automation can potentially increase production rates [12, 23, 33].

1.2 Problem statement

The aim of this project is to implement an intelligent robotic system to automate the process of picking boxes from a production line. The challenge involves designing a collaborative robot capable of safely navigating an environment shared with human workers and picking boxes from a conveyor belt. The boxes vary in size and shape, requiring the robot's picking mechanism to be highly flexible and adaptable. As a limitation, we will evaluate our picking mechanism solely on stationary boxes.

The scope of this project is to look into the components of box picking by an intelligent robot and discuss these briefly:

- **Perception**: An intelligent robot must be capable of 'recognizing' objects or locations. This involves not only object detection, but also overcoming challenges like camera calibration and hand-eye calibration, as well as estimating object pose.
- **Knowledge**: An intelligent robot must comprehend what objects are and where specific locations lie. This refers to a robust understanding of the world, an essential component of intelligent robotic tasks.
- **Actuation**: This covers aspects related to the execution of the pick skill. This includes maneuvering the robot arms through a series of movements that allow for object detection, approach, secure grasping, and finally, returning to the point of origin.

We look at how a skill should be designed with these three components in mind, and identify challenges associated with the task. To streamline the project, we have decided to perform the picking process when the box is stationary and placed on a table. We will initiate the project by using ArUco markers for pose-estimation [36]. Recognized as a standard type of fiducial markers, these are reference points located on objects or locations, characterized by patterns that can be easily computed and detected in an image.

In short, the main objectives of the task are:

- Detect location and pose.
- Pick an object.

- Develop strategies to accommodate variations in box dimensions and configurations.

With a literature study, we look into what others have accomplished in the field, how they applied intelligent robotics to industry, and how classic *Industry 3.0* robotics have been enhanced.

1.3 Research Questions

We can now summarize the discussion above to these following research questions:

1. How can a robot skill be designed to reliably pick up a variety of Tetra Pak boxes repeatedly?

In particular:

2. How do we get the orientation of the target object based on ArUco markers? How does the orientation affect the picking?

3. What design considerations should be taken into account to ensure that the skill allows for subsequent inspection of picked items?

4. In order to pick up multiple Tetra Pak boxes of different dimensions, where and how do we implement the necessary knowledge?

1.4 Research Plan

In order to develop an effective intelligent autonomous system for quality control, the research plan will encompass four key stages that address object detection and pose estimation, calibration processes, picking strategies, and system evaluation and optimization. These stages can be summarized as follows:

- **Investigate detection and pose estimation techniques:** To effectively analyze the quality of objects, it is essential to accurately detect and estimate their pose. The initial stage of this research will focus on identifying suitable methods for achieving this, with an emphasis on building a baseline model using fiducial markers. This baseline will serve as a benchmark for comparison as we explore alternative approaches.

- **Address camera and hand-eye calibration prerequisites:** To ensure precise pose estimation of the objects, it is necessary to consider the camera and hand-eye calibration processes. These steps are crucial prerequisites for enabling the subsequent picking phase of the project.

- **Develop a robust picking strategy:** The next step in the research plan involves devising a method for grasping objects irrespective of their shape or initial orientation. A reliable picking strategy is required to facilitate the inspection and assessment of the objects.

- **Evaluate and optimize the overall system:** The final stage of the research plan involves conducting a thorough evaluation of the proposed intelligent autonomous system for quality control. This will include assessing the performance of the detection, pose estimation, calibration, and picking components, as well as optimizing the system to ensure optimal performance. The results of these evaluations will provide valuable insights into the effectiveness and potential applications of the developed system in real-world quality control scenarios.

1.5 Related Work

The oyster-mushroom picking robot by Y. Qian et.al. [30], with real-time detection and localisation using machine learning serves as an insight into the design of a picker robot.

A recent work, engineering an automatic tomato-plant planter, by I. Yang et.al [37], forms the basis for a solution using ArUco for pose estimation. The introduction to ArUco markers in Section 2.2 is based on the article from 2013 [7].

The collaborative robot 'Flippy2' developed by *Miso Robotics* has begun to be widely used by three major fast food chains in the United States as a means to provide better safety and quality, and its purpose is to meet the demand for automated goods in the industry [4, 18]. "Flippy2" relies on ArUco markers for estimating pose and identifying what goes where, as well as associating cooking timers with markers[4].

An abstract proof-of-concept project in actuation, which explicitly used ArUco to estimate the pose of marked objects for picking [13], serves as the primary foundation for the basic steps of picking as a skill.

Pose estimation in the field of bin picking has been explored using various methods. One recent approach suggests utilizing 3D matching [11], while another proposes using 6D poses extracted from RGB-D inputs [2]. Other techniques that have been previously employed include 2D matching with 3D object recognition by searching for maximum similarity, using a laser scanner system to determine pose by detecting horizontal and vertical lines, utilizing simulation to learn grasping poses that can later be transferred to the robot's end effector, and leveraging deep machine learning for object detection. Additionally, some researchers have utilized 3D point-cloud data and 3D CAD models [11]. Although there are state-of-the-art commercial systems available, such as FANUC's 3D Area Sensor [6] and Pick-itTM Robotic Picking [29], these solutions can be quite costly and complex to implement.

1.6 Contribution

The insights derived from our research aim to significantly contribute to the field of intelligent robotics, particularly in terms of a pick skill for quality control. Our primary contributions include:

1. Developing a robust robotic skill capable of reliably picking up a diverse range of Tetra Pak boxes repeatedly.

2. Establishing a methodology to determine the orientation of a target object using ArUco markers, and understanding the impact of this orientation on the picking process.

3. Identifying key design considerations that ensure the developed skill facilitates subsequent inspection of picked items.

4. Proposing strategies for equipping the robot with the necessary knowledge to handle Tetra Pak boxes of varying dimensions.

These contributions are designed to advance our understanding and utilization of robotics in quality control and inspection processes.

Chapter 2
Theory

To address the research questions outlined in 1.3, we delve into the theory and background of several essential areas. In the realm of robotics, the comprehension of transformation between different frames or coordinate systems is essential. We first explore this concept.

Next, we focus on object detection, a task that can be achieved through a variety of methods, one of which includes the use of ArUco markers. Despite the susceptibility of advanced detection methods to environmental factors like lighting and reflection, we can employ robust coordinates of markers, such as ArUco, to estimate the object's pose [26].

Once the desired object's pose is computed, any picking skill requiring an object's pose can execute the vision-based task. Ensuring that the pose estimation is precise and consistent is critical, leading us to examine calibration as our third focus area. We will start with camera calibration and proceed to hand-eye calibration.

In conjunction with these aspects, actuation plays a significant role in system operation. Manipulating the robot arm requires appropriate control, tailored to the intended working environment. To reduce the risk of material damage or personal injury, we incorporate a compliant controller, enhancing the robot arm's responsiveness to external forces. The strategies involve position-based negative feedback or impedance control, with the latter being our preference for compliant control.

Lastly, we delve into ontology, behavior trees and SkiROS, studying its practical application in picking containers of various sizes.

2.1 Transformations in Coordinate Systems

In three-dimensional space, any point $p \in \mathbb{R}^3$ can be characterized from various frames of reference. These frames provide unique coordinate systems that a robot can leverage for navigation. A point's location within these systems isn't strictly tied to specific values. Instead, it can also be interpreted relative to a predefined coordinate within a given frame [21, 14]. As exemplified in Figure 2.1, a point is defined in frame **A** and the same point is illustrated in reference to frame **B**.

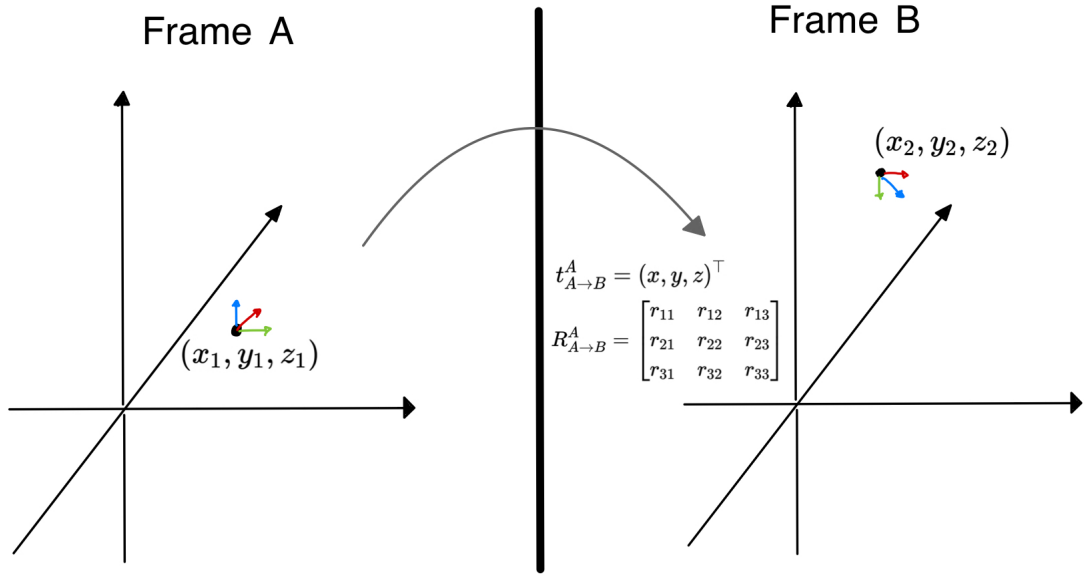


Figure 2.1: Illustration of point in frame **A** and its representation in frame **B**.

Figure 2.1 showcases a point p_1 in frame **A**, represented by coordinates (x_1, y_1, z_1). The same point is expressed as p_2 in frame **B**, with different coordinates (x_2, y_2, z_2). It's crucial to understand that p_1 and p_2 are the same point p referenced in different frames, **A** and **B**. Transitioning from one frame to another involves capturing the translation $t^A_{A\to B}$, which determines the placement of point p when transitioning from frame **A** to frame **B**. Similarly, the rotation from frame **A** to **B** is recorded as $R^A_{A\to B}$, where each column represents the orientation for the x, y, and z axes.

To calculate the coordinates of point p when transitioning from frame **B** to **A**, one can employ the following formula:

$$p_1 = R^A_{A\to B} p_2 + t^A_{A\to B} \tag{2.1}$$

This can also be represented as a matrix equation (refer to Eq. 2.2):

$$\begin{bmatrix} p_1^\top \\ 1 \end{bmatrix} = \begin{bmatrix} R^A_{A\to B} & t^A_{A\to B} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_2^\top \\ 1 \end{bmatrix} \tag{2.2}$$

The rotation and translation together constitute a transformation $T_{A\to B}$. Hence, it is not necessary to describe each point in every frame. It is sufficient to understand the point's relation to a specific frame and how this frame relates to all other frames.

A transformation matrix is utilized to map the points from the source coordinate system to the target coordinate system. The general structure of a transformation matrix for converting coordinates from one system to another can be represented as:

$$T_{A\to B} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Here, the elements in Eq.(2.3) r_{ij} form the rotation matrix, which accounts for the orientation change between the two coordinate systems, and t_x, t_y, and t_z represent the translation components, which describe the displacement between the systems. However, the rotation matrix in the Robot Operating System (ROS) is typically represented as a quaternion due to its compact and reliable nature [17]. The quaternion, denoted as $\mathbf{Q}$, consists of four components: x, y, z, and w [17]. This representation necessitates the conversion between the rotation matrix $\mathbf{R}$ and quaternion $\mathbf{Q}$, and vice versa [1].

2.2 ArUco

An ArUco marker is a square binary fiducial marker, patterns used as reference points to be seen in an image. Like other fiducial markers, they are used for identification and pose estimation. The markers come in sets of predefined dictionaries; each contains a unique integer for the patterns. The integer is known as the identifier, or ID, which has a corresponding pattern. ArUco was proposed and launched in 2014 as an initiative to create fiducial markers capable of producing unique patterns that could not be mistaken for others [7]. This earned ArUco a reputation for reliability compared to other options available at the time.

The development of ArUco also included the release of an open-source library, free for everyone to use [7], allowing everyone to immediately pick up and use ArUco for pose estimation [37]. ArUco's identification, via the pattern's associated integer ID, and its robustness in pose estimation makes it relevant for our work. Details will be elaborated later on in methodology.

2.3 Pose estimation

Pose estimation serves several distinct, yet functionally related applications, each with goals revolving around different poses. These applications include estimating the trajectory between two known poses, determining the joint-states of a pose to meet specific requirements, and identifying the global location of objects perceived by a robot. While these applications have distinct objectives, they often employ the same mathematical solvers due to their shared functionalities.

The first forms the backbone of robot actuation and where the solvers calculate an optimal trajectory from one pose to the end pose [10]. Some solvers account for the robot's spatial awareness for obstacles, some do not and only find the most straightforward path.

The second one is often called *inverse kinematics* and is one of the major everyday problems with robots, since almost all tasks have variable locations that change over time. This task requires more computing since the inverse kinematics involve inverses of matrices and several accounts of matrix multiplication, and the degrees of freedom depends on how many joints the robot limb has.

The third is less associated with actuation but more with computer vision, another important topic in robotics. We wish the robot to have a way to see and locate an object, the so-called object localization. Once we find the coordinates of an object, the depth image of the same image is used to calculate the world frame of the object that contains the position and orientation. As mentioned in 1.2 the way an object can be localised can be accomplished in different ways where ArUco marker is a simple way for computer vision to localise the object or a location an ArUco marker is placed on.

2.4 Camera calibration

Camera calibration is the process of determining the parameters that influence how the camera projects the real world onto the image plane. These parameters include the image center, focal length, and lens distortion. Obtaining these values allows for the alignment of the image space coordinates with the real-world coordinates as seen by the camera. Furthermore, it enables the estimation of real-world coordinates from points within two or more images. Consequently, calibration is a critical step in computer vision and image processing, particularly when determining the location of an object is necessary.

Computing the calibration yields the camera matrix $\mathbf{K}$. Each point in the image corresponds to two equations (one for x and one for y), necessitating at least four known points to compute $\mathbf{K}$. Checkerboards are a common example of objects with known dimensions and properties, making the calibration process straightforward [25]. $\mathbf{K}$ is defined as:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

In the $\mathbf{K}$ matrix, f_x and f_y represent the camera's focal lengths, while c_x and c_y denote the optical coordinates of the camera's center [25]. These variables are also referred to as intrinsic parameters of the camera. To transform a 3D point's coordinates into the image coordinate system, extrinsic parameters are also required [25]. The intrinsic parameters are unique to each camera but can change over time [15]. The intrinsic parameters usually come as factory settings that can easily be recovered but are not necessarily accurate, on the other end the distortion coefficient are not provided and have to be calculated [25].

To correct the distortion introduced by the camera's lens, it is necessary to use a known object with well-defined points, such as the aforementioned checkerboard. Knowing the points' coordinates in both the real world and the image enables the computation of the distortion coefficients, which are defined as [15]:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{2.5}$$

To obtain accurate coefficients, a minimum of 10 test patterns (images) is typically required, although using more could potentially yield better results [25]. However, beyond a certain point, additional images do not significantly improve the results . Some cameras come with on-chip calibration that uses depth sensors to determine if degradation has occurred. Another approach to verify camera integrity involves comparing the on-chip calibration results with those obtained using an external reference, such as a checkerboard pattern [9]. Recalibration is generally unnecessary unless the camera has been subjected to physical shocks, such as high vibrations, knocks, or drops [9].

The intrinsic and extrinsic matrices together constitute the *projection matrix*. Typically, the *projection matrix* is used as follows:

$$ImagePoint = \mathbf{K} \times \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \times \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.6}$$

In the last equation, X, Y, and Z represent the real-world coordinates of a 3D point in the object's coordinate system. By applying the *projection matrix* to these coordinates, the corresponding image point in the image plane can be determined, enabling the transformation of real-world coordinates into image coordinates.

2.5 Hand-Eye Calibration

To get the unknown transform between the "eye" (camera) and the "hand" (gripper) of the robot there are two ways of viewing the problem, either the camera is in a fixed observing position or it is connected with the gripper, the end effector [15], shown in Figure 2.2.

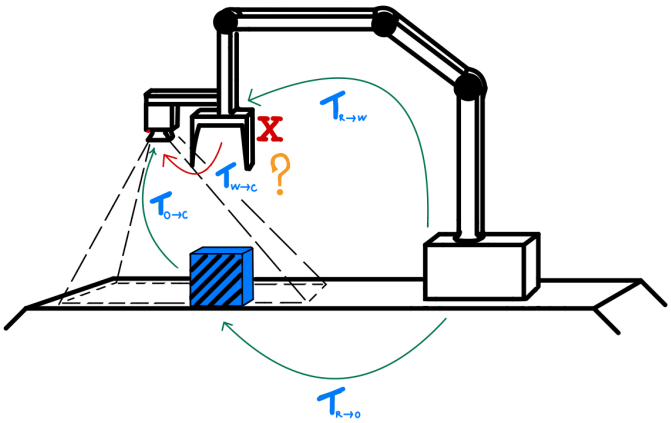


Figure 2.2: This image demonstrates the computation of the transformation $T_{W\to C}$ through Hand-Eye calibration. It is calculated using the transformation's $T_{R\to O}T_{O\to C}$ and $T_{R\to W}$. These transformations are fundamental to ensuring accurate robotic perception and actuation.

The matrix equation typically used to address the problem in Figure 2.2 is represented as $T_{R\to W}X = T_{R\to O}T_{O\to C}$, which is commonly referred to as the $AX = YB$ problem. Obtaining the transformation matrix is a non-trivial task and is accomplished through the process known as *Hand-Eye Calibration*.

Hand-eye calibration is a crucial step in determining the relationship between an object's location in the robot's coordinate system and the transformations between them [15]. The kinematic chain and joint settings can be computed from the *Unified Robot Description Format* (URDF) [20]. In Figure 2.2, the camera is denoted as C, the robot wrist as W, the object as O, and the robot base as R. The goal is to obtain the transformation $T_{W\to C}$, which requires solving the following equation:

$$T_{R\to W}X = T_{R\to O}T_{O\to C} \tag{2.7}$$

This corresponds to the problem shown below:

$$\begin{pmatrix} R_{T_{R\to W}}R_X & R_{T_{R\to W}}t_X + t_{T_{R\to W}} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{T_{R\to O}}R_{T_{O\to C}} & R_{T_{R\to O}}t_{T_{O\to C}} + t_{T_{R\to O}} \\ 0 & 1 \end{pmatrix} \tag{2.8}$$

In this equation, all components are known except for the transform X. Solving Eq. (2.8) involves capturing multiple images to determine the object's pose within the camera's frame, with a minimum of two distinct positions. Ideally, each position should be significantly different from the others [35]. Alternatively, the Y component can be estimated as part of the $AX = YB$ problem, as described in *Continuous Hand-Eye Calibration Using 3D Points* [8]. Generally, estimating the Y component is avoided due to its cumbersome nature. However, it carries the added benefit of continuous calibration during the execution of other tasks. For instance, should the camera become displaced suddenly during execution, the continuous

estimation of the $\boldsymbol{Y}$ component and consequent error reduction would allow for adaptability to the abruptly changed environment.

2.6 MoveIt

MoveIt Motion Planning Framework is the default motion planning software bundled with official ROS distributions. Its features are not just limited to motion planning but cover the following aspects [31]:

- Gripper manipulation
- Inverse kinematics
- Control theory actuation
- Collision checking
- Graphical user interface
- Deep integration with ROS

However, we will not be using MoveIt for our project, for performance and reliability reasons; instead, we will be using the motion planner mentioned in the forthcoming section. MoveIt serves as a fallback and reference if the previously mentioned method fails, and it is also used to fetch joint configuration data.

2.7 Cartesian Impedance Controller

Controlling the motion of a robotic system demands a strategic approach, particularly in the manipulation of robotic arms. At one end of the control strategy spectrum is the *position-based* negative feedback control. This approach is often rigid, with a high gain on small errors, leading to less flexibility in responding to environmental variations [5]. On the other end of the spectrum, *compliant control* strategies, such as model-based control with torque control, offer a more flexible alternative.

A Cartesian impedance controller belongs to the latter category, employing a control strategy designed to regulate motion in Cartesian space in response to applied forces and torques [22]. This type of controller is designed to establish a specific dynamic relationship between the motion of the robot and external forces [27]. In the context of a Cartesian impedance controller, this relationship is described in terms of the end-effector's coordinates, which depict its motion [27].

Unlike controllers that adhere to a pre-defined path, a Cartesian impedance controller reacts to external forces. For instance, if the robot arm encounters an obstacle or experiences any external forces, the controller can adjust the arm's position and orientation to compensate for these environmental variations. This makes the Cartesian impedance controller ideal for scenarios requiring delicate touch or interaction with uncertain environments. A prime example is a picking task involving objects of unknown size or shape. Its utility extends to collaborative robots operating in close cooperation with humans, where adaptability and safety are crucial [22].

2.8 Ontology

An ontology describes the *knowledge domain* of a robot. Basically, the knowledge domain is a database of graphs, relations, data models, and other technical schematics and properties of the data about the world [32, 34, 24].

Ontology enables robotics to abstract both hardware and software specifics when it comes to behaviors, actions, and skills. The goal is to make these elements independent of particular hardware and software details, relying instead on what a robot anticipates from itself, its environment, and other areas of interest. This abstraction falls within the domain of ontology. By doing so, ontology facilitates the development of features akin to mnemonics, cognitive abilities, and spatial awareness, which might otherwise be constrained by finite state machines and the constant need for bespoke solutions to unique problems [16].

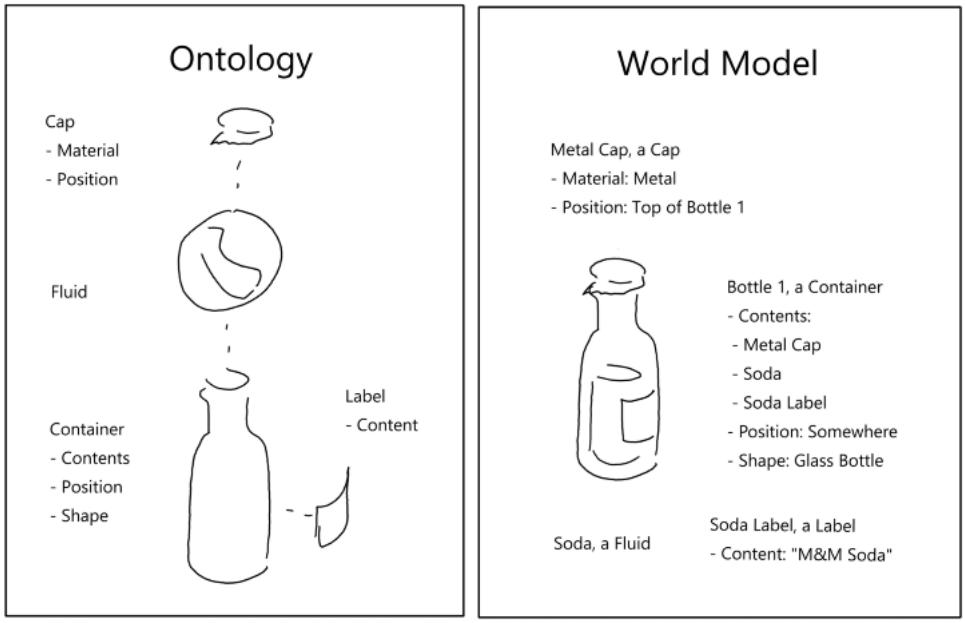


Figure 2.3: A sketch illustrating a knowledge domain modelled around an example; a soda bottle.

See the figure 2.3 illustrating an example of the responsibilities of ontology and world model. To left, the domain of the knowledge describing what can be known. To right, an instance of the knowledge complete with physical relationships and values assigned to attributes. In short; world model is "what do we know?", compare to ontology's "what can we know?".

There are various different schematics and plain-text languages that describe the robot's knowledge of the world. The ontology used in this work are written in Description Logic (DL), a first-order logic specialisation for the ease writing descriptions and proprieties of the data models.

2.8.1 Behaviour Trees

The Behaviour Tree (BT) is a directed acyclic graph structure commonly used to model the behavior of intelligent systems. Each node in the tree represents either an action or a specific behavior that can be exhibited. The BT allows for both modularity and reactivity of the system [3]. Originally used in computer game development, the BT was designed to control non-player units with a structure that allowed for efficient reuse of code [3].

In formal terms, the internal nodes of a BT are referred to as *control flow nodes*, while the leaf nodes are called *execution nodes* [3]. Each time step of execution is referred to as a *tick*, and *ticks* can have a specific frequency. An execution node can communicate one of three notions to its parent node: *running* (if the execution is ongoing), *success* (if the goal has been achieved), or *failure* [3]. The control flow nodes include *sequence*, *fallback*, *parallel*, and *decorator*, as shown in Table 2.1. The BT has been widely adopted in the field of robotics and AI as an alternative representation to finite state machines, as it has proven to be an effective tool for designing and implementing complex behaviors [3].

Node Type	Symbol	Succeeds	Fails	Running
Fallback	?	$\exists c \in C, S(c) = true$	$\forall c \in C, F(c) = true$	$\exists c \in C, R(c) = true$
Sequence	→	$\forall c \in C, S(c) = true$	$\exists c \in C, F(c) = true$	$\exists c \in C, R(c) = true$
Parallel	⇉	$\lvert\{c \in C, S(c) = true\}\rvert \geq \lvert M\rvert, M \subseteq C$	$\lvert\{c \in C, F(c) = true\}\rvert > \lvert C\rvert - \lvert M\rvert$	else
Action	*text*	Upon Completion	If impossible to complete	During completion
Condition	*text*	If true	If false	Never
Decorator	◊	Custom	Custom	Custom

Table 2.1: The different node types for a Behaviour Tree. C stand for the set of children to a parent, functions $(S)uccess : c \mapsto bool$, $(F)ail : c \mapsto bool$ and $(R)unning : c \mapsto bool$.

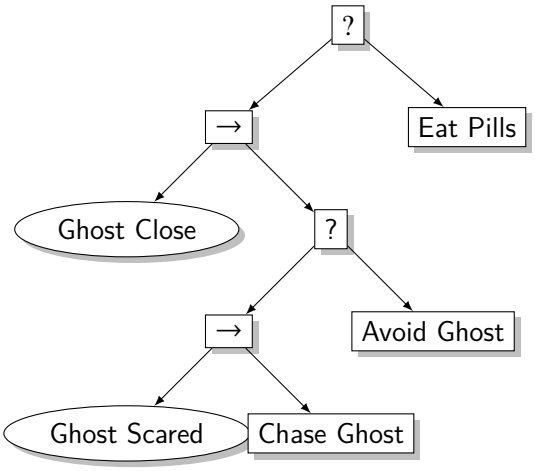


Figure 2.4: Behavior tree modeling a simple game of Pac-Man.

Figure 2.4 illustrates the utilization of some of the most prevalent node types in behaviour trees, as outlined in Table 2.1. This figure presents a simplified version of the well-known

game, Pac-Man, demonstrating the use of fallback, sequence, condition, and action nodes within its design.

2.9 SkiROS Skill-based Platform

The SkiROS framework, thoroughly integrated with ROS, is designed to manage robot behaviors through modular blocks, similar to behavior trees—an abstract model for executing tasks based on various conditions [34]. This tool is especially advantageous in environments with a substantial initial understanding, but it may encounter difficulties in unforeseen scenarios.

In such situations, SkiROS's flexibility allows the robot to adapt. However, certain challenges may arise, such as difficulties in managing complex, dynamic environments or integrating with other robotics software platforms. Specific attention and strategies are required to handle these circumstances effectively.

SkiROS encompasses several notable features:

- **World model**: This is a structure depicted as a resource description framework (RDF) database, a standard model for data interchange on the Web. It stores valuable environmental information that the robot can leverage [34].
- **Robot behavior designer**: SkiROS facilitates the design of behavior through Python code, allowing the robot to respond to changes during execution [34].
- **Skill system**: This system aids in organizing coded behaviors, promoting modularity and reusability. The behavior trees within SkiROS are assembled into units, called skills, consisting of one or multiple nodes. Each skill carries pre- and post-conditions that verify execution status at any given moment [34].

The implementation of skills can be structured into Python packages, which can be deployed and imported in other robots, functioning similarly to plugins.

A skill in SkiROS is classified as either a *primitive skill*—an atomic unit implementing code that affects the world model, or a *compound skill*—a composite modeling complex behaviors by combining primitive skills [34]. The capability to categorize skills as primitive or compound allows for greater adaptability and efficiency in coding and executing tasks.

Each skill in SkiROS includes a skill description detailing parameters (inputs and outputs of the skill, with the ability to apply conditions), pre-conditions (conditions required before execution), hold-conditions (conditions required during execution), and post-conditions (conditions required after execution). These elements provide a robust structure that guides the development and deployment of skills, ensuring a thorough validation process for successful task completion.

Chapter 3
Approach

Constructing a skill for picking up a Tetra Pak container seems intuitively simple when thinking from a human perspective, but when it comes to intelligent robotic systems, there are a few things to consider, three of which are sensing, actuation, and coordinate transfers.

For sensing, there are several options available, such as using fiducial markers or employing pure object detection techniques. The latter is often associated with deep learning methodologies that utilize bounding boxes, segmentation, and other tools. As outlined in Section 2.2, we propose the use of ArUco markers as a foundational element for our approach. In order to have a functioning sensing for the system it has to be ensured that the sensing is accurate. This requires two essential but different calibrations, *camera calibration* as described in 2.4 and *Hand-Eye calibration* in 2.5.

The implementation of the skill is described in Section 3.1. We explain the use of ArUco markers, how the skill makes use of the framework provided by SkiROS, and the overall skill design. Answering the following; what features are needed for the skill; what knowledge does the robot need; what does it expect from the environment.

3.1 Implementation

Overall, the robot, referred to as *Heron*, operates with a computer that runs the Noetic release of ROS 1 on Ubuntu 20.04. All the software code discussed in this thesis is implemented using Python 3. Major libraries include OpenCV for image processing and `librealsense` for supporting the Intel RealSense color depth camera used in this work. For defining the robot's actions as skills we used SkiROS, a framework for implementing Ontology and Skills concepts in ROS [32].

OpenCV plays a pivotal role in the acquisition and localization of objects, and it is particularly valuable for both camera and hand-eye calibration tasks; in this work, we will focus on ArUco markers and also on checkerboards for the calibration step.

3.1.1 ArUco Markers

ArUco markers are considerably simple, but reliable to use, and all the necessary functionality required to detect them for many systems is available out of the box. It will be considered the *baseline* method of computer vision - for object detection and localisation. The primary purposes of the ArUco markers will be following:

- Identification of a package type
- Calculation of a transformation that translates the ArUco marker's position to the geometric center of the object.

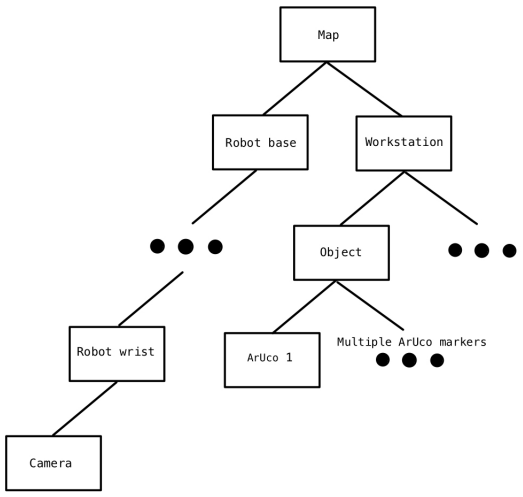


Figure 3.1: Transformation Tree.

(a) *Tetra Brik* 1000ml container.

(b) *Tetra Brik* 250ml container.

Figure 3.2: Figure 3.1 illustrates the tree structure of the transforms and their organization. Subfigures 3.2a and 3.2b depict *Tetra Brik* containers of 1000ml and 250ml, respectively, each featuring an ArUco marker in the bottom left corner.

For each of unique type or size of package, we assign an unique integer to the combination of type and size of package. Print out the associated marker pattern and then glue the marker to the *bottom-left* corner of a package, see Figure 3.2. The idea is that a package could be picked up in different orientations and to solve this, a unique marker will be placed on each side of the package as shown in the Figure 3.3, the tree structure in the world model will look as shown by Figure 3.1.

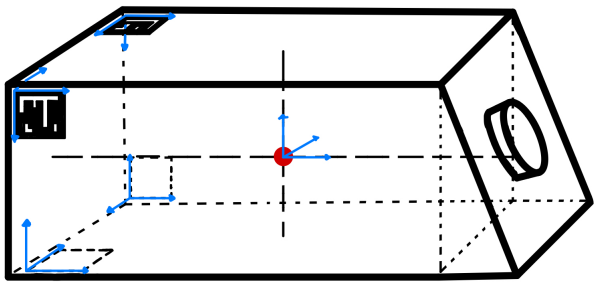


Figure 3.3: Displaying the placement of markers and their coordinate system.

Each ArUco marker encodes information about the transformation $\boldsymbol{T}_{O \to A}$, which represents the transformation from the ArUco marker to the geometric center of the object, as illustrated in Figure 3.4.

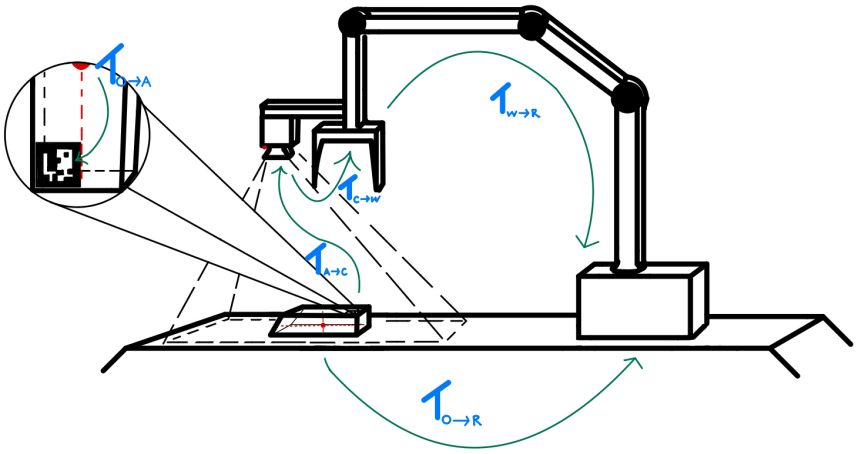


Figure 3.4: Transform from ArUco marker to object geometrical center, and complete simplified transform cycle.

This enables the construction of a complete kinematic chain from the object's geometric center to the robot frame, as shown in Eq. (3.1). Consequently, the robot can successfully pick up the object.

$$T_{O\to A}T_{A\to C}T_{C\to W}T_{W\to R} = T_{O\to R} \tag{3.1}$$

On the software side, the detection of the ArUco markers will be implemented using the OpenCV ArUco library. This functionality will be encapsulated within a SkiROS skill, and the picking skill will be integrated as a subsequent step

3.1.2 SkiROS - skills for ROS

Due to the complexity of the challenges of picking skills, it is necessary for them to be intelligent and flexible to the circumstances. We can design the solution as a reusable skill that can be applied to any robot. By leveraging the SkiROS framework [34] for implementing the necessary functions, actions, and knowledge, we can create a skill that is transferable between different types of robots. Following knowledge is needed for the picking skill:

- Types of Packages
  - Dimensions
  - ArUco Marker(s)
    - ID value
    - Location on package
    - Orientation
    - Dictionary used
    - Approach Pose
    - Grasp Pose
- Instances of above packages

3.1.3 Knowledge

There are two types of knowledge to be covered here:

- **Declarative knowledge** are the kind of knowledge that cannot be extracted or deduced from the live environment and must be defined first.

- **Procedural knowledge** covers the kind that the robot can deduce by itself from a live environment. Common example being the position-orientation poses of arbitrary objects.

Declarative knowledge will primarily consist of ArUco markers and specification about package types. An ArUco marker in the declarative knowledge contains information such as *BaseFrameID* which points to an object, pose which is physically measured by how it relates to the object, the marker's size and length. An object is declared to have a *BaseFrameID* which in our case is called *workstation*, a pose, and a list of ArUco markers. The *BaseFrameID* specifies the frame in which the object is defined. This, in turn, influences the object's position and orientation in relation to the *BaseFrame*.

The knowledge is written to a database file in a plain-text format called Turtle, short for Terse TDF Triple Language, a whole file is called a 'scene' in Skiros. This format allows the user to pick different files to save, load, or reload the world model and its description to a file, directly editable by a human.

3.1.4 Hand-eye calibration skill

The hand-eye calibration process involves the following components:

- An ArUco marker detector and estimator.

- A pose generator that produces stochastic points on a sphere surrounding a reference point.

- A 'Look-at' alignment mechanism that computes and rotates the random pose to face the reference point.

- OpenCV's hand-eye calibration module which utilizes Park's method [28], executed on N unique points.

The calibration process begins by running the marker detector to locate and store the reference point's position. Following this, the pose generator creates a new pose and utilizes the 'Look-at' alignment mechanism to orient the camera towards the reference point. Once the robotic arm has maneuvered its end-effector to the intended location, it waits until its velocity reaches zero, then captures an image and estimates a marker position.

If the estimated marker is both valid and distinct from previously recorded markers, it is saved in a buffer. This sequence of steps repeats until N points have been recorded. Following this, the camera transform can be calculated using OpenCV's hand-eye calibration module.

3.1.5 Design of Picking Skill

In the process of designing a picking skill, we realized that traditional methods such as hard-coded paths and finite-state machines prove to be inadequate. Therefore, we leveraged advancements in technology such as the Cartesian controller. Moreover, we found implementing ontology to be particularly beneficial as it enabled us to develop our desired functionalities as robot skills.

The design process involves several considerations which form the description of the skill:

- Declarative knowledge, what kind of knowledge do we have to declare first?
- Procedural knowledge, what kind of knowledge can the robot figure out on its own?
- What does the robot need for this skill? E.g. does it have a gripper?

The fundamental concept of the design involves the robot's end effector, equipped with a vision system, first moving to a predetermined detector pose. This pose initially provides a rough approximation of the object's location, prompting the system to re-estimate the object's actual pose. It's crucial to carefully define the detector pose. Despite its initial lack of precision, it should be positioned so the marker remains detectable, thereby ensuring the successful execution of the pick-and-place operation.

3.2 Evaluation Procedure

We designed an automated evaluation procedure for hand-eye calibration, ArUco marker pose estimation, and picking. Our goal was to minimize human interaction and manual intervention during these processes.

3.2.1 Hand-Eye Calibration Evaluation

The evaluation of hand-eye calibration proceeded as follows:

1. **Position the robot's arm** in a lookout pose, either automatically or manually.
2. **Place an object**, such as an ArUco marker, whose pose can be estimated.
3. **Reposition the arm** to a location on a sphere of fixed radius from the object, ensuring a random orientation while maintaining focus on the object.
4. **Estimate and record the pose** of the object.
5. **Repeat** steps 3 and 4 for N iterations.

This procedure was performed twice: initially without hand-eye calibration, and then repeated after hand-eye calibration.

3.2.2 ArUco Pose Estimation Evaluation

To evaluate ArUco marker pose estimation, we adapted parts of the hand-eye calibration evaluation procedure. The steps are as follows:

1. **Position the robot's arm** in a lookout pose, either automatically or manually.
2. **Place an object marked with an ArUco marker**.
3. **Reposition the arm** to a location on a sphere of fixed radius from the object, ensuring a random orientation while maintaining focus on the marker.
4. **Estimate and record the pose** of the ArUco marker.
5. **Repeat** steps 3 and 4 for N iterations.

Although the size of the ArUco marker varied between trials, the distance from which the pose was estimated remained constant.

3.2.3 Pick Skill Evaluation

The pick skill was evaluated using the following procedure:

1. **Position the robot's arm** in a lookout pose, either automatically or manually.
2. **Place an object**, such as an ArUco marker, whose pose can be estimated.
3. **Start a timer**.
4. **Estimate the pose** of the object.
5. **Initiate the pick skill**, completing all steps up to and including grasping, then returning to the lookout pose.
6. **Stop the timer** and record the time, then reset the timer for the next step.
7. **Initiate the placement sequence**, returning the object to its original position and returning the arm to the lookout pose.
8. **Stop the timer** and record the duration of the skills.
9. **Repeat** steps 3 to 8 for N iterations.

The lookout pose was determined based on the orientation of the object, whether standing vertically or lying horizontally. This procedure was repeated for different types of packages and orientations.

Chapter 4
Experimental Setup

The experimental setup comprises the following components:

- ArUco markers
- Tetra Pak containers
- Environment description
- Heron robot
- Intel® RealSense™ D400 camera
- UR5e collaborative robot arm
- MiR100 mobile base
- WSG-50 Gripper

ArUco markers serve the dual purpose of camera calibration and pose estimation for Tetra Pak containers. The environment is described through a knowledge-base or world model. The Heron robot is a composite of various robotic actuators, including the UR5e robotic arm, MiR100 base, and WSG-50 Gripper. The Intel® RealSense™ D400 camera is mounted on a 3D-printed fixture between the end effector link and the end effector.

4.1 The robot's workspace

The workspace includes four different Tetra Pak Aseptic samples, two of them have a 20mm ArUco marker pasted to the bottom left corner of the package, and on the other two the larger container has a 70mm marker and the smaller 35mm. The marker IDs are unique to the package's dimensions. When the packages are put in a testing environment, they are either laid on their side or standing.

The reasoning behind the sizing of the ArUco markers was twofold, we aimed to evaluate the impact of using a single marker in various sizes, while simultaneously ensuring that the containers weren't excessively cluttered with ArUco markers.

4.2 Physical evaluation

We evaluate our thesis work by conducting experiments directly on the robot in a physical setting.

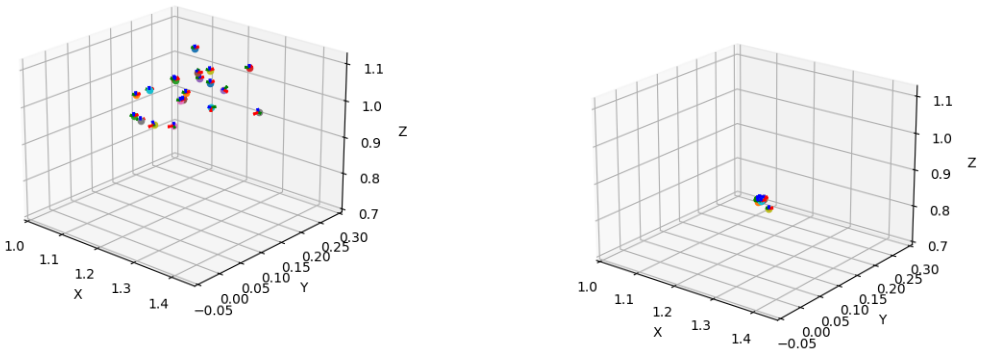
Initially, we intended to conduct a quantitative analysis using the Robot Operating System's (ROS) simulation environment, specifically Gazebo simulator. However, we discovered that the simulator was unable to meet our testing requirements in the following aspects:

- **Cartesian Trajectory control did not work in simulation**, since it is the fastest actuation controller and our skills require it, it was not an option to change to another one. The scope of addressing this issue was estimated to be too late and not within the scope of this project.
- **Generation of ArUco markers on objects within the simulation was not easily doable.** Not being able to test our fiducial markers in the simulated environment prevents testing this project's focal features. To integrate the models into Gazebo, they would need to be scanned and added. However, even if ArUco markers are available in Gazebo, they couldn't be attached directly to an object. Instead, they would need to be statically placed on the object's surface.

Chapter 5
Results

To validate the skill's reliability, we initially assessed the hand-eye calibration aspect and compared the outcomes with and without it. All results following hand-eye calibration were evaluated post a proper calibration. To evaluate the ArUco marker pose estimation, we conducted continuous pose estimations at a 45cm height, generating 20 poses in a sphere around a reference point. For the picking skill, we established a baseline by repeatedly picking the object from a single orientation and position. Subsequently, we investigated the influence of object orientation and position on the skill's reliability and performance.

5.1 Hand-Eye Calibration



(a) Pose estimates without hand-eye calibration.

(b) Pose estimates with hand-eye calibration.

Figure 5.1: Three-dimensional visualization of 6D pose estimation incorporating hand-eye calibration versus the uncalibrated approach.

The outcomes of hand-eye calibration demonstrate enhanced and more dependable pose estimations following the calibration procedure, as depicted in Figure 5.1. In the absence of hand-eye calibration, the consistency of pose estimation is low.

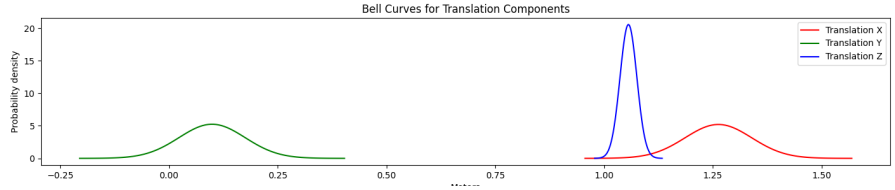


Figure 5.2: Dispersion of Translation without hand-eye calibration.

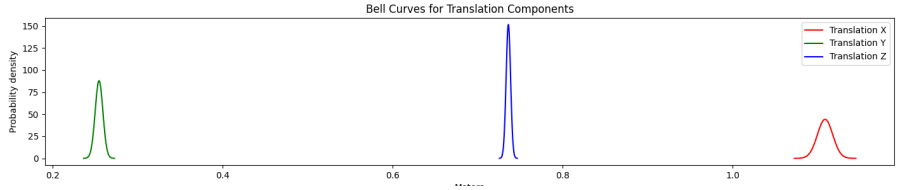


Figure 5.3: Dispersion of Translation with hand-eye calibration.

As illustrated in Figure 5.2, 5.3, the employment of hand-eye calibration leads to the concentration of x, y, and z coordinates around their respective means, consequently reducing

the dispersion of the distributions. It can be noted that the variances are much higher without proper calibration.

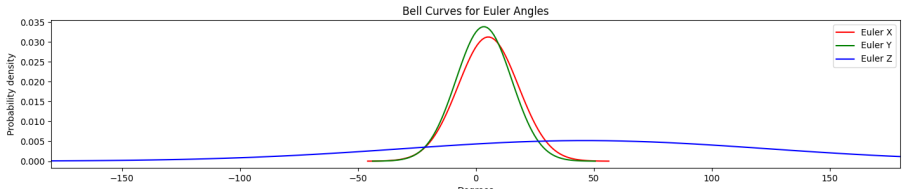


Figure 5.4: Dispersion of Euler angles without hand-eye calibration.

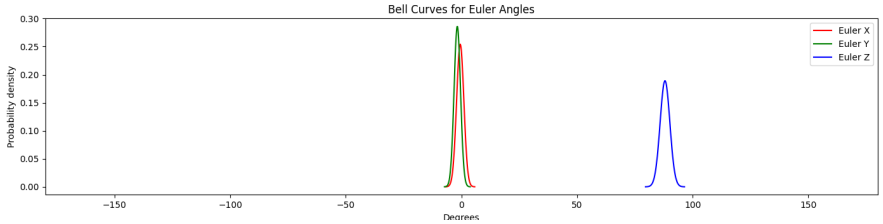


Figure 5.5: Dispersion of Euler angles with hand-eye calibration.

Similar to the translation illustrated in Figure 5.2, 5.3, the Gaussian distributions in Figure 5.4 and 5.5 display the dispersion of Euler angles with and without hand-eye calibration, respectively. These figures reveal that incorporating hand-eye calibration leads to a substantial improvement in consistency, as evidenced by the stable angular values maintained throughout the process.

Translation	No Hand-Eye calibration	Hand-Eye calibration
	σ *(m)*	σ *(m)*
x	0.076	0.009
y	0.076	0.004
z	0.019	0.002

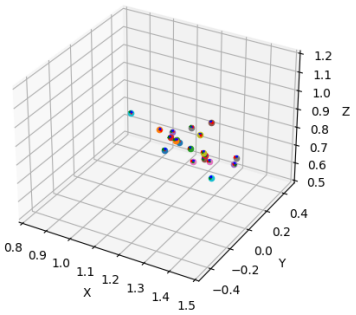
Table 5.1: Standard deviation values (σ) for translations along the x, y, and z axes in meters, comparing results with and without hand-eye calibration implementation.

Quaternions	No Hand-Eye calibration	Hand-Eye calibration
	σ	σ
x	0.117	0.010
y	0.074	0.015
z	0.589	0.014
w	0.108	0.012

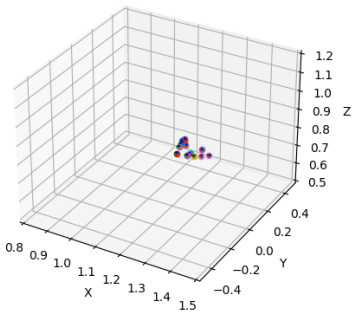
Table 5.2: Standard deviation values (σ) for quaternion components (x, y, z and w), comparing results with and without hand-eye calibration implementation.

The variances for both translation and rotation, as presented in Tables 5.1 and 5.2, demonstrate a significant reduction in the dispersion of the standard deviation, indicating improved consistency and precision in pose estimation.

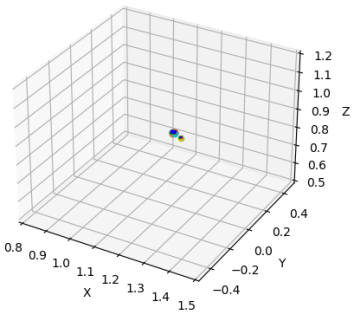
5.2 ArUco Marker Pose Estimation



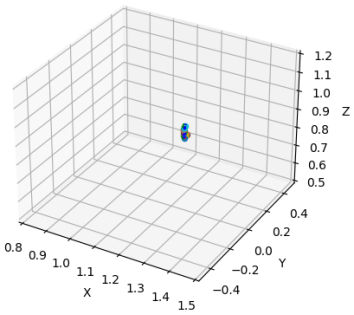
(a) Pose estimation of a *Tetra Brik* 1000ml with 20mm ArUco marker.



(b) Pose estimation of a *Tetra Brik* 250ml with 20mm ArUco marker.



(c) Pose estimation of a *Tetra Brik* 1000ml with 70mm ArUco marker.



(d) Pose estimation of a *Tetra Brik* 200ml with 35mm ArUco marker.

Figure 5.6: Three-dimensional visualization illustrating the varied outcomes of pose estimation on *Tetra Briks* of distinct sizes, employing ArUco markers of varying dimensions.

The pose estimation employing ArUco markers reveals the necessity for substantial marker size to achieve precise estimation during detection, ensuring a valid pose for object picking. The size of the ArUco marker is directly proportional to the distance from which it can be be detected with low error. Note that these measurement were done post hand-eye calibration.

Thus, while the hand-eye calibration underpins the system's general accuracy, the specific pose estimation is primarily determined by the ArUco marker's size and distance from the detector (camera).

Translations	20mm ArUco (1000ml) σ	20mm ArUco (250ml) σ	35mm ArUco (200ml) σ	70mm ArUco (1000ml) σ
x	0.1165	0.0340	0.0041	0.0090
y	0.0949	0.0536	0.0102	0.0045
z	0.0317	0.0267	0.0125	0.0026
Quaternions				
x	0.1428	0.1223	0.0731	0.0103
y	0.1407	0.1497	0.1307	0.0151
z	0.7195	0.6186	0.0129	0.0137
w	0.3241	0.2124	0.0189	0.0123

Table 5.3: Standard deviation values (σ) for translations (x, y, z) in meters and quaternion components (w, x, y, z) in pose estimation of different-sized containers marked with ArUco of varying sizes. The table compares results for a 1000ml container marked with 20mm and 70mm ArUco, a 250ml container marked with 20mm ArUco, and a 200ml container marked with 35mm ArUco.

5.3 Implementation and Performance Analysis of the Pick Skill

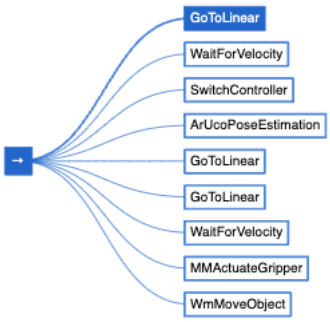


Figure 5.7: Skill implementation of the pick skill.

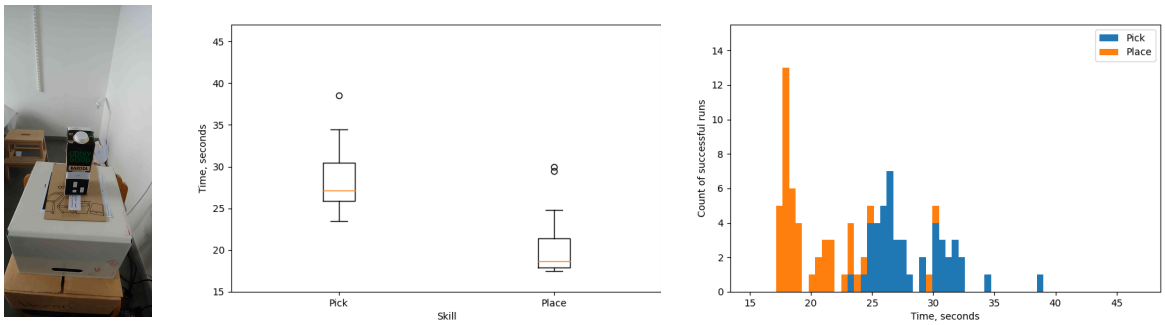
Figure 5.7 presents the behavior tree for the pick skill (referred to as a *SkillBase* in SkiROS), which comprises a series of actions, interspersed with condition checks for velocity and pose estimation.

- `GoToLinear`: This skill is tasked with planning and moving the end effector in compliant mode to a predetermined location in Cartesian space.
- `WaitForVelocity`: This skill ensures all the joints maintain a velocity under a certain threshold.

- `SwitchController`: This skill facilitates the transition between *compliant* and *joint_config* modes.
- `ArUcoPoseEstimation`: This skill captures a series of images at a predetermined frequency and computes the average estimation of the marker's pose.
- `MMActuateGripper`: This skill actuates the gripper, moving its tips to a certain offset from the gripper's center point.
- `WmMoveObject`: This skill ensures that the pose of the picked object is updated in relation to the gripper after pick up.

Each skill mentioned above plays a vital role in the successful completion of a picking sequence. We bundle these skills together to streamline the action sequence. It's crucial to understand that multiple sub-skills operate concurrently within each primary skill. For instance, while executing the `WaitForVelocity` skill, the system continually monitors joint velocities until they reach a specific threshold. This uses the foundational primitive skill, `Wait`, to perform the waiting. Though not explicitly illustrated here, these sub-skills significantly influence the successful execution of the main skill.

5.3.1 Baseline - Tall package standing up



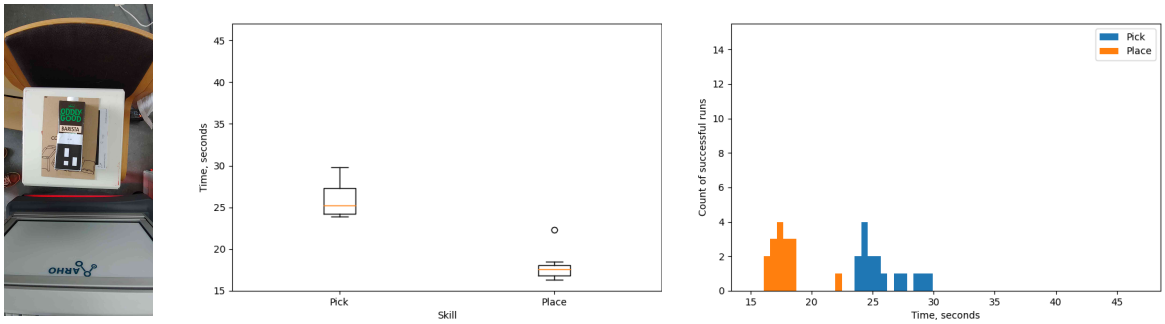
(a) (b) (c)

Figure 5.8: Plots for 50 points measured for the baseline; package standing up with marker side facing the robot. (a) Position. (b) Box plot. (c) Stacked histogram.

The Figures 5.8b and 5.8c show the result of 50 different measured points. These are the *baseline* result which we will compare with to other variations of other picking evaluations.

Each point has 5 timestamps; start, picked, lookout, placed, finished. Pick is the time taken between start and lookout, place being the time between lookout and finished stamps. A point that has all the five stamps present is considered a *successful* point, a *failure* otherwise.

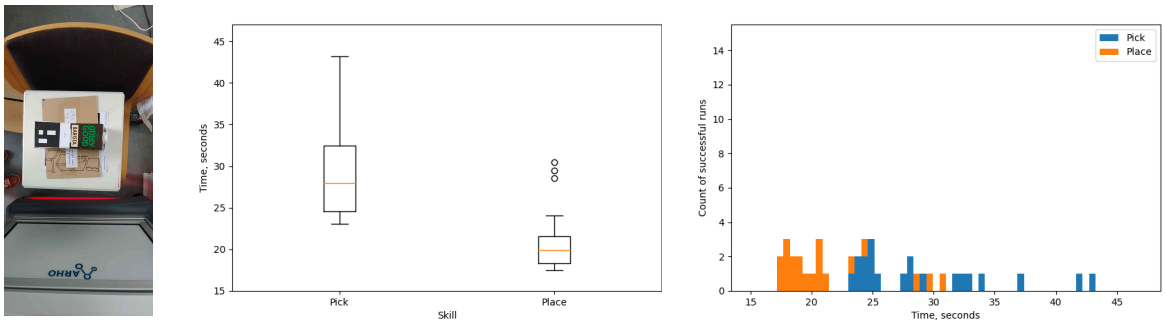
5.3.2 Tall package, lying down, 0 degree turn



(a) (b) (c)

Figure 5.9: Plots for 20 points measured when the package is lying down, with marker end forward to robot. (a) Position. (b) Box plot. (c) Stacked histogram.

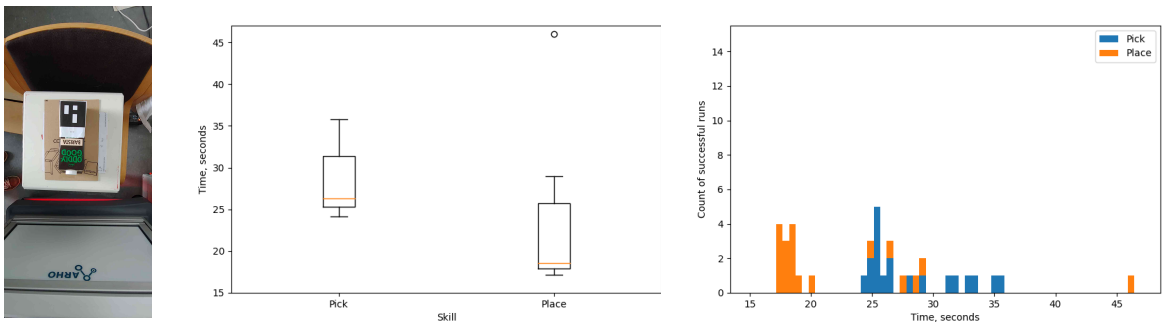
5.3.3 Tall package lying down, 90 degree turn



(a) (b) (c)

Figure 5.10: Plots for 20 points measured when the package is lying down at 90deg angle away from the robot. (a) Position. (b) Box plot. (c) Stacked histogram.

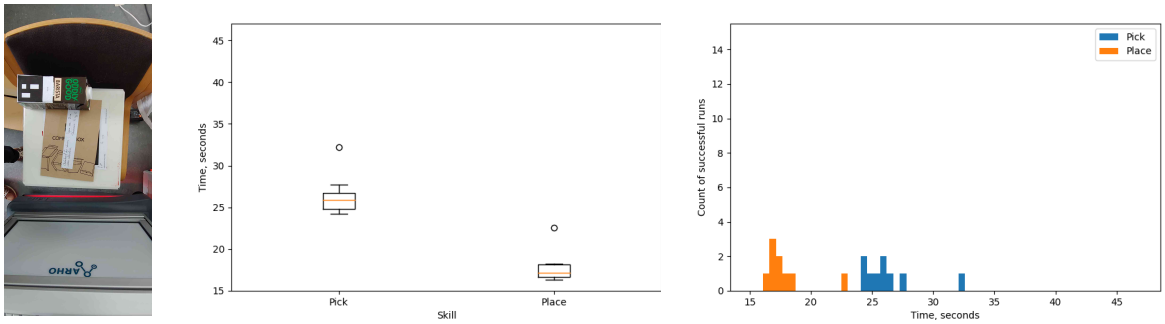
5.3.4 Tall package lying down, 180 degree turn



(a) (b) (c)

Figure 5.11: Plots for 20 points measured when the package is lying down at 180deg angle away from the robot. (a) Position. (b) Box plot. (c) Stacked histogram.

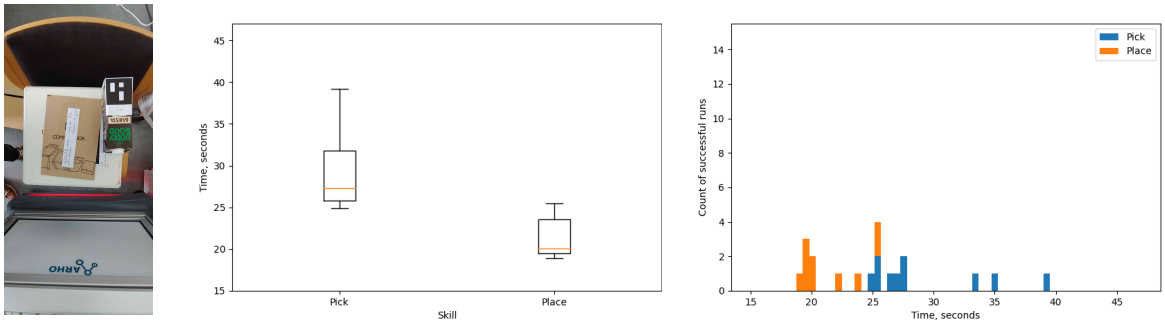
5.3.5 Tall package, lying down, top-left corner position



(a) (b) (c)

Figure 5.12: Plots for 10 points measured when the package was on the top-left position of the square. (a) Position. (b) Box plot. (c) Stacked histogram.

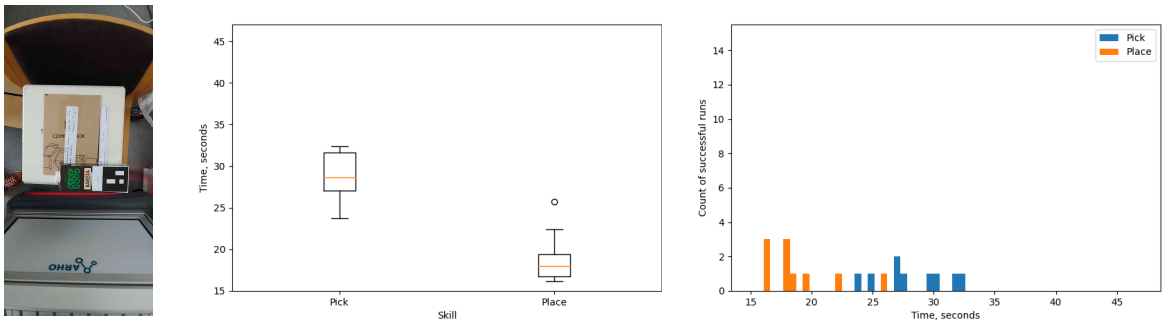
5.3.6 Tall package, lying down, top-right corner position



(a) (b) (c)

Figure 5.13: Plots for 10 points measured when the package was on the top-right position of the square. (a) Position. (b) Box plot. (c) Stacked histogram.

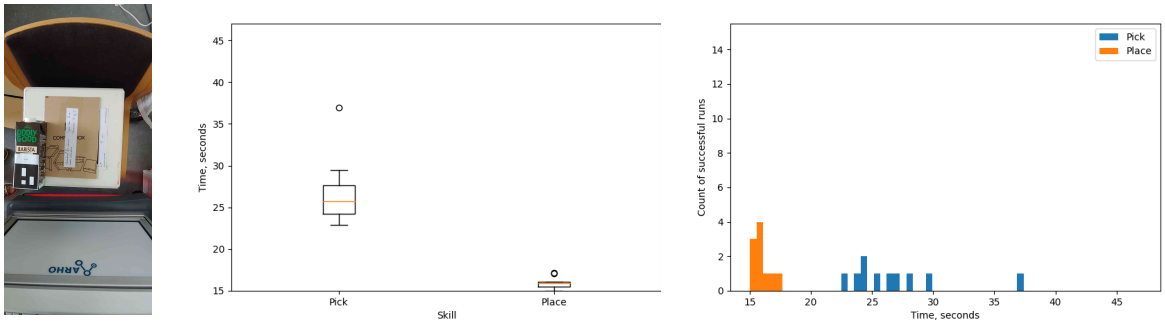
5.3.7 Tall package, lying down, bottom-right corner position



(a) (b) (c)

Figure 5.14: Plots for 10 points measured when the package was on the bottom-right position of the square. (a) Position. (b) Box plot. (c) Stacked histogram.

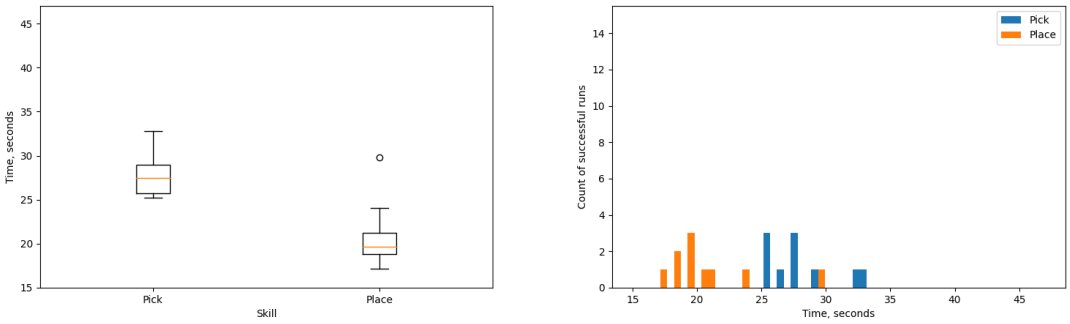
5.3.8 Tall package, lying down, bottom-left corner position



(a) (b) (c)

Figure 5.15: Plots for 10 points measured when the package was on the bottom-left position of the square. (a) Position. (b) Box plot. (c) Stacked histogram.

5.3.9 Juice package, standing



(a) Box plot **(b)** Stacked histogram

Figure 5.16: Plots for 20 points measured when the juice package was standing vertically.

5.3.10 ArUco-marker size & picking performance correlation

Description	Marker Size	# Points	# Successful (%)	μ	σ
Tall, lying	70mm	20	20 (100.0%)	29.29sec	5.90sec
Juice, lying	35mm	20	20 (100.0%)	27.35sec	3.17sec
Stout, lying	20mm	15	5 (33.3%)	26.89sec	5.09sec

Table 5.4: Overview of picking evaluation with focus on the correlation between the size of ArUco marker's size and picking performance. Distance between camera and object during the initial lookout pose is approximately 60cm in all tests.

Chapter 6
Discussion

During the evaluation, we found some major challenges that impacted the reliability and accuracy of our purposed system:

- The use of small-sized ArUco markers led to inaccurate pose estimations, a significant factor contributing to incorrect orientation detection.
- The actuation driver called MoveIt wasn't robust enough to fulfill the 'random look-at' step of an evaluation task. The robot becomes stuck in a loop and requires the evaluation task to be restarted.
- The Cartesian trajectory generator and controller, which serve as a rapid and compliant control mode alternative to MoveIt. This controller exhibited a tendency to become trapped in loops and displayed comparatively lower accuracy than the slower yet more precise joint-config controller. To address this issue, implementing safety measures to prevent the driver from entering a loop was necessary.
- Heron encountered a range of technical complications, which subsequently affected the robot's performance capabilities and resilience to latency, as well as other undesirable behaviors instigated by these problems. A recurring issue involved the robot's arm driver, which periodically lost connection to the hardware, leading to disruptions in actuation. This connection is essential for relaying commands to the arm, and a disconnection therefore renders the arm immobile until the connection is reestablished. We unfortunately did not possess the necessary technical expertise to diagnose and tackle the root causes of this problem. Furthermore, we observed that the arm's performance declined as the battery charge neared depletion. This was likely due to the robot's power consumption exceeding the recharge rate, thereby causing the battery to drain faster than it could be replenished, even when the robot was actively operating.

6.1 Analysis of Hand-Eye Calibration Outcomes

As demonstrated in Figures (5.1, 5.3, 5.5), hand-eye calibration is a crucial step for ensuring accurate and reliable pose estimation in precision-oriented vision-based tasks.

While generating different observation poses for the target, we discovered that poses 180 degrees apart did not contribute valuable data to the final computation. To address this, we discarded observation poses similar to previous ones and restricted the range of angles used by the skill.

Moreover, for optimal hand-eye calibration, a reasonably sized ArUco marker was necessary to ensure accurate pose estimation. This, in turn, contributed to the overall quality and effectiveness of the calibration process.

The resulting skill demonstrated high effectiveness and reliability, provided the previously mentioned controller issues and other difficulties with the robot were absent. It's worth noting that during the evaluation, we didn't account for the number of unique poses required or the total execution time of the skill. Incorporating these metrics could have added depth to our overall analysis of the skill's performance.

6.2 Evaluation of ArUco Marker Pose Estimation Results

The utilization of ArUco markers for pose estimation proves to be an effective method, yielding satisfactory results while presenting certain advantages and disadvantages.

The primary challenges of using ArUco markers include:

- **Marker Size Dependency**: The size of the ArUco marker plays a critical role in its detectability, particularly for vision-based tasks. The choice of marker size influences the distance at which the marker can be effectively detected and consequently, be useful for pose estimation. This is especially crucial when the task requires early detection or estimating the pose from a significant distance, such as in pick-and-place operations.
- **Attachment Requirements**: Utilizing ArUco markers necessitates their attachment to the object of interest, which may not always be feasible or convenient depending on the specific application or environment.

However, the advantages of employing ArUco markers are compelling:

- **Ease of Use**: ArUco markers are user-friendly, supported by well-tested and reliable libraries that simplify their implementation.
- **High Precision**: When applied appropriately, such as ensuring sufficient marker size, ArUco markers can yield highly accurate pose estimation results.
- **Cost-Effectiveness**: ArUco markers present a cost-effective solution for pose estimation, given their ease of production and implementation.

- **Adaptability**: The potential to use markers of varying sizes caters to a wide range of requirements, making ArUco markers an adaptable solution for diverse scenarios.
- **Real-time Capability**: The simple structure of ArUco markers facilitates rapid detection, making them suitable for real-time applications where swift execution is paramount.

In summary, despite some limitations, the benefits of ArUco markers make them a worthwhile tool for pose estimation tasks.

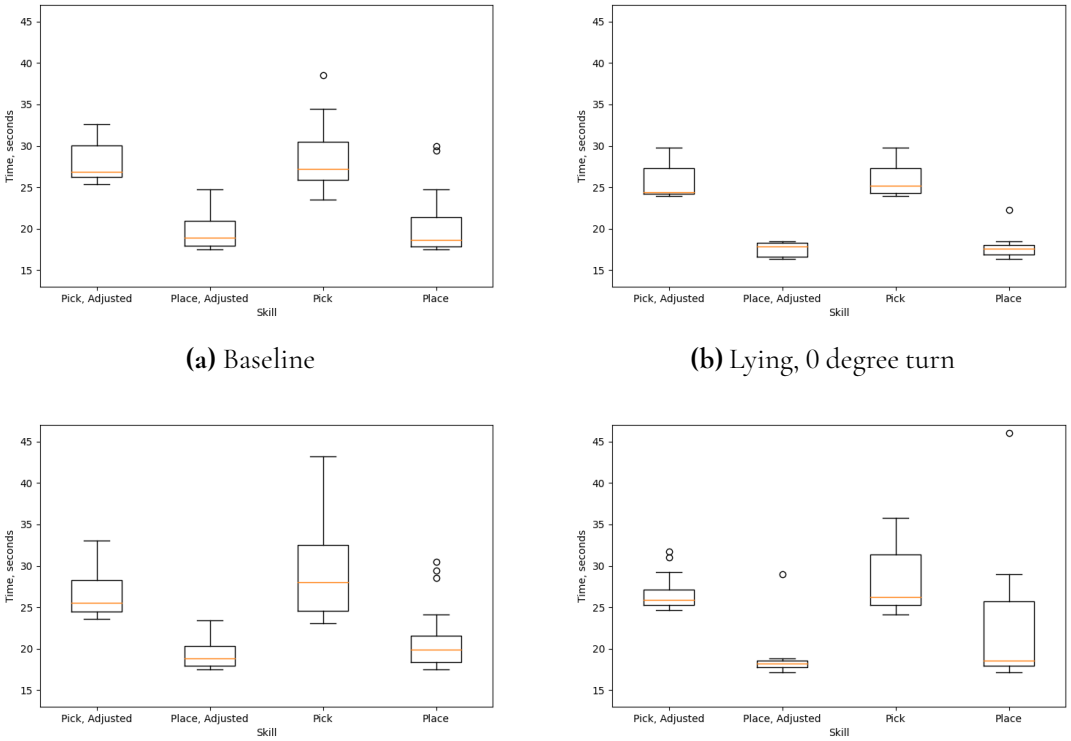
In reference to Table 5.3 and Figure 5.6a, it is evident that the pose estimation results using a 20mm marker were not optimal when applied to the larger container. However, a better outcome was achieved with the same-sized marker on a smaller container. These containers significantly differ in size and color as depicted in Figure 3.2, with the larger being a 1000ml brown container and the smaller being a 250ml white container. The underlying reasons for this performance discrepancy remain speculative and could potentially be attributed to differences in color and reflective properties of the containers.

On the other hand, the use of a 70mm marker, as demonstrated in Figure 5.6c, validates the effectiveness of ArUco markers in pose estimation tasks. However, the application of a 35mm marker presented in Figure 5.6d shows a higher dispersion, particularly in rotation. While the translations along the x, y, z axes and the standard deviation values for this marker size, as detailed in Table 5.3, seem promising, the rotation estimation was less satisfactory.

6.3 Analysis of Picking Evaluation Results

All the Figures of the evaluation results (from Figure 5.8 to 5.11) show that the duration of the picking has a fairly consistent mean of $\approx$ 25 seconds, while the times for placing has a mean of $\approx$ 20 seconds. But they also show a large amount of variation and outliers, those are noted to be product of the robot's technical difficulties.

As evident from Table 5.4, the size of the ArUco marker does not significantly impact the time taken for the pick and place operation. However, it does play a pivotal role in the successful execution of these operations. As discussed in Section 5.2, the use of smaller markers initially led to poor pose estimation. This issue was effectively addressed by introducing a *detector pose* for the object.



(a) Baseline

(b) Lying, 0 degree turn

(c) Lying, 90 degree turn

(d) Lying, 180 degree turn

Figure 6.1: Box-plots for all picking evaluation tests, the data adjusted to remove outliers based on $\mu \pm \sigma$, to approximate the same result if the technical difficulties were not present. Compared to the actual results on the right side of each plot.

In Figure 6.1, we present the results adjusted using only data from instances where the robot performed as expected. This adjustment involved considering only attempts that fell within the range of $x \in [\mu - \sigma, \mu + \sigma], x \forall \mathbf{X}$, where μ and σ represent the mean and standard deviation, respectively. The revised results thus approximate the outcomes we might expect if evaluations were conducted without any technical difficulties and that all points were successful.

A comparative analysis of figures in 6.1 demonstrates that a significant portion of the observed variation can be attributed to the outliers present in the actual data. These outliers correspond to instances of technical difficulties resulting in suboptimal performance.

The robot's performance was most consistent in the scenario depicted in Figure 6.1b. However, the scenarios presented in Figures 6.1c and 6.1d exhibit extended whiskers, indicative of the challenges encountered. These situations necessitated the rotation of the robotic arm's wrist and a secondary pose estimate of the package when the hand was approaching the target, potentially contributing to the increased variability.

In the baseline vertical picking scenario (Figure 6.1a), the average performance was comparable to that of Figure 6.1b. This scenario involved the package standing upright, thereby necessitating more rotations and consequently increasing the complexity of the task. Never-

theless, the three scenarios involving rotations did not exhibit a linear relationship between time variation and complexity.

A second potential contributing factor to the longer whiskers observed in the box plots of Figure 6.1, seen in all cases except for the baseline, could be attributed to variations in the number of sampled data points. While Figure 6.1a is based on 50 data points, only 20 data points were sampled for Figures 6.1c and 6.1d. While this discrepancy might potentially introduce a sample size bias, the choice was made due to the assumption that lying down at different angles essentially share the same rotation point - the package's geometrical centre. Hence, by collecting all data points for the 'lying down' position, we would have a total of 60 data points.

Despite facing various challenges, our data indicates a high success rate for the implemented picking skill across all aspects, including object orientation and position. Considering the limitations and challenges encountered during the evaluation phase, we are confident that the picking skill, facilitated by the use of the ArUco marker, has demonstrated satisfactory performance. This success suggests that our approach is robust and resilient, even under demanding circumstances, reaffirming the viability of our design and its potential for further development and refinement.

6.4 Future Work

In the course of our project's evaluation and further work, we identified several areas for potential enhancement and proposed modifications to our design or implementation. Due to the project's scope and time constraints, we were unable to pursue these changes. However, our recommendation on future work are as follow:

- Develop a skill to determine the most suitable side of the package for the robot to approach, particularly when two sides are visible.
- Consider storing the arbitrary offset value, which enhances the gripper's pinching force for a more secure grasp, in the package world model. This should be individually tailored to each specific type.
- Explore the use of different types of grippers. While the WSG-50 Gripper is sufficient, it often exerts too much force on the containers, causing deformation. An alternative, such as a vacuum gripper, might provide a more delicate grasp.
- Enhance operational speed by conducting detection and pose estimation processes in parallel. This could significantly reduce the overall time required for each pick and place operation.
- Investigate the feasibility of applying the developed skills to moving objects on a conveyor belt, and identify the necessary adaptations or enhancements required to facilitate this.

These suggestions could potentially improve the performance of the robot and provide a more effective and efficient solution for object detection and manipulation.

6.5 Controller-Arm Connection Issue

A crucial aspect requiring attention and improvement is the controller's robustness, a quality integral to the functionality of a robot. We discussed earlier that the controller ceases to operate effectively when the arm driver in the computer loses its connection to the arm, rendering the controller unable to transmit commands. Upon reestablishing connection with the arm driver, the controller does not attempt to reestablish its own connection, operating under the presumption of a sustained connection. This leads to a discrepancy in the perceived and actual position of the arm: the controller may perceive the arm as having reached its intended position, when in reality, it remains at the location where the connection was lost.

Addressing this reconnection issue led to the discovery of a side effect - an offset in the arm's position that accumulates each time the arm disconnects from the computer. Although seemingly unaffected, the controller's performance deteriorates over time due to this offset. It increasingly struggles to guide the arm to its target position, as the offset precludes the arm from falling within the error margins set by the user and the controller.

We have thus far identified two temporary workarounds:

- **Complete restart of the ROS software** on the computer, although this approach does result in some downtime as we wait for the software to ready itself for use.
- **Complete restart of the robot unit**, a more time-consuming process that necessitates a substantial waiting period.

The root cause of this issue - whether it lies within the controller or the arm - remains uncertain due to our limited knowledge in this area. We therefore recommend a thorough technical investigation into the arm to gain a more comprehensive understanding of the problem.

Chapter 7
Conclusion

According to the results detailed in Section 5.1, the implementation of hand-eye calibration significantly improved the accuracy of our pose estimation compared to manual measurements. Without this calibration, achieving consistent and repeated object pickups would have been unachievable.

Our final implementation demonstrates the possibility of creating a pick skill that is not only efficient and reliable, but also versatile, capable of handling objects of various sizes and orientations. While the results were affected by certain systemic issues during evaluation, the skill successfully utilized a world model to accumulate knowledge about the objects it could pick and the methods to do so.

ArUco markers fulfilled the pose estimation requirements necessary to execute object pickups. However, they introduced additional constraints such as requiring sufficiently large markers and the need for attaching them to the objects. Thus, future work could explore alternative or complementary methods to ArUco markers to mitigate these constraints while maintaining pose estimation performance.

References

[1] J.M Baker. Maths - transformations using quaternions. http://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/transforms/index.htm, Accessed 07 Jun 2023.

[2] Kai Chen, Rui Cao, Stephen James, Yichuan Li, Yun-Hui Liu, Pieter Abbeel, and Qi Dou. Sim-to-real 6d object pose estimation via iterative self-training for robotic bin picking. https://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsarx&AN=edsarx.2204.07049&site=eds-live&scope=site, Accessed 09 June 2023.

[3] Michele Colledanchise and Petter Ögren. Behavior trees in robotics and ai: An introduction. https://doi.org/10.1201/9780429489105, 2017.

[4] Thomas H. Davenport and Steven M. Miller. Fast food hamburger outlets: Flippy - robotic assistants for fast food preparation. In *Working with AI: Real Stories of Human-Machine Collaboration*, pages 147–150, 2022.

[5] Alexander Fabisch. A comparison of policy search in joint space and cartesian space for refinement of skills. http://arxiv.org/abs/1904.06765, Accessed 23 May 2023.

[6] FANUC. 3d area sensor. https://www.fanuc.eu/ch/en/robots/options/3d-area-sensor, Accessed 18 March 2023.

[7] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014.

[8] Bjarne Großmann and Volker Krüger. Continuous hand-eye calibration using 3d points. https://doi.org/10.48550/arXiv.2004.12611, 2020.

[9] A. Grunnet-Jepsen, J. Sweetser, T. Khuong, S. Dorodnicov, D. Tong, O. Mulla, H. Eliyahu, and E. Raikhel. Intel® realsense™ self-calibration for

d400 series depth cameras. https://dev.intelrealsense.com/docs/self-calibration-for-depth-cameras, Accessed 06 Jun 2023.

[10] Dan Halperin, Oren Salzman, and Micha Sharir. Algorithmic motion planning. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, page 1311–1342. CRC Press LLC, third edition, 2017.

[11] Le Duc Hanh and Khuong Thanh Hieu. 3d matching by combining cad model and computer vision for autonomous bin picking. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 15(2-3):239–247, 2021.

[12] Tushar Jadhav, Manisha Jadhav, Abhijit Chitre, and Ajit Patil. Development of vision based test jig and mechanism for automatic inspection and sorting of industrial objects. In *2022 6th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pages 1–5, Aug 2022.

[13] Sree S.S. Katta, J. Adnan, S. Chaudhary, S. Dutta Roy, Chetan Arora, S. K. Saha, and Magid E. Pose estimation of 5-dof manipulator using on-body markers. In *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, pages 897–902, 2021.

[14] Jeff Knisley. Coordinate transformation. https://math.etsu.edu/multicalc/prealpha/chap3/chap3-1/printversion.pdf, Accessed 09 June 2023.

[15] Volker Krüger. Calibration: For intelligent autonomous systems. EDAP20 Lecture slides, 2020.

[16] Volker Krüger. Actuation 2: Manipulation and robot skills. EDAP20 Lecture slides, 2022.

[17] Jack B. Kuipers. *Quaternions and rotation sequences : a primer with applications to orbits, aerospace, and virtual reality*. Princeton University Press, 1999.

[18] Phil Lavelle and Sandra Stojanovic. Robots are making french fries faster, better than humans. https://www.reuters.com/technology/want-fries-with-that-robot-makes-french-fries-faster-better-than-humans-do-2022-1 Accessed 02 February 2023.

[19] Eva Maia, Sinan Wannous, Tiago Dias, Isabel Praça, and Ana Faria. Holistic security and safety for factories of the future. *Sensors (14248220)*, 22(24):9915, 2022.

[20] The MathWorks. Urdf primer. https://www.mathworks.com/help/sm/ug/urdf-model-import.html#responsive_offcanvas, Accessed 22 May 2023.

[21] Wojciech Matusik. Coordinates and transformations. https://ocw.mit.edu/courses/6-837-computer-graphics-fall-2012/5cbb1bf32a92fad91e8ad6c37a473240_MIT6_837F12_Lec03.pdf, Accessed 09 June 2023.

[22] Matthias Mayr and Julian M. Salt-Ducaju. A c++ implementation of a cartesian impedance controller for robotic manipulators. https://arxiv.org/abs/2212.11215, Accessed 09 June 2023.

[23] Nobuyuki Nakatani, Yuhki Shiraishi, and Fumiaki Takeda. Development of oval-shaped agricultural product inspection and sorting system with simultaneous six-angle photograph taking. In *2011 Fourth International Conference on Modeling, Simulation and Applied Optimization*, pages 1–6, April 2011.

[24] Alberto Olivares-Alarcos, Daniel Beßler, Alaa Khamis, Paulo Goncalves, Maki Habib, Julita Bermejo, Marcos Barreto, Mohammed Diab, Jan Rosell, João Quintas, Joanna Olszewska, Hirenkumar Nakawala, Edison Pignaton, Amelie Gyrard, Stefano Borgo, Guillem Alenyà, Michael Beetz, and Howard Li. A review and comparison of ontology-based approaches to robot autonomy. *The Knowledge Engineering Review*, 34:e29, 2019.

[25] OpenCV. Camera calibration. `https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html`, Accessed 02 February 2023.

[26] OpenCV. Detection of aruco markers. `"https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html"`, Accessed 02 February 2023.

[27] Christian Ott. *Cartesian Impedance Control: The Rigid Body Case*, pages 29–44. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[28] F.C. Park and B.J. Martin. Robot sensor calibration: solving ax=xb on the euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721, 1994.

[29] Pick-it. Pick-ittm robotic picking. `https://www.pickit3d.com/`, Accessed 18 March 2023.

[30] Yang Qian, Rong Jiacheng, Wang Pengbo, Yang Zhan, and Geng Changxing. Real-time detection and localization using ssd method for oyster mushroom picking robot. In *2020 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 158–163, Sep. 2020.

[31] PickNik Robotics. Moveit motion planning framework. `https://moveit.ros.org/`, Accessed 25 May 2023.

[32] Francesco Rovida, Matthew Crosby, Dirk Holz, Athanasios S. Polydoros, Bjarne Großmann, Ronald P. A. Petrick, and Volker Krüger. *SkiROS—A Skill-Based Robot Control Platform on Top of ROS*, pages 121–160. Springer International Publishing, Cham, 2017.

[33] Aashay Sathe and Anjali Deshpande. Automated visual quality inspection and sorting. In *2017 International Conference on Advances in Computing, Communication and Control (ICAC3)*, pages 1–6, Dec 2017.

[34] RVMI Lab Aalborg university. Skill-based robot control platform for ros v2.0 (skiros2). `https://github.com/RVMI/skiros2`, Accessed 8 May 2023.

[35] Technical university of munich. Hand-eye calibration. `https://campar.in.tum.de/Chair/HandEyeCalibration`, Accessed 7 May 2023.

[36] Open Source Computer Vision. Aruco marker detection (aruco module). `https://docs.opencv.org/4.x/d9/d6d/tutorial_table_of_content_aruco.html`, Accessed 22 May 2023.

[37] I Yung, Yamuna Maccarana, Gabriele Maroni, and Fabio Previdi. Partially structured robotic picking for automation of tomato transplantation. In *2019 IEEE International Conference on Mechatronics (ICM)*, volume 1, pages 640–645, March 2019.

EXAMENSARBETE Intelligent Robotic Systems for Quality Control
A Study on a Pick Skill for a Collaborative Robot
STUDENTER Marcus Nagy, Martin Lyrå
HANDLEDARE Volker Krueger (LTH)
EXAMINATOR Jacek Malec (LTH)

Flexibel plockarrobot: Plocka vad från var som helst

POPULÄRVETENSKAPLIG SAMMANFATTNING **Marcus Nagy, Martin Lyrå**

Kvalitetskontroll är ett uppdrag som utförs för hand av personer, där sker ett vilje för att kunna automatisera denna process. Detta arbete utforskar på möjligheten att använda så kallade *collaborative robots* till hitta paket samt plocka upp dem som en del av automatiseringen.

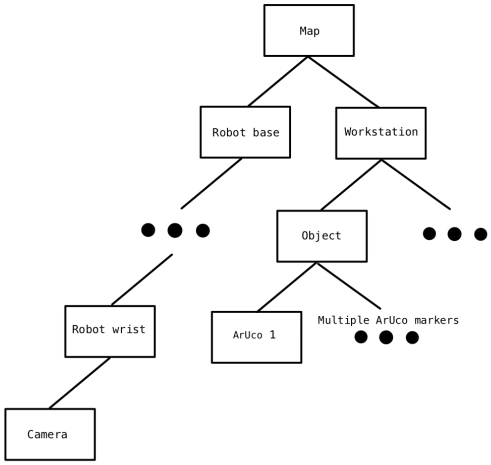
Tetra Pak är en av världens ledande producenter av förpackningslösningar för flytande vätskor avsedda för konsumenter. Det pappersmaterial som används för förpackningarna levereras på stora rullar som sedan trimmas, viks och limmas ihop i en systematisk process.

Förpackningarna som kommer ut ur maskineriet förväntas uppfylla vissa standarder, både gällande utseende och mindre synliga aspekter, som att innehållet är korrekt.

Industrins utveckling rör sig mot mer automatisering, även känt som "Industry 4.0", vilket har lett till en explosionsartad utveckling inom området. För att kunna bedöma förpackningarnas kvalitet och automatisera denna process, finns det vissa delmål som måste uppnås. Bland dessa är förmågan att identifiera och plocka upp förpackningarna avgörande för att i slutändan kunna göra någon form av kvalitetsbedömning.

Identifieringen av förpackningarna bygger på ArUco-markörer, som är välkända för att fungera bra för lokalisering. Men för att kunna få tillförlitliga uppskattningar krävs det bland annat att både kameran och avståndet mellan kamera och gripdon är kalibrerade.

Vi kunde konstatera att ArUco-markörerna fungerar tillräckligt bra för att användas som



lokalisering för denna typ av uppgift, men de har vissa negativa aspekter som kanske inte är önskvärda. Resultaten visade också att utan korrekt kalibrering är det omöjligt att utföra någon typ av uppgift som kräver precision. Vi har kunnat visa att det är möjligt att implementera en funktion som tillåter en robot att upprepade gånger plocka upp förpackningar av olika storlekar och i olika orienteringar genom att modellera den kunskap som behövs av roboten för att realisera detta.