Autonomous Dynamic Grasping with a Robotic Arm: Real-Time Motion Prediction and Adaptive Control

BY

SIMONE UGHETTO
B.S., Politecnico di Torino, Turin, Italy, 2023

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Chicago, 2025

Chicago, Illinois

ACKNOWLEDGMENTS

I want to express my deepest gratitude to all those who supported me throughout my academic journey and the completion of my Master's thesis.

Firstly and most importantly, a debt of gratitude is owed to my advisor, Professor Pranav A. Bhounsule, for entrusting me with this project. His invaluable support, patience, and guidance provided me with an exceptional opportunity to delve into the field of robotics and manipulators.

My sincere thanks are also extended to the members of my defense committee, Professor Marcello Chiaberge and Professor Piergiorgio L. E. Uslenghi for their valuable and constructive advice and feedback.

I would also like to thank all my lab colleagues for fostering a constructive and collaborative work environment.

Last but not least, I extend my heartfelt gratitude to my family and girlfriend, who have been unconditionally encouraging and supportive at any time, allowing me to pursue my dreams.

SU

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

FIGURE **PAGE**

LIST OF FIGURES (continued)

FIGURE **PAGE**

LIST OF ABBREVIATIONS

DOF	Degrees of freedom
UGV	Unmanned ground vehicle
ROS	Robot operating system
RGB	Red, green, and blue
RGB-D	Red, green, blue, and depth
LSTM	Long short-term memory
PID	Proportional-integral-derivative
3D	Three-dimensional
6D	Six-dimensional
MPC	Model predictive control
DH	Denavit-Hartenberg
URDF	Unified robot description format
API	Application programming interface
MoCap	Motion capture
QoS	Quality of state
SDK	Software development kit
EE	End-effector

SUMMARY

Dynamic grasping of moving objects represents one of the most challenging and actively researched areas in robotic manipulation. Unlike static grasping, this task requires the integration of perception, prediction, and control in real-time. Typical approaches involve tracking the target's position in real-time using vision-based systems, estimating its velocity, and generating tailored Cartesian-space trajectories to ensure the robot's end-effector coincides with the object's position at the precise time required for successful grasping.

The presented work addresses the development, optimization, and hardware implementation of a novel control algorithm for dynamic grasping. The research utilizes the ROS 2 framework alongside the WidowX 250 S robotic arm, which features six degrees of freedom. This configuration allows for autonomous grasping of objects in motion while dynamically adapting to alterations in their trajectory, which are not necessarily restricted in space.

To achieve this objective, a motion capture system continuously monitors the target's position relative to the manipulator, streaming real-time data to the robot's controller, which estimates the relative velocity. Based on these estimates and predefined constraints, the controller intelligently determines the optimal timing to initiate the grasp sequence, predicting the target's trajectory and evaluating its transit in the dynamically computed reachable workspace of the robotic arm, determined at runtime using an efficient analytic inverse kinematics function tailored to the robot. Additionally, an advanced adaptive trajectory planner generates detailed trajectories, including position, orientation, velocity, and acceleration for the robot's end-effector, in under a millisecond. These trajectories are then relayed to the motor controllers, ensuring precise and coordinated motion.

Finally, experimental validation confirms that the developed adaptive dynamic grasping algorithm performs as expected, demonstrating high repeatability and accuracy, effectively showcasing its robustness in real-world scenarios.

CHAPTER 1

INTRODUCTION

1.1 Dynamic grasping

Robotic technology has revolutionized industrial automation since its inception in the 1960s, when robotic arms were first introduced to streamline repetitive manufacturing processes. Initially adopted primarily in automotive manufacturing, robotic manipulators rapidly expanded their presence across various industries due to their precision, repeatability, and ability to perform tasks under challenging environmental conditions. Today, robotic arms are indispensable tools in numerous sectors, including manufacturing, logistics, agriculture, and healthcare, significantly enhancing productivity and consistency and, at the same time, handling tasks that could potentially be harmful or wearing for human beings.

One fundamental task robotic arms perform across industries is grasping, which is essential for pick-and-place operations, assembly tasks, and object sorting. Traditionally, grasping operations have been conducted in static environments where objects are stationary and their positions known beforehand. However, increasing demands for efficiency and adaptability in dynamic environments, such as warehouses with moving conveyors or agricultural fields with harvesting robots, have propelled research toward dynamic grasping, a challenging but highly rewarding domain.

Dynamic grasping involves accurately identifying, tracking, and capturing objects in motion. Unlike static grasping, it requires integrating advanced perception techniques, real-time motion prediction, and adaptive control algorithms. Successfully implementing dynamic grasping has the great potential to significantly reduce the cycle time of pick-and-place operations, enhancing overall operational efficiency and flexibility. In addition, robotic arms capable of grasping moving targets can be effectively integrated into complex environments, including those involving mobile robots. This integration combines locomotion and manipulation, thereby facilitating the execution of tasks while consistently adapting to the dynamic characteristics of the surroundings. As a result, these robotic systems can operate without interrupting their functions or requiring other robots to alter their original tasks.

1.2 Motivations

The ability to grasp objects at a nonzero relative speed with respect to a robot arm offers significant advantages over a static framework. First of all, it enables more time-efficient operations, especially when performed in a sequence with other tasks. At the same time, it could preserve task continuity in a multi-robot environment, where the action of a subsystem does not need to pause the motion of other devices and interrupt other processes.

However, despite significant advancements, dynamic grasping remains a challenging and actively evolving area within robotics. The complexities inherent in real-time perception and control, uncertainties in object motion, and the necessity for swift and precise robotic responses contribute to the ongoing research in this field. Addressing these challenges necessitates the integration of robotic manipulators, advanced vision systems, and mobile robotic platforms. Such integration holds substantial promise across various sectors, including manufacturing, warehousing, agriculture, and construction.

In automated warehouse environments, efficiency in handling products is paramount. The deployment of mobile robots equipped with manipulators and integrated vision systems can facilitate the transfer of items between moving platforms without necessitating halts in their operation. This capability enables seamless exchanges of products and packages during transit, thereby enhancing throughput and reducing downtime. Furthermore, quality control processes can be optimized; for instance, defective items identified by vision systems during transportation can be selectively removed without rerouting entire consignments, thus preserving the efficiency of the supply chain.

In the agricultural sector, adopting UGVs equipped with robotic arms presents opportunities to revolutionize harvesting processes. These mobile manipulators can navigate through orchards and fields, harvesting delicate fruits from trees and bushes without necessitating stops. Continuous motion harvesting not only increases productivity but also minimizes the physical stress on crops, preserving their quality.

Similarly, space operations can benefit significantly from advancements in dynamic grasping. Robotic manipulators mounted on servicing spacecraft can autonomously capture satellites or space debris that are spinning or tumbling unpredictably. Such autonomous grasping in microgravity conditions demands precision, real-time motion estimation, and delicate handling to ensure mission success and to avoid secondary motion disturbances. This capability is crucial for satellite servicing, repair missions, and active debris removal, enhancing safety and sustainability in space operations.

The convergence of mobility, perception, and manipulation in robotic systems is pivotal for advancing automation across these sectors. By enabling robots to interact dynamically with their environments, we can address current limitations and unlock new potential in efficiency and functionality.

1.3 Document structure

This thesis begins with Chapter 1, providing an introduction to dynamic grasping, highlighting its significance, and establishing the motivations behind this research. It also briefly outlines the document structure for clear navigation throughout the dissertation.

Chapter 2 presents related work, exploring foundational and contemporary research in robotic grasping. It focuses specifically on static grasping, real-time dynamic grasping involving perception, prediction, and control, and introduces loco-manipulation, which combines locomotion and arm-based manipulation.

Chapter 3 details the methods and software architecture central to the dissertation. This chapter initially covers the kinematic modeling of the robotic arm, describing both forward and inverse kinematics and incorporating differential kinematics. It then delves into grasp decision logic, state machine implementation, and trajectory planning, providing a comprehensive overview of the theoretical and practical approaches used.

Chapter 4 describes the simulation framework developed to validate the proposed methodologies and algorithms.

Chapter 5 presents the experimental results from hardware implementations, showcasing the practical application and performance of the developed techniques.

The results from simulations and experiments are discussed and analyzed in Chapter 6, ensuring a clear understanding of the achievements and limitations of this research.

Finally, Chapter 7 concludes the thesis, summarizing key findings and providing suggestions for future research directions.

1.4 Additional notes

Let us now define a brief list of remarks that are relevant for a clearer understanding of the following chapters:

- Vectors and matrices are consistently represented in boldface type to ensure clarity and consistency in mathematical notation across all textual content, tabular data, and graphical representations.
- When expanded to show their elements, vectors will be denoted by round brackets and matrices with square brackets.

CHAPTER 2

RELATED WORK

2.1 Static grasping in robotic manipulation

Initial efforts in robotic grasping targeted static objects within well-defined environments. These systems generally assumed perfect knowledge of the object's geometry and location, and mainly focused on obtaining a good quality and efficient grasp of objects. Planning algorithms such as those in GraspIt! [5] used 3D object models and sampled candidate grasp poses, optimized using metrics like grasp quality or wrench space analysis [6].

Analytic solutions to inverse kinematics and pre-computed trajectories were typically sufficient in static scenes. However, these approaches are insufficient under uncertainties or variations in the targets' pose. Moreover, their reliance on complete prior models and rigid scene structure limited their practical deployment in dynamic environments.

2.2 Dynamic grasping: real-time perception, prediction, and control

Compared to the extensive research on grasping stationary objects and relative sub-problems, fewer studies have tackled the more complex challenges of grasping moving objects, which therefore have not been explored to the same extent. One of the first contributions in this area is the research conducted by Houshangi [7], who developed an adaptive control framework that allows a robotic manipulator to grasp a moving target using vision-based feedback. Recognizing that vision systems inherently introduce latency due to image processing, the study proposed a real-time prediction strategy based on an autoregressive model. This model leveraged past positional data to predict the future trajectory of the target, enabling the manipulator's end-effector to anticipate the target's movement and effectively align with its expected future position. The controller was demonstrated on a Puma 600 robot using a fixed monocular vision system.

Around the same period, Allen et al. [8] proposed a complementary approach focused on robotic hand-eye systems, which means the camera and manipulator were rigidly connected. Their method emphasized high-speed visual tracking of moving objects and real-time coordination between cameras and end-effector. In particular, by embedding the vision system directly into the manipulator's structure, the authors significantly reduced latencies and calibration complexities.

Moving to very recent developments, a noteworthy contribution is the real-time dynamic grasping system proposed by XiaoYang et al. [9], which integrates an eye-in-hand RGB-D camera (Intel RealSense D435i) mounted on a UR5 industrial manipulator. Developed in a ROS framework, their system performs real-time tracking and grasping of objects moving along a conveyor belt with translational motion only. A fixation approximation-based visual feedback control strategy is implemented, which computes end-effector offset via depth sensing and continuously refines the robot's trajectory using PID control. Motion planning is handled by an enhanced RRT-Connect algorithm, allowing smooth, real-time trajectories to be generated. The system demonstrates good reliability and accuracy, with successful grasps in a controlled lighting environment and target speeds lower than about $0.15m/s$.

A parallel learning-based approach was introduced by Zhang et al. [10], who proposed a dynamic behavior cloning framework enhanced with temporal feature prediction. Instead of relying uniquely on visual feedback or classical planning, their system learns to determine the future grasping point and time from historical trajectory data using LSTM-based sequence modeling. The key innovation over standard techniques is that this method enables the robot to act proactively rather than reactively, improving grasp success in scenarios with predictable motion patterns. Their method outperformed standard model-based planners in dynamic object pickup tasks where trajectory regularity was present.

A related approach was presented by Nguyen et al. [11], who developed a zero-shot grasping system that combines 6D pose estimation with model-predictive control. Their method uses FoundationPose, a vision-based pose estimation framework, to estimate the object's pose and MP-TrajOpt (Model Predictive Trajectory Optimization) to optimize the end-effector trajectory. The system updates pose estimates at 30 Hz and replans trajectories at 10 Hz, allowing it to adjust its motion in response to changes in the target's position. In addition, the framework supports task specification through natural language, which is translated into grasp goals using a vision-language model. The proposed system operates without requiring retraining for specific tasks, and the experiments show that it is able to grasp moving objects accurately by continuously integrating visual feedback into the planning process.

2.3 Loco-manipulation: combining locomotion and arm-based manipulation

While most traditional grasping approaches assume a fixed-base manipulator, many real-world applications require robots to move through their environment while interacting with objects. This is where loco-manipulation comes into play. Integrating dynamic grasping into these tasks can significantly boost efficiency and flexibility, especially in cooperative or not entirely predictable settings where objects are not stationary.

Recent research has made notable progress in enabling quadrupedal robots with arms to perform complex loco-manipulation behaviors. Sleiman et al. [12] introduced a planning and control framework based on bilevel trajectory optimization and multicontact reasoning. Their system was designed for the ANYmal robot and can handle multi-stage tasks like opening doors or pushing objects, where the robot must coordinate its limbs and base to maintain balance while interacting with the environment. Instead of relying on predefined motion sequences, the planner incrementally builds a tree of possible contact and motion combinations, using a combination of MPC and whole-body tracking to follow the best path. This allows the robot to switch between different contact modes, like using its legs or arms for pushing, depending on the situation.

Another example is RoLoMa, proposed by Weerasekera et al. [13], which focuses on making loco-manipulation more robust in real-world scenarios. The framework uses a hierarchical planner that combines kinodynamic planning, motion planning that considers both kinematic and dynamic constraints, and compliant whole-body control, allowing the robot to adapt to unexpected events like terrain changes or object slippage. One of the key features of RoLoMa is its ability to adjust grasping strategies on the fly, making it suitable for field tasks such as clearing debris or turning valves. It prioritizes flexibility over rigid planning, which makes it especially useful in dynamic or partially known environments.

Besides pure legged platforms, wheeled mobile manipulators are also part of the loco-manipulation landscape. Jiang et al. [14] presented a learning-based whole-body controller for a hybrid robot with both wheels and legs. Their system is trained to follow 6D end-effector pose trajectories by coordinating the motion of the arm and the base together. This is particularly useful when the robot operates in tight or cluttered spaces, as it can adjust its base position and arm configuration in tandem to maintain reachability and avoid obstacles. The use of reinforcement learning allows the robot to handle a range of tasks without relying on hand-crafted behaviors.

Altogether, these approaches demonstrate that dynamic grasping, combined with locomotion, can make robotic systems much more capable of performing manipulation tasks in complex, changing environments. As these platforms continue to evolve, their ability to collaborate, adapt, and execute tasks smoothly while in motion will be key to deploying robots in real-world settings

CHAPTER 3

MOTION CONTROL ARCHITECTURE

3.1 Kinematic modeling

3.1.1 Robot arm description

Before deriving the forward and inverse kinematics, we first introduce the physical structure of the robotic arm under study. The robot model is WidowX 250 S, manufactured by Trossen Robotics. For brevity, throughout this dissertation, we will often refer to it simply as WidowX. As shown in Figure 1, the system consists of a 6-degree-of-freedom serial-link manipulator driven by nine DYNAMIXEL servomotors: seven of the XM430-W350-T units and two of the XL430-W250-T units. Together, these servomotors actuate the arm's six joints [15][16]. Each of the six arm joints is a single-axis revolute joint, providing one degree of rotational freedom about its axis.

To enhance clarity, all joints are explicitly named following the naming convenition adopted by Trossen Robotics, as shown in Table I, facilitating consistent referencing throughout this thesis.

Note that only the Shoulder and Elbow joints employ dual motors. This design choice arises from the significant torque required to lift the entire downstream kinematic chain, including subsequent joints, the end-effector, and the carried payload. Given the size and torque constraints, a single motor would either fail to reliably meet operational torque requirements or

TABLE I: WIDOWX 250 S SERVO MAPPING AND JOINT LIMITS.

Joint	**Joint limits**		**Servo ID(s)**
	Min	**Max**	
Waist (Joint 1)	$-180°$	$180°$	1
Shoulder (Joint 2)	$-108°$	$114°$	2 + 3
Elbow (Joint 3)	$-123°$	$92°$	4 + 5
Forearm Roll (Joint 4)	$-180°$	$180°$	6
Wrist Angle (Joint 5)	$-100°$	$123°$	7
Wrist Rotate (Joint 6)	$-180°$	$180°$	8
Gripper	30 *mm*	74 *mm*	9

run excessively hot under continuous load. A secondary (”shadow”) servo in parallel effectively doubles available torque at these critical joints, ensuring reliable operation and improved thermal management.

Additionally, WidowX 250 S includes embedded feedback sensors within each DYNAMIXEL motor, allowing precise position, velocity, and torque control. This facilitates robust motion planning, dynamic adaptation during manipulation tasks, and improved overall operational safety and efficiency.

All the specifications are summarized in the table below [15]:

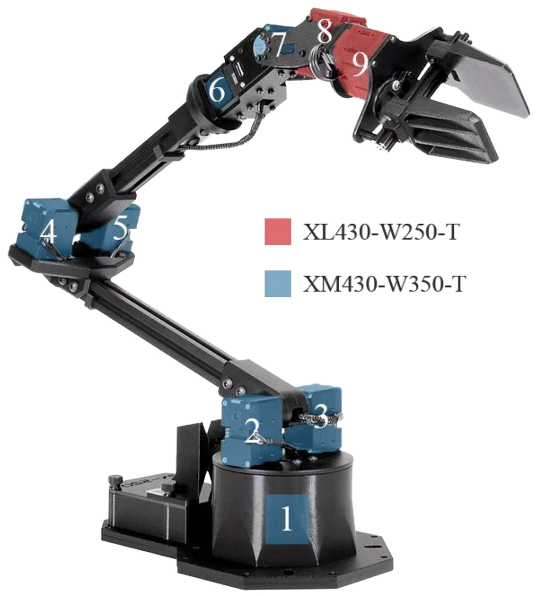


Figure 1: WidowX 250 S with highlighted motors. Adapted from [3]: modifications by the author.

3.1.2 Forward kinematics

Forward kinematics refers to the computation of the robot's end-effector pose from a given set of joint angles $\boldsymbol{q}$. The pose encompasses both the position and orientation of the end-effector with respect to a predefined reference frame. In the following, we provide the derivation of the forward kinematic function for WidowX 250 S, which is an open-chain manipulator, using Denavit-Hartenberg parameters, which constitute a systematic and conventional method for kinematic analysis. The end-effector's pose can be mathematically characterized by specifying the position vector of its origin along with the unit vectors that define its attached coordinate

TABLE II: WIDOWX 250 S SPECIFICATIONS

Degrees of freedom	6 degrees
Reach	650 mm
Span	1300 mm
Repeatability	1 mm
Accuracy	5-8 mm
Servomotors	Seven XM430-W350-T, Two XL430-W250-T
Power supply	12 V - 5 A
Weight	2.8 kg

frame, where all quantities are referenced to the coordinate system of the robot's base. This complete representation is efficiently encoded using a homogeneous transformation matrix, expressed as $\boldsymbol{T}_e^b(\boldsymbol{q})$, in which the subscript e denotes the end-effector coordinate frame and the superscript b designates the base coordinate frame [17]. The Denavit-Hartenberg convention provides a systematic approach for determining the relative position and orientation between two consecutive links in a kinematic chain. This method follows a specific set of rules for defining both the reference frames of the links and the parameters that describe the relative transformation between them. The four DH parameters are conventionally denoted as follows:

- a: Link length;

- α: Link twist;
- d: Link offset;
- θ: Joint angle.

These parameters are employed to construct the homogeneous transformation matrix between two consecutive frames ($i-1$ and i), which follows the standard DH structure:

$$\boldsymbol{T}_i^{i-1}(q_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\,\cos\alpha_i & \sin\theta_i\,\sin\alpha_i & a_i\,\cos\theta_i \\ \sin\theta_i & \cos\theta_i\,\cos\alpha_i & -\cos\theta_i\,\sin\alpha_i & a_i\,\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.1}$$

It is important to note that each homogeneous transformation matrix from frame i to frame $i-1$ depends only on one of the four parameters, referred to as the joint variable [17]. Since WidowX manipulator has only revolute joints, the joint variable q_i of joint i always corresponds to the joint angle θ_i, up to a constant offset. By applying the DH convention and utilizing WidowX's specifications and technical drawings, all the DH parameters can be determined and are summarized in Table III. Note that δ is a constant whose value is $\operatorname{atan2}(0.25, 0.05) \approx 78.69°$ (see Figure 3). This constant is introduced based on the robot's structure to align the joint variables' zero position with the motors' presets. Figure 2 illustrates the chosen link frames overlaid on the side view of the robot arm's technical drawing.

TABLE III: DENAVIT-HARTENBERG PARAMETERS FOR WIDOWX 250 S

Joint	$a\,(m)$	$\alpha\,(rad)$	$d\,(m)$	$\theta\,(rad)$
1	0	$-\frac{\pi}{2}$	0.11025	q_1
2	0.25495	0	0	$q_2 - \delta$
3	0	$-\frac{\pi}{2}$	0	$q_3 - \left(\frac{\pi}{2} - \delta\right)$
4	0	$\frac{\pi}{2}$	0.250	q_4
5	0	$-\frac{\pi}{2}$	0	q_5
6	0	0	0.15875	q_6

The reference frame (frame 0), whose origin position along the z-axis is technically non-unique according to the DH convention, is chosen to match the default position and orientation of the base frame in the URDF file of the manipulator, an XML-based model that defines its physical and kinematic structure, essentially considered the blueprint of the robot used to simulate, control, and visualize it. This choice allows the subscript 0 to be interchanged with the superscript b in our notation. To ensure that the end-effector frame is perfectly aligned with the base frame when the arm is in its home position (where all joint positions equal zero), as shown

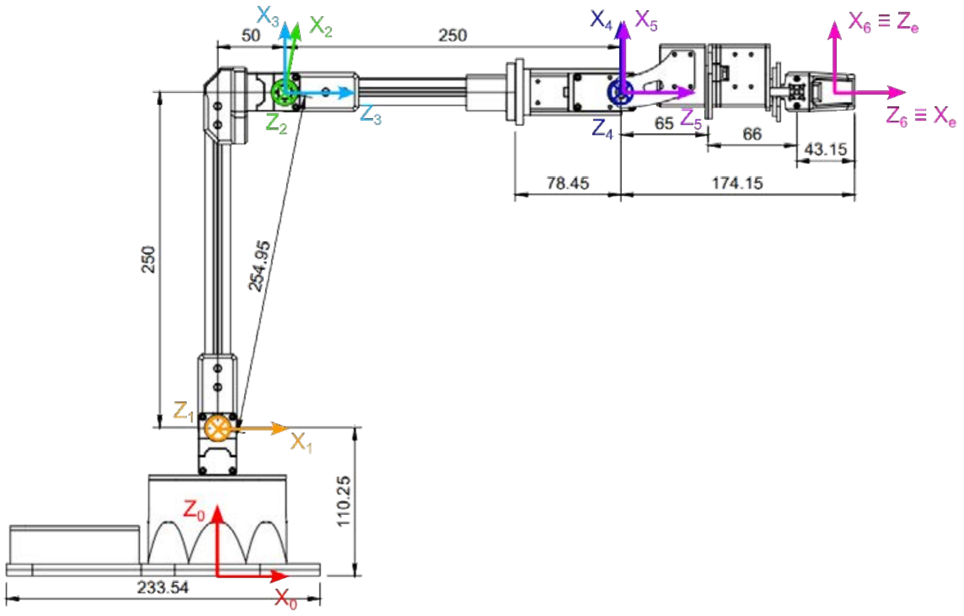


Figure 2: WidowX 250 S technical drawing with Denavit-Hartenberg frames. Adapted from [4]: modifications by the author.

in Figure 2, an additional constant homogeneous transformation matrix $\boldsymbol{T}_e^6$ is incorporated into the matrix multiplication and is given as follows:

$$\boldsymbol{T}_e^6 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

This transformation is essentially a pure rotation that accounts for the desired end-effector orientation. The complete forward kinematic function is obtained by pre-multiplying all transformation matrices from the base frame to the end-effector frame:

$$\boldsymbol{T}_e^b(\boldsymbol{q}) = \boldsymbol{T}_e^0(\boldsymbol{q}) = \boldsymbol{T}_1^0(q_1)\boldsymbol{T}_2^1(q_2)\boldsymbol{T}_3^2(q_3)\boldsymbol{T}_4^3(q_4)\boldsymbol{T}_5^4(q_5)\boldsymbol{T}_6^5(q_6)\boldsymbol{T}_e^6 \tag{3.3}$$

This equation represents the complete forward kinematic transformation that maps the joint space configuration $\boldsymbol{q}$ to the end-effector pose in the base frame coordinate system.

3.1.3 Inverse kinematics

Working with relative position between robotic arm and target in the Cartesian space, real-time grasp planning necessitates evaluating the inverse kinematics mapping the end-effector pose $\boldsymbol{x}_e = \left(\boldsymbol{p}_e^\top, \boldsymbol{\phi}_e^\top\right)^\top$ into the joint position vector $\boldsymbol{q}$ at a very fast rate, to allow operation with little to no delay. Although iterative numerical methods such as Gauss–Newton or Levenberg–Marquardt can find solutions, they may result in greater latency overall, indeterminism when the initial guess is not good enough, and risk converging to undesirable or suboptimal solutions.

WidowX, being a 6-DOF manipulator with a spherical wrist, allows decoupling the inverse kinematics into a positional component (q_1, q_2, q_3), represented by the wrist center position, and an orientation component (q_4, q_5, q_6), that represents the end-effector orientation. An analytic solution therefore exists, delivering deterministic and high-speed computation of the order of a few milliseconds and circumventing the drawbacks mentioned above.

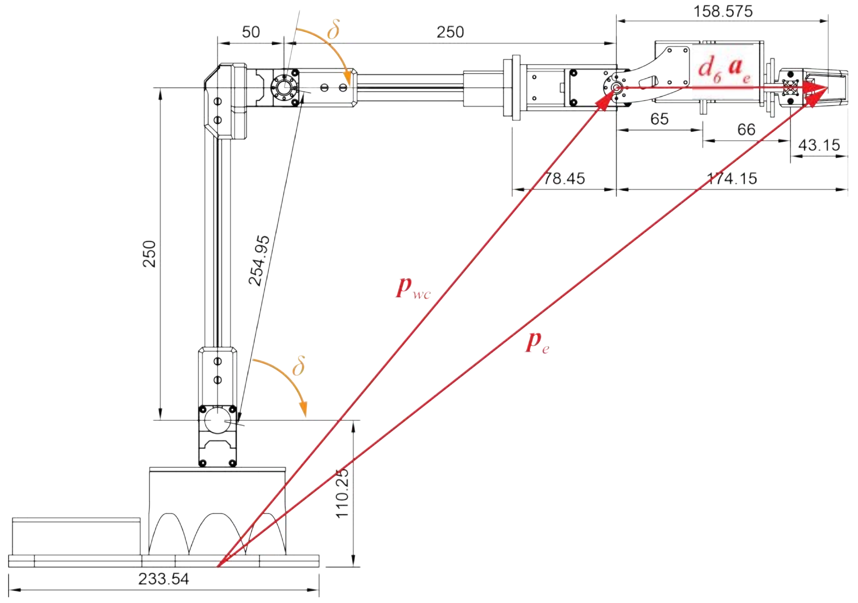


Figure 3: Wrist-center position vector $\boldsymbol{p}_{wc}$ from the end-effector pose $\boldsymbol{p}$.

In this derivation, the zero position for each joint is chosen to ensure a one-to-one correspondence with the robot's motor presets.

The analytical inverse kinematics solution begins by computing the wrist center, which is the point of intersection of the axes of the three joints that form the spherical wrist (Forearm Roll, Wrist Angle, Wrist Rotate, in the case of WidowX). Given a target end-effector position $\boldsymbol{p}_e$ and attitude $\boldsymbol{\phi}_e$ (expressed with a quaternion or Euler angles), the latter is first converted to rotation matrix form $\boldsymbol{R}_e^0 = [\boldsymbol{a}_e, \boldsymbol{s}_e, \boldsymbol{n}_e]$, then the wrist-center position $\boldsymbol{p}_{wc}$ is computed by offsetting the end-effector position by the known distance d_6 from Table III along its approach vector $\boldsymbol{a}_e$:

$$\boldsymbol{p}_{wc} = \boldsymbol{p}_e - d_6 \boldsymbol{a}_e. \tag{3.4}$$

Note that, as mentioned above, by design choice, the approach direction of the end-effector is along the x-axis of its frame. From this result, the ”cylindrical coordinates” of the wrist center relative to the base frame, visible in Figure 4a, are given by:

$$r_{wc} = \pm\sqrt{p_{wc,x}^2 + p_{wc,y}^2}, \quad z_{wc} = p_{wc,z} - d_1. \tag{3.5}$$

They are not technically cylindrical coordinates, since r_{wc} should also assume negative values to represent all feasible poses accurately. To attach a meaningful sign, the following steps are performed. First, the horizontal distance of the wrist center from the base z-axis is evaluated as the magnitude:

$$r_{wc}^* = \sqrt{p_{wc,x}^2 + p_{wc,y}^2} > 0. \tag{3.6}$$

Then, the horizontal unit vector is constructed as:

$$\boldsymbol{u}_e = \frac{(p_{e,x}, p_{e,y}, 0)^\top}{\|(p_{e,x}, p_{e,y}, 0)^\top\|} \tag{3.7}$$

and the following scalar projection is computed:

$$s = (p_{wc,x}, p_{wc,y}, 0)^\top \cdot \boldsymbol{u}_e. \tag{3.8}$$

The signed radial distance is then given by:

$$r_{wc} = \begin{cases} r^*_{wc} & \text{if } s \geq 0, \\ -r^*_{wc} & \text{if } s < 0. \end{cases} \tag{3.9}$$

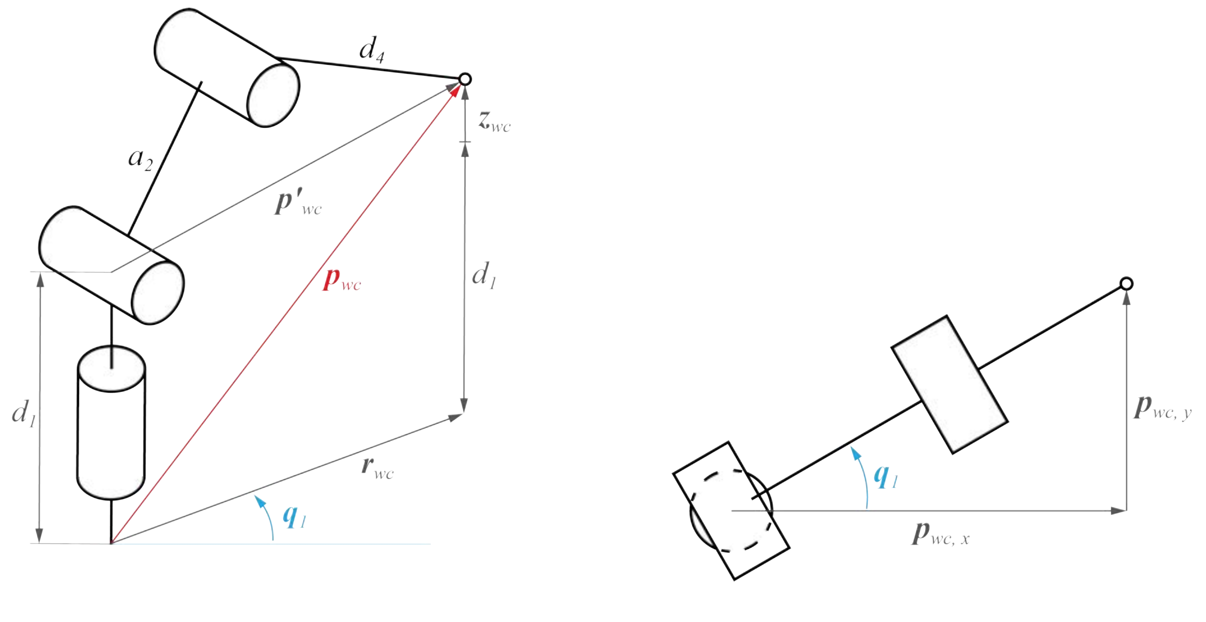
A negative value of r_{wc} signals that the wrist center lies on the opposite side of the z-axis of the base frame with respect to the center of the end-effector (also known as the tool center point). This single sign flip lets the subsequent formulas for q_1, q_2, and q_3 yield the correct configuration without extra branching.

At this point, the first joint angle can be obtained by applying simple trigonometry (see Figure 4b) and is directly computed as:

$$q_1 = \text{atan2}(p_{wc,y}, p_{wc,x}), \tag{3.10}$$

with a predetermined fallback approach if $r_{wc} \to 0$ to handle shoulder singularities robustly (Figure 4). The use of atan2, together with the signed values of $p_{wc,y}$ and $p_{wc,x}$ guarantees a solution in $(-\pi, \pi)$, which also corresponds to the range of feasible positions for the Waist joint.

Now that joint 1 is solved, the position of joints 2 and 3 can be determined by geometrically analyzing links 1 and 2 on the xy-plane of frame 1. Although the same wrist center position yields two possible inverse kinematic solutions, commonly referred to as the *elbow-up* and *elbow-down* configurations, the following analysis will focus exclusively on the *elbow-up* solution, as it is the only configuration relevant to the application at hand. Figure 5 shows an example



(a) (b)

Figure 4: 3D view (a) and top view (b) of the kinematic chain from the base frame up to the wrist center, with the wrist center position $\boldsymbol{p}_{wc}$ decomposed.

configuration and all the geometric entities involved. In this figure, both q_2 and q_3 are inwards, hence positive when clockwise. Moreover, the position of the links for which their value is zero is indicated by a thin blue line: following the adopted convention, the image then shows a configuration for which q_3 is negative. Given the lengths a_2 and d_4 from the DH parameters of Table III, the distance $\|\boldsymbol{p}'_{wc}\|$ between shoulder and wrist center is:

$$\|\boldsymbol{p}'_{wc}\| = \sqrt{r_{wc}^2 + z_{wc}^2}. \tag{3.11}$$

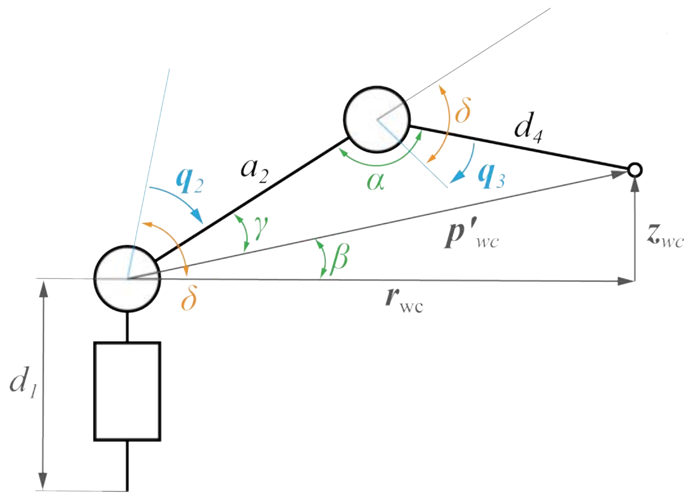


Figure 5: Geometric analysis of links 1 and 2 to solve joints 2 (Shoulder, q_2) and 3 (Elbow, q_3) given the wrist center position.

Applying the law of cosines (Figure 5):

$$\cos(\pi - \alpha) = \cos(\delta + q_3) = -\cos(\alpha) = \frac{\|\boldsymbol{p}'_{wc}\|^2 - a_2^2 - d_4^2}{2a_2d_4} \tag{3.12}$$

The joint angles q_2 and q_3 are then computed as:

$$q_2 = \delta - (\beta + \gamma), \tag{3.13}$$

$$q_3 = \pi - \alpha - \delta, \tag{3.14}$$

where $\delta = \text{atan2}(0.25, 0.05) \approx 1.373\ rad$ is the same angle present in Table III and shown in Figure 3, and angles β, γ can be obtained as:

$$\beta = \text{atan2}(z_{wc}, r_{wc}), \tag{3.15}$$

$$\gamma = \text{atan2}(d_4 \sin(\pi - \alpha), a_2 + d_4 \cos(\pi - \alpha)). \tag{3.16}$$

Note that the quantity $\cos(\pi - \alpha)$ also has another fundamental role: it determines whether the analyzed Cartesian pose actually has a feasible solution in the joint space. In fact, if

$$\left| \frac{\|\boldsymbol{p}'_{wc}\|^2 - a_2^2 - d_4^2}{2a_2 d_4} \right| > 1 \quad \Longrightarrow \quad |\cos(\pi - \alpha)| > 1, \tag{3.17}$$

the inverse kinematics problem admits no solution, and the target pose is out of reach. This is a key feature that will be later recalled, as it is employed in the logic that determines whether to start the grasp sequence.

Once the first three joint variables are determined, we can proceed to solve for the wrist joints. By utilizing the same matrix construction method and DH parameters as in the forward kinematics function, we can compute the homogeneous transformation matrix $T_3^b(q_1, q_2, q_3)$. From this matrix, we can isolate R_3^b and calculate the residual orientation R_6^3 as follows:

$$R_6^3 = (R_3^b)^\top R_6^b, \quad \text{with} \quad R_6^b = R_e^b (R_e^6)^\top. \tag{3.18}$$

Note that in order to work with the standard rotation matrix for spherical wrists (in our case R_6^3), it is necessary to "remove" the extra transformation from frame 6 to the end-effector frame (R_e^6). Cartesian positions of the gripper will be based on the end-effector-specific frame described in the analysis of the forward kinematics. The rotation matrix for the WidowX spherical wrist, expressed with respect to the DH frames of Figure 2, expands to:

$$R_6^3(q_4, q_5, q_6) = R_4^3(q_4)R_5^4(q_5)R_6^5(q_6) \tag{3.19}$$

$$= \begin{bmatrix} c_4 & 0 & s_4 \\ s_4 & 0 & -c_4 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} c_5 & 0 & -s_5 \\ s_5 & 0 & c_5 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} c_6 & -s_6 & 0 \\ s_6 & c_6 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

$$= \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & -c_4s_5 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & -s_4s_5 \\ s_5c_6 & -s_5s_6 & c_5 \end{bmatrix}, \tag{3.21}$$

with the shorthands $c_i = \cos q_i$ and $s_i = \sin q_i$. From this matrix:

$$s_5 = \pm\sqrt{\left(R^3_{6_{(1,3)}}\right)^2 + \left(R^3_{6_{(2,3)}}\right)^2}, \tag{3.22}$$

$$c_5 = R^3_{6_{(3,3)}}. \tag{3.23}$$

Because of the indeterminacy on the sign of s_5, the fifth joint admits two mathematically distinct values:

$$q_5^{(A)} = \text{atan2}(+s_5, c_5), \tag{3.24}$$

$$q_5^{(B)} = \text{atan2}(-s_5, c_5), \tag{3.25}$$

corresponding to the two possible (mirrored) orientations of the Wrist Angle joint. If $s_5 < 10^{-3}$ (chosen threshold), the wrist is in a singular configuration, and the solver sets:

$$q_4 = 0, \tag{3.26}$$

$$q_5 = 0, \tag{3.27}$$

$$q_6 = \text{atan2}\left(R^3_{6_{(2,1)}}, R^3_{6_{(1,1)}}\right). \tag{3.28}$$

Otherwise, two geometrically valid branches exist:

$$q_5^A = \text{atan2}(s_5, c_5),\ q_4^A = \text{atan2}\left(-R^3_{6_{(2,3)}}, -R^3_{6_{(1,3)}}\right),\ q_6^A = \text{atan2}\left(-R^3_{6_{(3,2)}}, R^3_{6_{(3,1)}}\right), \tag{3.29}$$

$$q_5^B = \text{atan2}(-s_5, c_5),\ q_4^B = \text{atan2}\left(R^3_{6_{(2,3)}}, R^3_{6_{(1,3)}}\right),\ q_6^B = \text{atan2}\left(R^3_{6_{(3,2)}}, -R^3_{6_{(3,1)}}\right). \tag{3.30}$$

By default, the implementation evaluates the squared Euclidean norms:

$$\left\|(q_4^A, q_5^A, q_6^A)\right\|^2 \quad \text{and} \quad \left\|(q_4^B, q_5^B, q_6^B)\right\|^2 \tag{3.31}$$

and retains the branch with the smaller value. This heuristic selects the solution nearest to the origin in joint space (and typically closest to the current configuration), thereby ensuring smooth wrist motion and avoiding abrupt orientation flips when the target pose varies incrementally.

Overall, this analytic inverse kinematic function uses fundamental trigonometric operations and a few small matrix multiplications, and ensures execution times of a few microseconds.

3.2 Differential kinematics

While forward and inverse kinematics establish the relationship between joint positions and end-effector pose, differential kinematics extends this analysis to the velocity domain and tries to capture the relationship between joint and end-effector velocities through the manipulator Jacobian matrix, which for a 6-DOF serial manipulator like WidowX is square (6×6). This mathematical framework becomes particularly crucial for dynamic grasping applications, where the robot must execute smooth, coordinated motions while tracking moving objects or adapting to changing target poses in real-time.

Since the end-effector's rotational velocity can be represented as either an angular velocity vector in space or as time rates of orientation changes, two types of Jacobian matrices arise in robotic differential kinematics: the geometric Jacobian and the analytic Jacobian.

Irrespective of the adopted framework, however, the following relation always holds:

$$\dot{\boldsymbol{x}}_e = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.32}$$

where $\dot{\boldsymbol{x}}_e, \dot{\boldsymbol{q}} \in \mathbb{R}^6$ are respectively the end-effector spatial velocity (or twist) and the joint velocity vectors. This relationship also shows that the Jacobian matrix is a function of the joint configuration $\boldsymbol{q}$, which implies that it is not a fixed entity and needs to be computed at runtime for each configuration under analysis.

3.2.1 Geometric Jacobian

When the end-effector spatial velocity vector $\dot{\boldsymbol{x}}_e$ is given as the combination of linear velocity $\boldsymbol{v}_e \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega}_e \in \mathbb{R}^3$ of the end-effector expressed in the base frame, the link between $\dot{\boldsymbol{x}}_e$ and $\dot{\boldsymbol{q}}$ is provided by the geometric Jacobian $\boldsymbol{J}(\boldsymbol{q})$.

Because all joints of WidowX are revolute, the i-th column of $\boldsymbol{J}$ is

$$\boldsymbol{J}_i = \begin{pmatrix} \boldsymbol{z}_{i-1} \times (\boldsymbol{p}_e - \boldsymbol{p}_{i-1}) \\ \boldsymbol{z}_{i-1} \end{pmatrix}, \qquad i = 1, \ldots, 6, \tag{3.33}$$

where the symbol $\times$ is used to denote the cross-product and the geometric entities are extracted from the homogeneous transformation matrices computed using the DH parameters from Table III:

- $\boldsymbol{p}_{i-1} \in \mathbb{R}^3$: position vector of the origin of frame $i-1$ relative to frame 0, namely the base frame, obtained as the translational component of $\boldsymbol{T}^0_{i-1}(\boldsymbol{q})$ [17];
- $\boldsymbol{z}_{i-1} \in \mathbb{R}^3$: unit vector along the z-axis of frame $i-1$ expressed in the base frame, extracted as the third column of the rotation matrix $\boldsymbol{R}^0_{i-1}$ from $\boldsymbol{T}^0_{i-1}(\boldsymbol{q})$;
- $\boldsymbol{p}_e \in \mathbb{R}^3$: position vector of the end-effector origin with respect to the base frame, obtained from $\boldsymbol{T}^0_e(\boldsymbol{q})$.

The transformation matrices are computed sequentially as usual:

$$\boldsymbol{T}_{i-1}^{0} = \boldsymbol{T}_{1}^{0} \cdot \boldsymbol{T}_{2}^{1}(q_2) \cdots \boldsymbol{T}_{i-1}^{i-2}(q_{i-1}) \tag{3.34}$$

and each individual transformation $\boldsymbol{T}_{j}^{j-1}(q_j)$ follows the standard DH structure presented in Section 3.1.2. From each transformation matrix $\boldsymbol{T}_{i-1}^{0}$, the required geometric quantities are then easily extracted as:

$$\boldsymbol{T}_{i-1}^{0} = \begin{bmatrix} \boldsymbol{R}_{i-1}^{0} & \boldsymbol{p}_{i-1} \\ \boldsymbol{0}^{T} & 1 \end{bmatrix} \tag{3.35}$$

$$\boldsymbol{z}_{i-1} = \boldsymbol{R}_{i-1}^{0} \cdot (0,0,1)^{\top} = \text{third column of } \boldsymbol{R}_{i-1}. \tag{3.36}$$

This form of the Jacobian matrix allows a more intuitive representation of the rotational motion of the end-effector as it moves through three-dimensional space.

3.2.2 Analytic Jacobian

Unlike the geometric Jacobian, the analytic Jacobian $\boldsymbol{J}_A(\boldsymbol{q}, \boldsymbol{\phi}_e)$ maps joint velocities $\dot{\boldsymbol{q}}$ to the time derivative of the end-effector pose expressed in terms of position and Euler angle rates:

$$\dot{\boldsymbol{x}}_a = \begin{pmatrix} \dot{\boldsymbol{p}}_e \\ \dot{\boldsymbol{\phi}}_e \end{pmatrix} = \boldsymbol{J}_A(\boldsymbol{q}, \boldsymbol{\phi}_e)\dot{\boldsymbol{q}} \tag{3.37}$$

where $\boldsymbol{\phi}_e = (\varphi_e, \theta_e, \psi_e)^T$ denotes the Z-Y-X intrinsic Euler angles (roll–pitch–yaw).

The angular velocity $\boldsymbol{\omega}_e$ of the end-effector is related to the Euler angle rates through a configuration-dependent transformation:

$$\boldsymbol{\omega}_e = \boldsymbol{T}(\boldsymbol{\phi}_e)\dot{\boldsymbol{\phi}}_e \tag{3.38}$$

with the transformation matrix given by:

$$\boldsymbol{T}(\boldsymbol{\phi}_e) = \begin{bmatrix} 1 & 0 & -\sin\theta_e \\ 0 & \cos\varphi_e & \cos\theta_e \sin\varphi_e \\ 0 & -\sin\varphi_e & \cos\theta_e \cos\varphi_e \end{bmatrix} \tag{3.39}$$

To compute the Euler angle rates from angular velocity, the inverse of this matrix is applied:

$$\dot{\boldsymbol{\phi}}_e = \boldsymbol{T}^{-1}(\boldsymbol{\phi}_e)\boldsymbol{\omega}_e \tag{3.40}$$

where the inverse has the explicit form:

$$\boldsymbol{T}^{-1}(\boldsymbol{\phi}_e) = \begin{bmatrix} 1 & \sin\varphi_e \tan\theta_e & \cos\varphi_e \tan\theta_e \\ 0 & \cos\varphi_e & -\sin\varphi_e \\ 0 & \sin\varphi_e / \cos\theta_e & \cos\varphi_e / \cos\theta_e \end{bmatrix} \tag{3.41}$$

Note that $\boldsymbol{T}^{-1}(\boldsymbol{\phi}_e)$ becomes singular when $\cos\theta_e = 0$, corresponding to pitch angles of $\theta_e = \pm\pi/2$. This condition, that occurs when the Euler representation loses rank, is known as gimbal lock and results in the loss of one degree of freedom in orientation control. To avoid undesired behavior, this singularity must be carefully accounted for during the design and implementation of trajectory planning algorithms, particularly when joint velocities are computed from end-effector velocities expressed using Euler angle rates. In practice, to mitigate numerical instability, the value of $\cos\theta_e$ can be clipped to a small nonzero threshold near zero (e.g., 10^{-4}) during computation to ensure safe matrix inversion.

To incorporate this relationship into the Jacobian, we define the block transformation matrix $\boldsymbol{T}_A(\boldsymbol{\phi}_e) \in \mathbb{R}^{6\times 6}$ for which:

$$\boldsymbol{T}_A^{-1}(\boldsymbol{\phi}_e) = \begin{bmatrix} \boldsymbol{I}_3 & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \boldsymbol{T}^{-1}(\boldsymbol{\phi}_e) \end{bmatrix} \tag{3.42}$$

It is important to note that the angular component of the geometric Jacobian $\boldsymbol{J}(\boldsymbol{q})$ typically expresses angular velocities in the end-effector frame, whereas the matrix $\boldsymbol{T}^{-1}(\boldsymbol{\phi}_e)$ expects them in the base frame. Therefore, before applying $\boldsymbol{T}^{-1}$, the angular velocity component must be rotated from the end-effector frame to the base frame using the rotation matrix $\boldsymbol{R}_e^0$ derived from the end-effector orientation:

$$\boldsymbol{J}_A(\boldsymbol{q}, \boldsymbol{\phi}_e) = \begin{bmatrix} \boldsymbol{I}_3 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{T}^{-1}(\boldsymbol{\phi}_e)\boldsymbol{R}_e^0 \end{bmatrix} \boldsymbol{J}(\boldsymbol{q}) \tag{3.43}$$

A relevant feature of the analytic Jacobian is its ”dual dependency” on both the joint configuration $\boldsymbol{q}$ and the current end-effector orientation $\boldsymbol{\phi}_e$. This is due to the presence of the nonlinear, orientation-dependent mapping $\boldsymbol{T}_A(\boldsymbol{\phi}_e)$, which varies with the pose of the manipulator.

The analytic Jacobian is especially beneficial in control frameworks that operate in Euler angle coordinates, enabling direct control over orientation rates without requiring intermediate conversions from angular velocity to Euler rate during runtime. However, the analytic Jacobian has a relevant downside: it does not always exist. In fact, all the angular velocities for which the end-effector attitude is such that

$$\sin(\theta_e) = 0 \tag{3.44}$$

cannot be expressed using $\dot{\phi}_e$. These attitudes, called representation singularities, lead to a null determinant of the tranformation matrix and should be avoided when planning trajectories in the Cartesian space with Euler angles rates, that are later converted to joint velocities using the analytic Jacobian.

3.2.3 Inverse Jacobian

The inverse of the Jacobian matrix plays a critical role in differential kinematics, particularly when solving for joint velocities given a desired end-effector velocity:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\,\dot{\boldsymbol{x}}_e \tag{3.45}$$

where $\boldsymbol{J}^{-1}(\boldsymbol{q})$ denotes the inverse of the Jacobian.

For square and full-rank Jacobian matrices, like in the case of WidowX, the inverse is uniquely defined. However, in redundant or near-singular configurations, a regular inverse does not exist or may yield numerically unstable solutions. To address this, we employ the damped least squares (DLS) method, which provides a robust approximation even near singularities.

Given the singular value decomposition of the Jacobian:

$$\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{3.46}$$

where $\boldsymbol{\Sigma} = \mathbf{diag}(\sigma_1, \ldots, \sigma_r)$ contains the singular values, the damped pseudoinverse is defined as:

$$\boldsymbol{J}^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T \tag{3.47}$$

with the regularized inverse of each singular value given by:

$$\sigma_i^\dagger = \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \tag{3.48}$$

where $\lambda > 0$ is the damping factor.

In practice, a small threshold ε is used to distinguish well-conditioned Jacobians. When the minimum singular value satisfies $\min_i(\sigma_i) > \varepsilon$, the undamped pseudoinverse is used:

$$\boldsymbol{J}^{-1} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T \tag{3.49}$$

Otherwise, the damped pseudoinverse $\boldsymbol{J}^\dagger$ is computed using Equation 3.47, and adopted in place of $\boldsymbol{J}^{-1}$.

This technique ensures numerical stability and smooth joint motions even in configurations close to singularities.

3.3 Trajectory Planning

Effective execution of dynamic grasping tasks, particularly those involving objects with non-zero relative velocities with respect to the robot's base frame, demands precise trajectory control to accurately synchronize the robot end-effector with the target object. Under such circumstances, Cartesian-space trajectory planning is preferred over joint-space planning due to its inherent ability to directly manage the end-effector's position and orientation at each time step. This explicit control simplifies the enforcement of precision constraints and task-specific requirements, such as maintaining a constant orientation or following a predefined geometric path.

However, Cartesian-space planning comes with certain drawbacks. Notably, it requires continuous computation of inverse kinematics within the control loop, and it does not inherently avoid singularities. These limitations can be significantly mitigated by avoiding known singular configurations and by employing the analytic inverse kinematic function previously derived in Subsection 3.1.3.

In the following subsections, we present all the elements, considerations, and strategies adopted to generate time-optimal end-effector trajectories required for grasping objects undergoing constant linear motion with respect to the robot.

3.3.1 Multi-DOF planning with Ruckig

Planning trajectories for dynamic grasping of moving objects necessitates a high degree of flexibility and responsiveness, as well as the ability to accommodate non-zero initial and final velocities. Additionally, it is crucial to consider the full range of constraints, including maximum allowable velocity and acceleration of the robot end-effector, as well as the current and target state (position and velocity) of the end-effector.

To satisfy these demanding requirements, the Ruckig library was adopted. Ruckig is a state-of-the-art motion planning library specifically designed for robotic applications, offering jerk-limited and highly time-optimized trajectory generation for multi-degree-of-freedom systems. It produces solutions in extremely short computation times, typically under 1 ms [18][19]. The framework also allows users to define a minimum duration for a trajectory, which acts as a soft constraint. If no feasible trajectory satisfies the constraints within this minimum duration, the solver computes the shortest possible trajectory that still fulfills all requirements.

Since Ruckig outputs profiles for position, velocity, and acceleration across each degree of freedom, the most practical approach for handling orientation components is to use Euler angles. Although quaternions generally provide better robustness against singularities in rotation representation, Euler angles are more suitable for integration with Ruckig's API, which enables specifying custom constraints per degree of freedom. This integration can directly influence the computed duration of the trajectory. As a consequence of this design choice, any potential issues related to gimbal lock are addressed separately during the computation of the target orientation.

It is important to note that each generated trajectory requires, in addition to kinematic constraints, accurate input regarding both the object's current position and velocity in the base frame and the end-effector's current state in the same reference frame.

3.3.2 Grasp point estimation

The predicted grasp point is computed using a straightforward linear extrapolation, under the assumption of constant object velocity. Letting $\boldsymbol{p}_{obj}$ denote the current position and $\boldsymbol{v}_{obj}$ the linear velocity of the object, the future grasp point at time t_{grasp} is estimated as:

$$\boldsymbol{p}_{grasp} = \boldsymbol{p}_{obj} + \boldsymbol{v}_{obj} t_{grasp} \tag{3.50}$$

This basic extrapolation plays a crucial role in the generation of the main trajectory during the grasping phase, as it defines the spatial goal that the end-effector must reach.

3.3.3 Motion-aware end-effector orientation strategy

By design, even though the grasp point can be predicted in advance, the robot arm is programmed to track the object for a brief period before executing the grasp. This strategy serves two main purposes. First, it allows the controller to remain adaptive to sudden variations in the object's relative motion, ensuring smoother and more reactive behavior from the end-effector, particularly when the object's velocity changes in magnitude or direction. Second, and equally important, is the fact that the gripper requires some time to close to the desired aperture (approximately 0.7 s). This duration practically introduces significant uncertainty when attempting to perform a grasp with a static end-effector positioned at a predicted grasp point. If the gripper closes too early or too late, the object may no longer be within its grasping range, resulting in failure. This issue is further exacerbated by the fact that the required timing for a successful grasp depends on the size of the object—the larger the object, the earlier the gripper can begin to close. Consequently, achieving consistent success with a purely predictive approach would demand an impractically high level of precision and tight synchronization across sensing, planning, and actuation subsystems. By allowing the arm to hover and track the object briefly before grasping, the system effectively mitigates these risks and increases the robustness of the grasping behavior. To facilitate this behavior while avoiding premature contact with the object, the desired orientation of the end-effector is dynamically adjusted based on the observed motion. This orientation computation assumes that the object either rests on or moves along a surface (not necessarily horizontal) and is never suspended freely in space.

If the object is in motion, the direction of motion is used as the normal direction of the end-effector (z-axis):

$$\boldsymbol{z}_e = \frac{\boldsymbol{v}_{obj}}{\|\boldsymbol{v}_{obj}\|}. \tag{3.51}$$

Then, the desired slide direction (y-axis) is computed as a vector orthogonal to the projection of the velocity on the x_0y_0-plane:

$$\boldsymbol{y}_e = \frac{(-v_{obj,y}, v_{obj,x}, 0)^\top}{\|(-v_{obj,y}, v_{obj,x}, 0)\|}. \tag{3.52}$$

The negative sign before the y-component is chosen based on the structure of the end-effector's local frame, such that the approach direction (x-axis) typically points downward. An example of a generated desired orientation is shown in Figure 6.

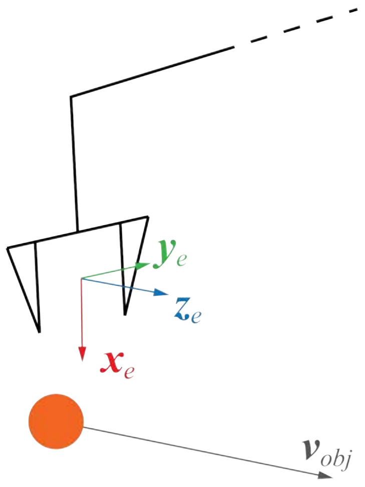


Figure 6: Example of desired end-effector orientation based on the object's motion.

To minimize unnecessary wrist rotations and remain within joint limits, the algorithm evaluates whether mirroring the normal direction $\boldsymbol{z}_e$ about the base frame z-axis $(0, 0, 1)$ would result in a better alignment with the object. This is done by comparing the alignment scores:

$$s_+ = \boldsymbol{z}_e^\top \boldsymbol{p}_{obj}, \quad s_- = \left(2(\boldsymbol{z}_e^\top \boldsymbol{z}_0)\boldsymbol{z}_0 - \boldsymbol{z}_e\right)^\top \boldsymbol{p}_{obj} \tag{3.53}$$

where $\boldsymbol{z}_0 = (0\ 0\ 1)^\top$. If $s_- > s_+$, then the mirrored version is chosen as the new z-axis:

$$\boldsymbol{z}_e \leftarrow 2(\boldsymbol{z}_e^\top \boldsymbol{z}_0)\boldsymbol{z}_0 - \boldsymbol{z}_e, \quad \boldsymbol{y}_e \leftarrow -\boldsymbol{y}_e. \tag{3.54}$$

This ensures the end-effector maintains a desirable orientation while minimizing abrupt angular displacements. Finally, the approach direction (x-axis) is computed as the cross product:

$$\boldsymbol{x}_e = \boldsymbol{y}_e \times \boldsymbol{z}_e. \tag{3.55}$$

When the object is stationary or its speed is below 1 mm/s, the orientation is derived from its position vector instead:

$$\boldsymbol{z}_e = \frac{\boldsymbol{p}_{obj}}{\|\boldsymbol{p}_{obj}\|}. \tag{3.56}$$

The slide direction is then computed as:

$$\boldsymbol{y}_e = \frac{(-p_{obj,y}, p_{obj,x}, 0)^\top}{\|(-p_{obj,y}, p_{obj,x}, 0)\|}. \tag{3.57}$$

The orthonormal frame is completed using the same cross product:

$$\boldsymbol{x}_e = \boldsymbol{y}_e \times \boldsymbol{z}_e. \tag{3.58}$$

With this strategy, when the object is not moving relative to the base frame of WidowX, the gripper is allowed to approach the object frontally, without any rotation of the Forearm Roll and the Wrist Rotate joints.

Additionally, in both stationary and non-statioray cases, a small corrective rotation about the y-axis is applied if $\boldsymbol{x}_e$ is nearly aligned with the global downward direction $[0, 0, -1]$, in order to avoid singularities in orientation.

3.3.4 Main Trajectory

As previously discussed, the objective of the main trajectory, which is the one used to grasp a moving object, is to bring the end-effector above the object, for a short time, with the correct orientation and a velocity that matches that of the object, to then proceed to approach and grasp the object.

This main trajectory can be described as a sequence of four distinct yet connected segments:

- First segment: this segment guides the end-effector from its current pose (Figure 7a) to a position directly above the object, at a distance o_a, referred to as the approach offset (Figure 7b). During this segment, the end-effector adjusts its orientation and velocity to match those of the object. A little negative contribution to the velocity along the z-axis is added in order to ”anticipate” the motion of the second segment, allowing to save some time overall. Notably, this is the only segment in which the orientation of the end-effector is altered.

- Second segment: this phase moves the gripper from the approach offset o_a down to a closer distance to the object, practically around it, at the grasp offset o_g, enabling the robot to prepare for the grasp (Figure 7c).

- Third segment: this is a brief constant-velocity segment during which the end-effector tracks the object while waiting for the gripper to fully close, thereby completing the grasp.
- Fourth segment: this final phase brings the end-effector to a complete stop.

Both the approach and grasp offsets (o_a and o_g) are configurable parameters that should be adjusted depending on the object's dimensions and geometry. Importantly, each segment after the first uses the final state of the preceding segment as its initial state, ensuring continuity in both position and velocity across the trajectory. Each of these trajectory segments is computed independently. This is necessary because the Community Version of Ruckig does not support the specification of intermediate waypoints that include both position and velocity constraints [19]. Additionally, this segmentation enables the use of different offline solvers provided by Ruckig: specifically, the last two segments are computed using a velocity-control approach without imposing position constraints.

To ensure that the overall trajectory is time-efficient, the trajectory planner dynamically determines the duration of the first segment, which typically exhibits the greatest variability due to the uncertain initial distance between the end-effector and the moving object at the time a new trajectory is generated. This is achieved by solving a preliminary optimization problem that estimates a lower bound for the segment duration, based on the current distance and the kinematic limits of the end-effector, which are the maximum velocity and maximum acceleration of the end-effector. An initial trajectory solution is then computed and its resulting duration is compared to the estimated lower bound. If the durations match, the solution is accepted

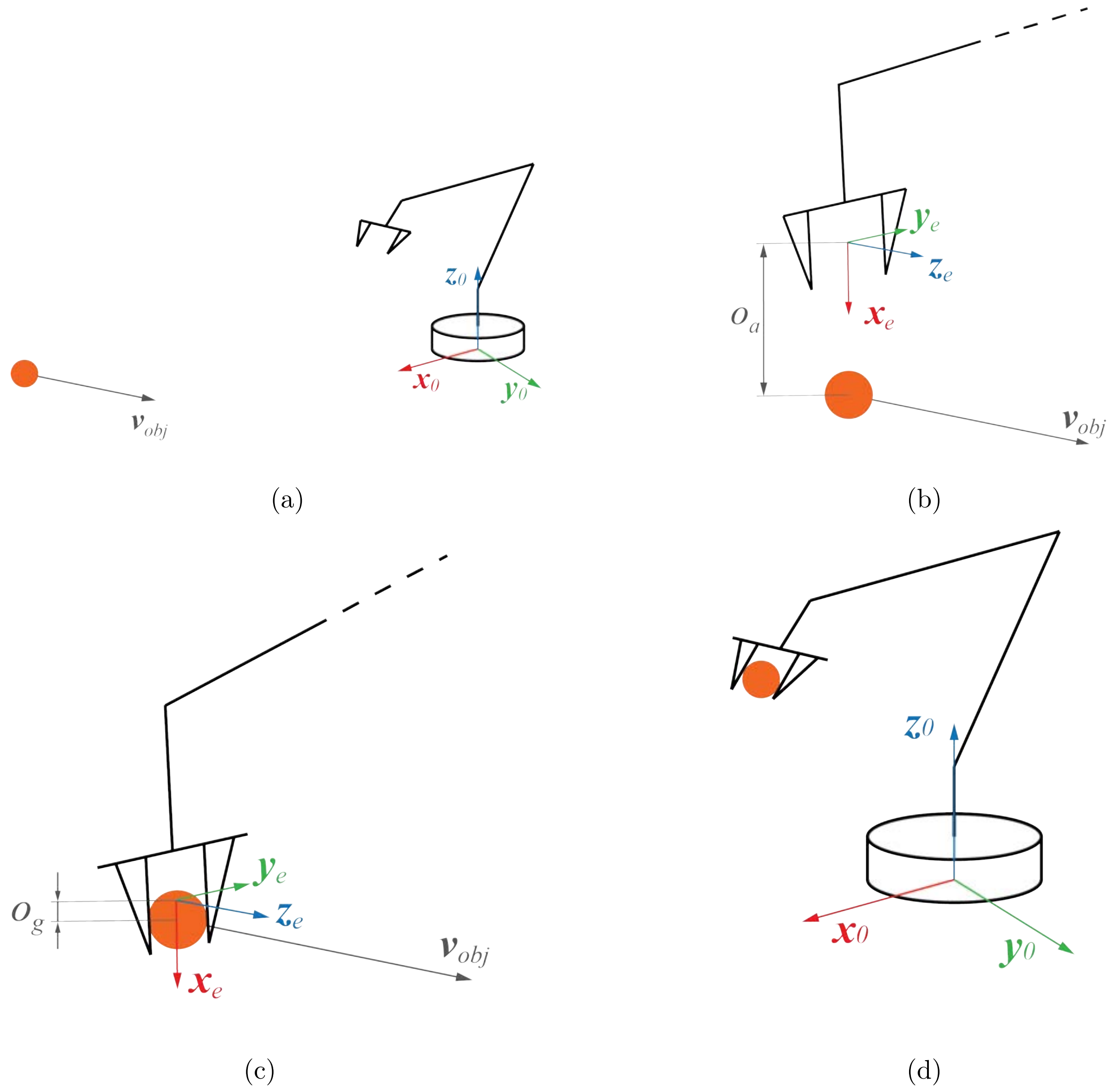


Figure 7: Main phases of the trajectory to grasp a moving object.

and the planner proceeds to solve the second segment. If the computed duration exceeds the bound, the planner stores the new duration and increases it by 10%. The expected end-effector state is then re-evaluated using the most recent measurements, and the Ruckig solver is called again to generate a new solution. This iterative process continues until the duration output by Ruckig's solver matches the duration provided as input, ensuring that the end-effector reaches the predicted target point precisely at the expected time.

The 10% increment is a design choice aimed at relaxing the lower bound constraint. While this results in a slightly longer trajectory, it reduces the number of iterations needed and thus accelerates the overall trajectory generation process. The feasibility of this iterative scheme is enabled by the high computational efficiency of Ruckig's trajectory generation.

In contrast, the durations of the subsequent trajectory segments are fixed and determined a priori, based on the velocity and acceleration limits of the end-effector and validated through simulation testing to ensure grasp success across a wide range of relative velocities. Specifically, the duration of the first segment typically ranges from 2 s to 4 s, while the second and third segments are both set to last 1 s. The fourth segment does not have a predetermined duration, as the planner attempts to decelerate the end-effector to a stop as quickly as possible.

3.3.5 Stop trajectory

While the main trajectory governs the entire grasping motion, it is not sufficient to handle all dynamic scenarios encountered during execution. To enhance the adaptability of the robotic arm to changes in the target's relative motion, the system must be capable of recognizing when such changes occur and reacting accordingly. In particular, the robot should halt its motion until the object's trajectory has stabilized. To accommodate this requirement, an additional trajectory type is introduced, referred to as the stop trajectory.

When the planner receives an external command to interrupt the current motion, Ruckig is used to compute a deceleration trajectory that brings the end-effector to a complete stop starting from its current state. Functionally, this operation is nearly identical to the fourth segment of the main trajectory, with the key difference that it can be triggered asynchronously whenever needed.

3.3.6 Go-to-rest trajectory

A successful grasping task requires not only reaching and securing the object but also ensuring that the robot transitions to a safe and stable post-grasp configuration. Relying solely on the main trajectory would leave the robot near the grasp point after completion, which is not ideal for further operations or readiness for subsequent tasks.

To address this, a dedicated trajectory is designed to return the robot arm to a predefined configuration named rest pose, illustrated in Figure 7d. In this pose, the links between the Shoulder and Wrist Angle joints are positioned horizontally, and the wrist center is located on the same side of the base frame's z-axis as the tool center point. This configuration is considered safe, as it retracts the robot away from potential collisions and avoids proximity to a shoulder singularity.

The rest pose is defined both in joint space and in Cartesian space, as follows:

$$\boldsymbol{q}_{rest} = (0, -1.8, 1.55, 0, 0.8, 0)^\top \tag{3.59}$$

$$\boldsymbol{x}_{e,rest} = (0.174, 0, 0.171, 0, 0.247, 0, 0.969)^{\top} \tag{3.60}$$

Following a successful grasp, once the planner receives the command to transition to the rest pose, Ruckig generates a smooth trajectory from the current end-effector state to the predefined configuration. This motion is designed to be completed in 3 s, as depicted in Figure 7d. This newly computed trajectory replaces the final segment of the main trajectory, which is retained only to allow the end-effector to stop in case the grasp fails or is aborted.

3.3.7 Discretization and Conversion of the Trajectory

Although all trajectories are generated in Cartesian space, the robot's position controllers operate in joint space. Therefore, an additional processing step is required to convert the planned trajectories into a format compatible with the control architecture.

The first step is to discretize the continuous Cartesian trajectory profiles. This is achieved by sampling the desired position and velocity profiles at a specified interval, referred to as the sampling time. In this work, a sampling interval of $\delta_t = 50\ ms$ is used, which has been shown to offer a good trade-off between trajectory fidelity and computational efficiency. If the total duration of a segment is not a multiple of δ_t, the final point of the segment is sampled at its actual timestamp to ensure that the full trajectory is represented.

After discretization, all segments that belong to the same motion (such as those composing the main trajectory) are concatenated into a single trajectory. Care is taken to avoid duplicating points at segment boundaries, since each new segment is initialized using the final state of the preceding one.

The resulting set of Cartesian trajectory points is then transformed into joint space. This involves computing the joint configurations using the analytical inverse kinematics solver, and deriving the corresponding joint velocities by multiplying the Cartesian velocity by the inverse of the analytical Jacobian matrix:

$$\dot{\boldsymbol{q}}_{t_i} = \boldsymbol{J}_A^{-1}(\boldsymbol{q}_{t_i}, \boldsymbol{\phi}_{e,t_i})\dot{\boldsymbol{x}}_{e,t_i}. \tag{3.61}$$

Each trajectory point retains its original timestamp throughout the conversion process. Once this transformation is complete, the entire trajectory—now represented in joint space and formatted appropriately—is transmitted to the robot's controller for execution.

3.4 Gripper Trajectory Planner

The gripper trajectory planner is responsible for generating position-based control commands for the WidowX's gripper in response to high-level symbolic inputs (like ”Released” or ”Grasping”) or numerical aperture values. Although the actuation mechanism of the gripper is a revolute joint, of the same type as that of the Wrist Rotate joint, the gripper planner abstracts this complexity by operating entirely in terms of linear aperture widths. This design choice allows intuitive control over the gripper, specifying how far apart the fingers should be rather than the angle they should rotate.

A command specifying a desired aperture width w_{des} is internally converted into a target joint position g_{target} using the following mapping:

$$g_{target} = \frac{w_{des} + w_{offset}}{2} \tag{3.62}$$

where $w_{off} = 1.8\ cm$ compensates for the fixed offset between the contact point of each finger and its connection with the joint servomotor. This formula accounts for the symmetric movement of the fingers: to achieve an opening of w_{des}, each finger must move half that distance from the center, plus the mechanical offset.

The trajectory planner supports both preset and custom commands. Presets include:

- *Grasping*: closes the gripper to a tight position suitable for grasping small objects.
- *Released*: fully opens the gripper.
- *Home*: a neutral or intermediate opening position.

Alternatively, a numeric width can be specified, provided it does not exceed a maximum allowed width. Invalid or out-of-range inputs are rejected to ensure mechanical safety. Once a valid command is received, the planner generates a time-parameterized trajectory consisting of three waypoints: the current finger position, a midpoint, and the final target position. This helps ensure smooth and gradual motion, reducing undesired motion that could affect object interaction. A simple time-based monitoring mechanism ensures that the motion completes within a specified window; otherwise, the system resets to a safe state.

A boolean state variable indicating whether the gripper is closed is also maintained. This state is derived based on the most recently executed command: any command other than *Released* is interpreted as a closed gripper state.

This approach enables reliable and user-friendly control over the gripper by abstracting low-level actuation and enforcing smooth, safe trajectories.

3.5 Relative state estimation

3.5.1 Streaming of position data

To enable real-time tracking of rigid bodies within the robot's workspace, this work employs a motion capture (MoCap) system by OptiTrack. The data acquisition is performed using the NatNet streaming protocol, which transmits position and orientation data with minimal latency over a local wireless or wired network. In practice, this allows for reliable motion capture performance even in mobile setups or when operating without physical connections between the tracking system and the robot controller.

The data stream is received and interpreted using a software interface provided directly by OptiTrack; it is an open-source package designed for integration with robotic platforms [20]. This package is fully compatible with the ROS 2 middleware, but it can also act as a generic client for NatNet streams and is capable of publishing the tracked pose data in a standardized format at high frequency. Each rigid body's pose is output as a separate message, timestamped and expressed with respect to a global inertial frame defined by the MoCap system in its dedicated software. This continuous stream of accurate pose measurements serves as the primary input for the velocity estimation and relative localization modules described in the following subsections.

3.5.2 Velocity estimation of rigid bodies from position data

Unfortunately, position data from MoCap suffer from micro-level jitter at high frequency and the adopted setup, with a mobile base, is characterized by mechanical vibration due to the moving base, which mainly affect the motion on the vertical axis (z-axis). The presence of these factors makes discrete differentiation an unreliable method to obtain the velocity from position data, as it would greatly amplify noise leading to chattery velocity signals.

To robustly estimate the linear and angular velocity of a rigid body from noisy position measurements, this work implements a method based on linear regression over a sliding window of timestamped pose samples. This approach is well-suited to scenarios where the rigid body are expected to exhibit approximately a constant linear motion over a short temporal window and where the available data are rich in temporal resolution, such as when using a motion capture system. Under these assumptions, within such a window, the trajectory of the object can be locally approximated by a linear model, enabling the application of least-squares regression. Since the robot arm will be able to recognize and plan trajectories to grasp only objects moving with a constant linear velocity, this velocity estimator will only estimate the translational components of the velocity vector, by determining the slope of a best-fit line through recent position measurements.

Let us now delve into the adopted linear regression method for estimating linear velocity from position data. Let the sliding window contain $N \geq 3$ samples, each consisting of a timestamp $t_i \in \mathbb{R}$ and a corresponding position $\boldsymbol{p}_i \in \mathbb{R}^3$, expressed with respect to the origin of the fixed frame of the vision system.

The goal is to model the position trajectory over time as a linear function:

$$\boldsymbol{p}(t) = \boldsymbol{v} \cdot t + \boldsymbol{b}, \tag{3.63}$$

where $\boldsymbol{v} \in \mathbb{R}^3$ is the constant linear velocity vector (i.e., the slope), and $\boldsymbol{b} \in \mathbb{R}^3$ is a constant bias term. The vector $\boldsymbol{v}$ is what we aim to estimate using a least-squares fit.

To improve numerical stability and simplify computation, we center the timestamps around their mean:

$$\tilde{t}_i = t_i - \bar{t}, \quad \text{where} \quad \bar{t} = \frac{1}{N}\sum_{i=1}^{N} t_i. \tag{3.64}$$

This centering ensures that the design matrix has zero mean, which leads to an unbiased and numerically stable estimation of the slope.

The least-squares regression tries to determine the parameter vector $\boldsymbol{v}$ that minimizes the sum of squared residuals, which is given by:

$$\min_{\boldsymbol{v}} \sum_{i=1}^{N} \|\boldsymbol{p}_i - (\boldsymbol{v} \cdot t_i + \boldsymbol{b})\|^2. \tag{3.65}$$

When timestamps are centered around their mean, the optimal solution for the slope vector becomes:

$$\boldsymbol{v} = \frac{\sum_{i=1}^{N} \tilde{t}_i (\boldsymbol{p}_i - \bar{\boldsymbol{p}})}{\sum_{i=1}^{N} \tilde{t}_i^2}, \tag{3.66}$$

where

- $\bar{\boldsymbol{p}} = \dfrac{1}{N}\sum_{i=1}^{N} \boldsymbol{p}_i$ is the mean position vector in the window;
- $\tilde{t}_i$ is the centered time, which shifts the time vector to have zero mean;
- $\boldsymbol{p}_i - \bar{\boldsymbol{p}}$ is the deviation of the i-th position sample from the mean position.;
- the numerator $\sum \tilde{t}_i(\boldsymbol{p}_i - \bar{\boldsymbol{p}})$ is the sample covariance between time and position;
- the denominator $\sum \tilde{t}_i^2$ is the sample variance of time.

This formula yields the slope of the line that best fits the time-position data in the least-squares sense. This estimate provides a robust approximation of the rigid body's translational velocity over the duration of the sliding window.

It can be seen that the regression estimate is a multivariate analog of the classical 1-dimensional slope formula from statistics:

$$\text{slope} = \frac{\text{Cov}(t, p)}{\text{Var}(t)}. \tag{3.67}$$

The sliding window linear regression method provides a simple yet effective solution for real-time velocity estimation, without relying on a predefined motion model or knowledge of the system dynamics. The primary tunable parameter governing the behavior of this estimator is the window size. Increasing the number of samples in the window enhances smoothing and yields a more stable velocity estimate, particularly beneficial when the target moves at constant speed. However, this comes at the cost of reduced responsiveness to abrupt changes in motion, effectively introducing latency in the output.

As a result, the choice of window size represents a trade-off between smoothness and responsiveness, and must be carefully tuned based on the sampling characteristics of the vision system and the temporal requirements of the application. In this work, the motion capture system used during experimental validation provides pose updates at a rate of 240 Hz, while maintaining full coverage of the workspace. Given this sampling frequency, the estimator operates with a window of 120 samples, corresponding to a temporal span of 0.5 s. This configuration offers a highly smooth velocity profile while maintaining a bounded maximum delay of 0.5 s, which is deemed acceptable for the requirements of the system under study.

3.5.3 Relative state estimation in the robot's reference frame

As the aim of this project is to perform grasping of objects in a dynamic environment where both robot arm and target can be non-stationary, it is not sufficient to know the absolute position and velocity of an object with respect to a fixed inertial frame. As a matter of fact, the robot's control system, as explained in the precedent sections, requires the object's state relative to its base frame (frame 0) to work as expected. As a consequence, velocity data obtained from the estimator needs to be processed further.

The goal of the relative state estimator is to determined the object's state with respect to the robot base frame.

Let $\boldsymbol{p}_{obj}^{w}$ and $\boldsymbol{p}_{0}^{w}$ denote the absolute positions of the object and the robot, respectively, both expressed in a common inertial "world" frame, denoted as w. Let $\boldsymbol{R}_{0}^{w}$ represent the rotation matrix that describes the orientation of the robot's base frame (frame 0) in relation to the world frame. The position of the object, as expressed in the robot's frame, is given by:

$$\boldsymbol{p}'_{obj} = \boldsymbol{p}^0_{obj} = (\boldsymbol{R}^w_0)^\top (\boldsymbol{p}^w_{obj} - \boldsymbol{p}^w_0), \tag{3.68}$$

where the term $\boldsymbol{p}^w_{obj} - \boldsymbol{p}^w_0$ represents the vector from the robot to the object in the world frame, and the rotation $(\boldsymbol{R}^w_0)^\top = \boldsymbol{R}^0_w$ transforms it into the robot's local frame.

A similar reasoning applies to linear velocities. Let $\boldsymbol{v}^w_{obj}$ and $\boldsymbol{v}^w_0$ be the linear velocities of the object and robot, respectively, expressed in the world frame. Then, the relative velocity of the object with respect to the robot, expressed in the robot's frame, is:

$$\boldsymbol{v}'_{obj} = \boldsymbol{v}^0_{obj} = (\boldsymbol{R}^w_0)^\top (\boldsymbol{v}^w_{obj} - \boldsymbol{v}^w_0). \tag{3.69}$$

This transformation follows directly from the Galilean principle of relative motion. The vector $\boldsymbol{v}^w_{obj} - \boldsymbol{v}^w_0$ represents the relative velocity between the two bodies in the world frame, and the rotation $\boldsymbol{R}^0_w$ maps it into the robot's coordinate system.

This formulation offers a mathematically consistent framework for expressing both the position and velocity of a moving object relative to the robot. From this point forward, we will refer to the object's position and velocity with respect to the robot arm's base frame simply as object position and object velocity. However, to clearly distinguish them from the absolute quantities, we will adopt the notation $\boldsymbol{p}'_{obj}$ and $\boldsymbol{v}'_{obj}$.

3.5.4 End-effector kinematics data

The last piece of the puzzle required to fully characterize the state of the robotic system is the position and linear velocity of the end-effector. These quantities can be computed from the joint states of the manipulator, under the assumption that the joint configuration $\boldsymbol{q}$ and the joint velocities $\dot{\boldsymbol{q}}$ are known at each time instant. This assumption is satisfied in practice, since each actuator in the robot provides direct feedback of its own joint state, including position and velocity data. These signals are then processed using the forward kinematics and the analytic Jacobian models introduced in Section 3.1.

3.6 Grasp-attempt decision logic and state machine

To manage the sequence of decisions required to dynamically grasp a moving object, this work implements a finite state machine that takes as input pose and velocity data of both the object and the end-effector, reacts to them, and plans the appropriate behavior for the robotic arm accordingly. The decision logic is evaluated periodically at a fixed rate of 30 Hz, ensuring that updates and transitions are responsive to fast-changing dynamics in both the object motion and robot state. A scheme of the state machine is shown in Figure 8. Each state corresponds to a specific sub-task in the overall grasping pipeline, and transitions between states are triggered based on both sensor feedback and logical conditions derived from predictive trajectory analysis.

in the following, a brief overview of the states:

- **Idle**: The system always begins in this inactive state (marked in red in Figure 8), waiting for the start command to initiate the grasp sequence. It is also set to return to this state every time a grasp is completed successfully.

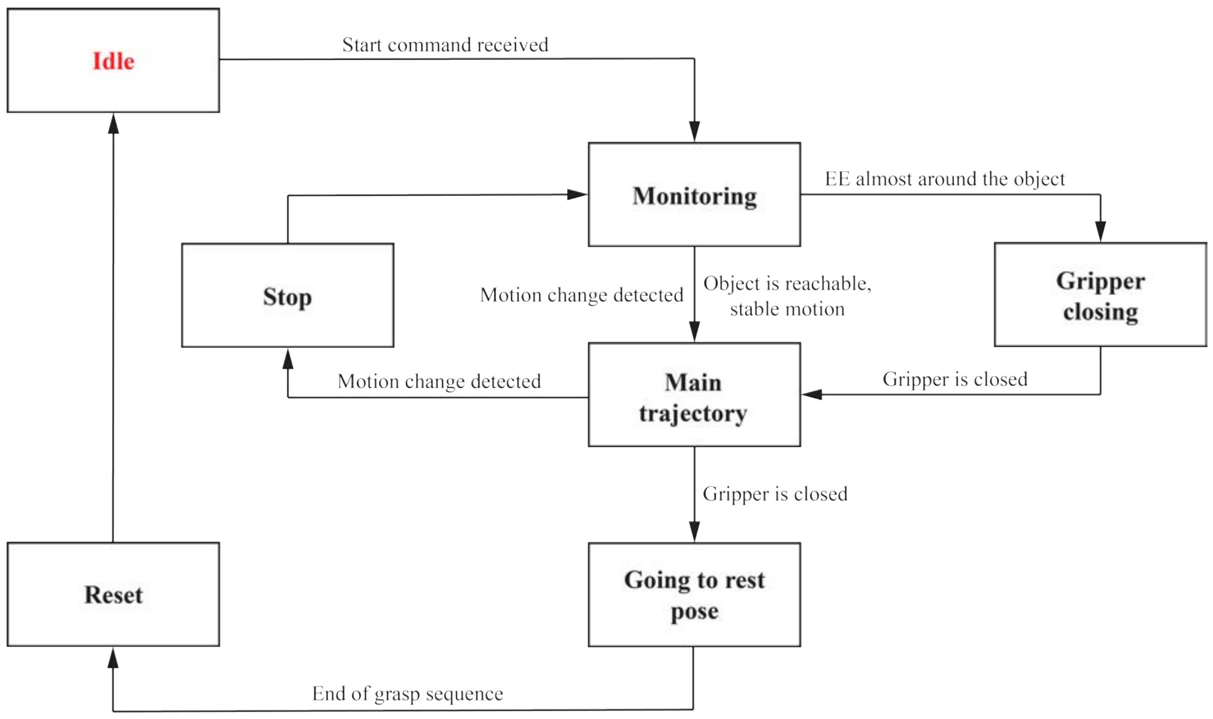


Figure 8: Simplified finite state machine of the dynamic grasping control system.

- **Monitoring**: It is a state that is always active whenever the system is not idling. After the start command, the system begins monitoring the position and velocity of both the object and the end-effector. In this state, if the grasp sequence has not started yet, the decision logic continuously evaluates whether the object has a stable trajectory and conditions are favorable to attempt a grasp. It is also responsible for determining when to stop the robot's movement if the object deviates from steady linear motion.

- **Main trajectory**: If the object is predicted to become reachable under suitable motion conditions, a trajectory is initiated to intercept the object. In this state, the planner generates a trajectory to grasp the moving target and sends it to the motors' controllers, which execute it.

- **Gripper closing**: As the end-effector nears the predicted interception point, the system checks whether it has entered the final grasp zone. Once sufficiently close, a signal is issued to close the gripper.

- **Going to rest pose**: After the grasp is confirmed, the system tells the robot to transition to the rest pose, a safe post-grasp configuration. This marks the conclusion of the manipulation phase.

- **Reset**: This state performs a full reset of internal variables and flags, clearing any residual state before preparing for a new grasping attempt. From here, the system returns to the Idle state once the reset phase is over.

- **Stop**: This emergency state is triggered if a sudden change in the object's motion is detected during grasp execution. The grasp attempt is aborted, and the planner generates a trajectory for a smooth stop of the arm. The system then keeps monitoring and awaits a new stable condition before resuming.

3.6.1 Predictive trajectory analysis and grasp triggering conditions

The core of the grasping decision logic lies in forecasting the object's motion and evaluating whether the robot can reach it in time. The core parameters that are evaluated at each cycle, on which the decision logic is based, are:

- $\tau_{obj,out}$: the estimated future time at which the object will enter a defined graspable workspace;
- $\tau_{obj,in}$: the expected duration for which the object will remain in that workspace;
- $\boldsymbol{p}'_{obj,in}$: the predicted position at which the object will enter the graspable region.

Given the quantities above, the distance between the end-effector and the entry point is computed as:

$$d_e = \left\|\boldsymbol{p}'_{obj,in} - \boldsymbol{p}_e\right\| \tag{3.70}$$

Based on this distance, the minimum time τ_e required by the robot to reach the object is estimated using a trapezoidal velocity profile, which is meant to approximate the actual trajectory time needed by the end-effector to move from its current state to being over the object once the latter enters the reachable workspace (see Figure 7b). In this profile, the end-effector first accelerates uniformly up to a maximum velocity $v_{e,max}$ over a time interval t_{acc}, cruises at constant speed for a duration t_{cruise}, and finally decelerates symmetrically back to zero. The acceleration phase covers a distance:

$$s_{acc} = \frac{1}{2}a_{e,max}t_{acc}^2 \tag{3.71}$$

and the time required to reach maximum speed is:

$$t_{acc} = \frac{v_{e,max}}{a_{e,max}}. \tag{3.72}$$

Since the deceleration phase is symmetric, the total distance covered during acceleration and deceleration is:

$$s_{ramp} = 2 \cdot s_{acc} = a_{e,max} \left(\frac{v_{e,max}}{a_{e,max}} \right)^2 = \frac{v_{e,max}^2}{a_{e,max}}. \tag{3.73}$$

If the total displacement d_e is greater than s_{ramp}, then a constant-velocity cruising phase exists, and its duration is:

$$t_{cruise} = \frac{d_e - s_{ramp}}{v_{e,max}}. \tag{3.74}$$

Therefore, the total time required is the sum of the three segments:

$$\tau_e = 2 \cdot t_{acc} + t_{cruise} = 2 \cdot \frac{v_{e,max}}{a_{e,max}} + \frac{d_e - v_{e,max}^2 / a_{e,max}}{v_{e,max}}. \tag{3.75}$$

On the contrary, if the required displacement is short enough that the end-effector cannot reach its maximum velocity before needing to decelerate, a triangular velocity profile is used. In this case, the trajectory consists solely of an acceleration phase followed by a deceleration phase, both symmetric. Let t_{peak} be the time to reach the peak velocity before reversing the motion. Then the total displacement is:

$$d_e = 2 \cdot \left(\frac{1}{2} a_{e,max} t_{peak}^2 \right) = a_{e,max} t_{peak}^2, \tag{3.76}$$

and solving for t_{peak} yields:

$$t_{peak} = \sqrt{\frac{d_e}{a_{e,max}}}. \tag{3.77}$$

Thus, the total time is:

$$\tau_e = 2 \cdot t_{peak} = 2 \cdot \sqrt{\frac{d_e}{a_{e,max}}}. \tag{3.78}$$

At the core of this decision pipeline is the ability to predict whether the object will enter a region that is truly reachable given the robot's kinematic constraints and the desired grasping orientation. Rather than relying on a predefined, static workspace volume, this work adopts a more adaptive and accurate approach. The object's future trajectory is extrapolated using a constant velocity model, and a sequence of 150 discrete positions is sampled along this path at a spatial resolution of 1 cm, using a time step inversely proportional to the object's speed. This corresponds to analyzing a segment of 1500 mm, which is about 15% longer than the WidowX's maximum span (see Table II). At each predicted position, the target grasping orientation is computed using the function described in Subsection 3.3.3, and a reduced version of the robot's analytic inverse kinematic function, based on Equation 3.17's principle, evaluates whether a valid solution exists that places the end-effector above that position with the required orientation. Only positions for which a valid joint configuration exists are considered reachable. This strategy effectively defines a dynamic, orientation-aware reachable workspace that adapts to both the object's path and the grasping constraints. Thanks to the efficiency of the closed-form inverse kinematics solver, this reachability check is executed in real time as part of each decision cycle. This approach ensures that grasp triggering decisions are based not just on geometric proximity, but on actual kinematic feasibility, which is particularly important when certain orientations restrict reachability or when the robot is operating near its joint limits.

Figure 9 illustrates a representative scenario in which a moving object crosses the reachable workspace of the robotic arm, potentially triggering the grasp sequence. The sketch highlights the main variables involved in the trajectory analysis and grasp decision process, offering a visual aid to better understand their geometric and temporal relationships. Note that the elements shown in pink do not represent physical distances; rather, they indicate the time required for the object or the end-effector to traverse the corresponding path segments under constant or planned motion conditions.

Once all these variables are computed, the system evaluates whether the conditions for initiating a grasp are met. The decision primarily depends on the object's current position relative to the robot's workspace and its velocity characteristics. More precisely, the situation is handled differently if the object is outside the workspace compared to when it is inside. If the object is currently outside the reachable workspace but predicted to enter it in the near future, the system evaluates whether the predicted motion allows for a timely and feasible grasp. Specifically:

- The predicted duration of stay inside the workspace must be sufficiently long: $\tau_{obj,in} > 2\ s$. This threshold is set to 2 s since it is know that the end effector takes roughly 1.6 s to grasp the object once it is on top of it (Figure 7b). The remaining 0.4 s is used to leave some tolerance.

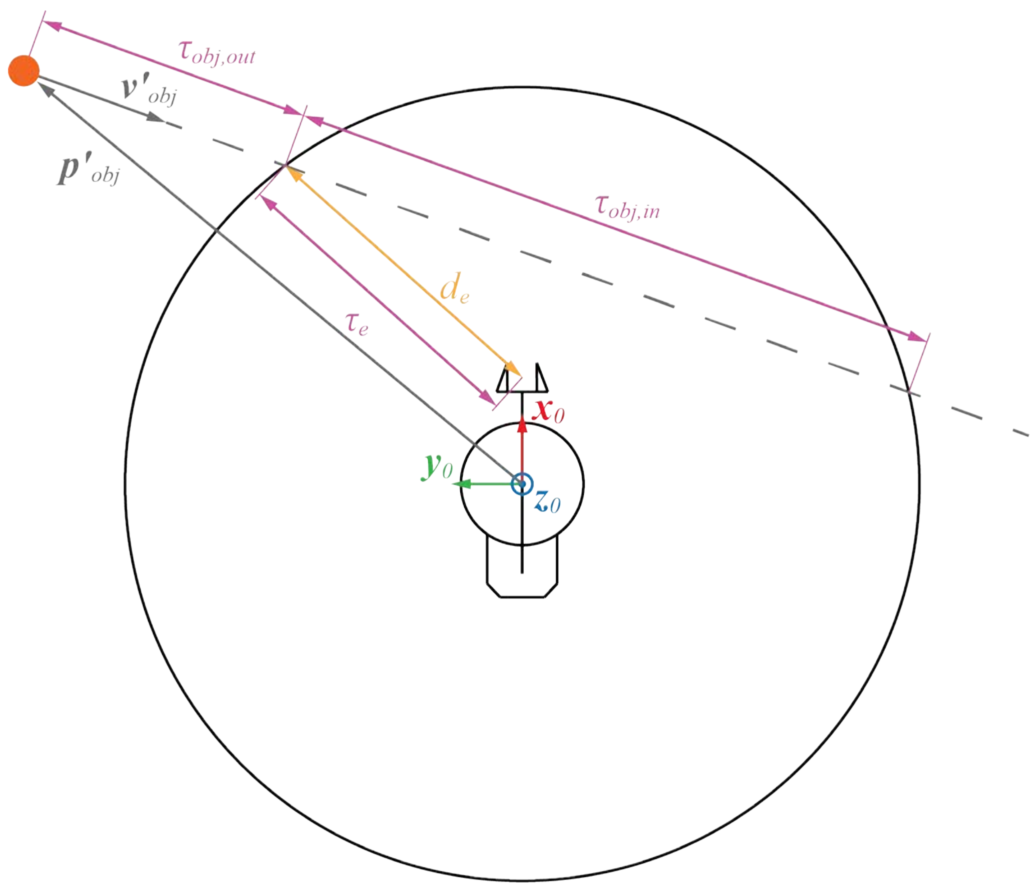


Figure 9: Illustration of the key variables involved in the grasp triggering logic in a typical scenario.

- For moving objects, their speed must be lower than the maximum end-effector speed, to allow for it to catch up:

$$\|\boldsymbol{v}_{obj}\| < 0.85 \cdot v_{e,max} \tag{3.79}$$

- For moving objects, the predicted entry time must fall within a feasible timing window. This ensures that the robot can reach the object before its stay in the workspace gets below the minimum required duration of 2 s. Specifically, the entry time must satisfy:

$$\max\left(0, \tau_e - (\tau_{obj,in} - 2)\right) \leq \tau_{obj,out} \leq \tau_e \tag{3.80}$$

  The lower bound of this interval guarantees that, even if the object arrives at the entry point earlier than the robot, it remains within the workspace long enough for the robot to complete the approach and execute the grasp. The offset $(\tau_{obj,in} - 2)$ represents the excess duration the object is expected to stay inside the workspace beyond the minimal required grasping window. The upper bound simply ensures that the object does not arrive later than the robot, which would result in a missed interception. The lower limit is clamped to zero to exclude cases where the object is already inside the workspace, which are handled separately. Appendix A shows an example of this metric for clarity.

Note that for stationary objects, no timing alignment is required between $\tau_{obj,out}$ and τ_e, since the object is expected to remain in place for the full duration.

In case the object is already inside the graspable region (for example when its motion stabilized again after a direction change), the decision logic simplifies and only the remaining dwell time is relevant:

- For stationary objects: the grasp can be initiated immediately, without further checks.

- For moving objects: the object must be expected to remain in the workspace for a longer duration to allow for the full execution. Specifically:

$$\tau_{obj,in} > 4\ s \tag{3.81}$$

The speed constraint of (Equation 3.79) still applies. Although it may initially appear advantageous for the object to already lie within the reachable workspace, this situation can, in fact, present more challenging conditions for initiating a successful grasp sequence. Unlike the case in which the object is predicted to enter the graspable region, which allows the robot to preemptively plan and synchronize its motion thanks to a longer time window, an object that is already inside the workspace may have already traversed a portion of it and is typically moving away from the end-effector's current position at the moment the motion begins. This introduces a key uncertainty: the spatial location at which the end-effector will successfully intercept the object, and the corresponding time at which this alignment will occur, are both unknown a priori and depend dynamically on the robot's acceleration profile and the object's current trajectory. Consequently, defining a reliable condition based on a minimum required duration of stay within the workspace becomes less straightforward. Moreover, even if one attempted to determine the minimum time required to intercept the object using an optimization-based formulation, it would be no assurance that the Ruckig-based trajectory planner will generate a trajectory that has exactly that duration, since, as mentioned before, this is considered a soft constraint. As a result, establishing a consistent and predictive threshold for triggering the grasp based solely on motion duration is not practical in this scenario. A 4 s condition has, however, proven to be effective in most scenarios, especially when the object's relative speed is 40% or less of the maximum end-effector velocity.

Eventually, if the conditions in either scenario are satisfied, the system can send the command to start planning the main trajectory, so that the robot begins its approach toward the predicted grasping position.

3.6.2 Gripper actuation logic

While executing the grasp trajectory, the system continually monitors the distance d_{obj} between the tool center point and the object's center of mass. Once this distance drops below a threshold, the robot initiates the grasp by closing the gripper:

$$d_{obj} < o_g + \delta_g. \tag{3.82}$$

In this equation, o_g denotes the grasp offset introduced in Subsection 3.3.4, while δ_g is an additional parameter used to slightly anticipate the closing phase of the gripper—typically by a few hundredths of a second. This anticipation allows the system to conclude the grasp sequence earlier and transition more efficiently to the subsequent phase. For optimal performance, δ_g should be tuned according to the object's geometry and size, ensuring that the gripper begins to close as early as possible without risking premature closure before having the gripper around it. The gripper closure represents a crucial stage of the grasp sequence, as it also triggers the monitoring logic to transition from executing the main trajectory to initiating the return motion toward the rest configuration.

3.6.3 Direction change detection and grasp plan adaptation

A key feature of the system is the ability to detect and respond to changes in the object's velocity. This is done through a rolling buffer of recent velocity measurements. At each step k, the average velocity over a window of N samples, including the current one and the previous $N-1$ samples, is computed:

$$\boldsymbol{v}'^{(k)}_{obj,avg} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{v}'_{obj,i} \tag{3.83}$$

The magnitude of the difference with the previous average is then evaluated:

$$\Delta \left\| \boldsymbol{v}'_{obj} \right\| = \left\| \boldsymbol{v}'^{(k)}_{obj,avg} - \boldsymbol{v}'^{(k-1)}_{obj,avg} \right\| \tag{3.84}$$

If this exceeds a predefined threshold ε_{vel} before the gripper has closed:

$$\Delta \left\| \boldsymbol{v}'_{obj} \right\| \geq \varepsilon_{vel} \tag{3.85}$$

the robot smoothly halts its current operation after transitioning into the Stop state. A stabilization mechanism is then used to monitor the object's velocity until it becomes constant and linear again, defined as a return to variations below threshold over a sufficient number of samples. As soon as this happens, if the grasp is still feasible, the main trajectory may be retriggered.

3.6.4 Post-grasp reset and reinitialization

After a grasp attempt concludes, the system enters a reset state in which the monitoring logic and all internal buffers, flags, and motion history are restored to their original state as when the robot first started, ensuring a clean environment for the next grasp. Once this reset is complete, the system returns to its idling condition, awaiting the next trigger to begin a new cycle.

Ultimately, the described state machine, supported by predictive analysis and stability monitoring, enables robust decision-making in scenarios involving both stationary and dynamically moving objects, adapting in real time to ensure safe and efficient grasp execution.

CHAPTER 4

ROS 2 INTEGRATION AND SIMULATION FRAMEWORK

4.1 ROS 2 Humble as middleware backbone

4.1.1 Motivation and relevant features

To deploy the motion-control architecture developed in Chapter 3, ROS 2 Humble was selected as the middleware foundation. Humble is a stable version of ROS 2 and the current long-term-support release, which makes it ideal in many scenarios [21]. Additionally, ROS 2 Humble brings enhanced performance, real-time support, and improved security features essential for reliable motion control and future scalability in more complex robotic ecosystems. [21][22].

Unlike ROS 1, ROS 2 replaces the centralized master node with a peer-to-peer architecture based on the Data Distribution Service (DDS). This eliminates single points of failure and enables seamless communication across distributed systems, a key requirement for multi-robot and cloud-integrated deployments [23]. Consequently, each algorithmic element from Chapter 3, like trajectory planner, velocity estimator, state detector, grasp logic state machine (dynamic grasping manager), and motion commands, is encapsulated within its own C++ ROS 2 node. All these nodes then send data and information to each other using topics, the traditional ROS communication system. As an example, */wx250s/arm_controller/joint_trajectory* carries messages of time-stamped joint-space trajectory points, */wx250s/joint_states* streams position, velocity, and effort feedback from the robot arm, and */object_state* transmits the object's relative states to the dynamic grasping manager and the trajectory planner.

An important feature of ROS 2 that enhances this communication framework is its support for Quality of Service (QoS) profiles [23]. These profiles allow developers to configure topic-specific communication characteristics based on application requirements. For instance, joint state data can be configured with a QoS profile that prioritizes low latency over guaranteed delivery, since occasional message drops are acceptable given the high publication frequency. Conversely, finite state machine events require a reliable QoS profile to ensure that critical state transitions are never lost, maintaining system consistency. This modular, node-based architecture provides substantial benefits for system development and maintenance. Individual nodes can be independently developed, tested, and refined without affecting other system components. Furthermore, the loose coupling between nodes facilitates system evolution, allowing for component replacement or algorithm improvements without requiring extensive system-wide modifications. This design philosophy proves particularly advantageous during the iterative development process characteristic of research-oriented robotics projects, where algorithms frequently undergo refinement and optimization. The modular approach allows researchers to experiment with different implementations of individual components while maintaining system stability. Beyond development benefits, this architecture also enables seamless integration with other ROS 2-based robotic platforms, such as UGVs and quadrupedal robots. This compatibility opens possibilities for developing sophisticated loco-manipulation systems, where mobile platforms and manipulator arms work in coordination to achieve complex tasks.

4.1.2 Integration with Interbotix X-Series ROS 2 packages

The backbone of our implementation begins with the open-source Interbotix X-Series ROS 2 packages, officially supported by Trossen Robotics and fully compatible with Ubuntu 22.04 and ROS 2 Humble [24]. These packages serve as a robust foundation, providing both real-hardware and Gazebo setups for DYNAMIXEL-based X-Series arms, including the WidowX 250 S, through components such as:

- X-Series robot arm descriptions: URDFs, meshes, and realistic inertial properties for robot models.
- Low-level control: launch systems for starting SDK nodes, driver interfaces (*interbotix_xs_sdk*), and motor configuration.
- Simulation environment setup: Gazebo Classic simulation settings, including controller plugins and tuned parameters.
- ROS 2 control setup: enabling the joint trajectory controller integration for both simulation and the physical robot.

These core packages work together out of the box to provide a working simulation and real-robot stack, including Gazebo launch, RViz visualization, and SDK-based control of the WidowX 250 S. However, to align with the specific requirements of dynamic grasping and our custom control pipeline, this base functionality has been extended significantly, including the custom dynamic grasping control system, mission-specific robot parameters and setups, and personalized Gazebo simulations.

This layered approach preserves the reliability of the Interbotix stack, while allowing flexibility in tailoring the simulation, control, and execution pipeline to our grasping scenario. By maintaining Gazebo Classic compatibility and the standard ROS 2 node interfaces, this integration ensures that both simulated and real-world experiments share identical software structures and behavior.

4.2 Joint-Trajectory Controller as Final Execution Interface

Another essential component of the control pipeline is the *joint_trajectory_controller*, part of the ROS 2 control framework. This controller serves as the final execution interface between the high-level planning modules and the physical robot hardware [25]. It is responsible for receiving, interpolating, and forwarding joint trajectory commands to the low-level actuation layer at the right time. Trajectories can be sent either through an action server interface, ideal for monitored execution, or directly to the topic */wx250s/arm_controller/joint_trajectory* when feedback is not required.

A key feature of this controller is its built-in trajectory interpolation. When the incoming trajectory includes both position and velocity information at each waypoint, that is what the trajectory planner of Section 3.3 streams by default, the controller leverages a cubic spline interpolation scheme. This results in smooth motion profiles that are continuous in both position and velocity (C^1 continuity), ensuring that the generated joint commands do not introduce abrupt changes in motion [26]. By contrast, if velocity information is omitted, the controller defaults to linear interpolation, which often produces undesirable velocity discontinuities at waypoint boundaries and degrades overall tracking performance [26].

In terms of hardware compatibility, the controller is designed to support different types of interfaces. For systems that can operate using position control, such as the WidowX control system under analysis, it forwards position commands at the right time to the nodes responsible for controlling the robot by communicating with its U2D2 microcontroller. When working with velocity or effort interfaces, however, it employs internal PID loops to convert tracking errors into appropriate velocity or torque values. This versatility allows the same controller to be reused across platforms with different actuation schemes, without modifying the trajectory format or upstream logic.

Another valuable feature for real-time robotics applications is the controller's trajectory replacement capability. When a new trajectory is received during the execution of a previous one, the controller does not discard the old command entirely. Instead, it preserves the portion of the active trajectory that has not yet been executed and splices it with the new one, creating a continuous motion that avoids sharp transitions or halts [26][27]. This seamless merging is particularly effective in applications requiring reactive behavior, and therefore fits perfectly with how the dynamic grasping manager handles state transitions such as sudden stops or returns to the robot's rest pose. In fact, once these trajectories are generated, they are transmitted to the ROS 2 arm controller, which replaces the remaining portion of the current trajectory (typically the main one being executed) with the new incoming trajectory.

As mentioned before, during execution, the controller generates time-synchronized joint commands and forwards them to the *xs_sdk* node via the topic */widowx/commands/joint_tra jectory*. This SDK node acts as the interface to the hardware, translating ROS commands into device-level instructions. In simulation, the identical control structure is preserved by replacing the SDK with its simulated counterpart *xs_sdk_sim*, which mimics the same interfaces [28]. This consistent control pathway guarantees that behaviors validated in simulation transfer reliably to the physical robot, simplifying testing and deployment.

4.3 Gazebo environment and simulation setup

4.3.1 Gazebo and ROS 2 integration for realistic simulations

To validate the functionality and integration of the control architecture prior to hardware testing, all grasping experiments were first performed within the Gazebo simulation environment. Gazebo provides a high-fidelity physics engine capable of simulating the full dynamics of the robot, the environment, and any interacting objects, enabling safe and repeatable testing conditions [29].

At the core of any Gazebo simulation is the launch file of all physical elements in the scene through URDF (Unified Robot Description Format) files. Each element, like robot, table, or graspable object, must be instantiated with correct physical properties such as mass, inertia, collision geometry, and friction. Additionally, Gazebo-specific plugin tags are included in the URDF to provide additional specific functionalities like sensor simulation or controller interfaces.

For the Interbotix WidowX 250 S arm, the robot's URDF includes both the actual physical links and joints, as well as virtual frames used for referencing, planning, and control. One such frame is *ee_gripper_link*, which represents the tool center point and plays a critical role in trajectory generation and grasp planning. The URDF also includes a ROS 2 control plugin that provides Gazebo with the interface needed to receive position commands from the standard ROS 2 controller infrastructure and to publish joint state feedback.

During execution, the robot controller sends joint commands to the same joint trajectory controller used for real hardware, also using the same ROS 2 interface (*/wx250s/arm_controller/joint_trajectory*). In simulation, these commands are received by the Gazebo plugin, which applies them to the virtual model and simulates joint motion. The only feedback available from Gazebo, however, is joint position; joint velocities are not natively provided. To address this, a simple moving average filter is applied only in simulation to the differentiated joint position data. This approach yields sufficiently accurate velocity estimates for the purpose of monitoring and control, and it introduces negligible delay due to the short window size used. In the actual robot, this issue does not arise, as the servomotors provide a direct feedback of joint positions and velocities through the ROS 2 topic */wx250s/joint_states*.

Once Gazebo, ROS 2, and the control components are fully integrated, the simulation framework replicates the entire dynamic grasping pipeline under realistic conditions. Each module of the control architecture (trajectory planners, estimators, and the finite-state machine) operates as an independent ROS 2 node and communicates using the same interfaces configured for real-hardware execution. This consistent middleware structure ensures that simulation results are directly transferable to the physical robot, enabling a smooth transition between testing phases.

The complete structure of the system is illustrated in Figure 10. The diagram highlights the interaction between the Gazebo physics engine and the various nodes responsible for planning, estimation, and control. Note that feedback is processed through the same computational pipeline that will later handle real sensor data.

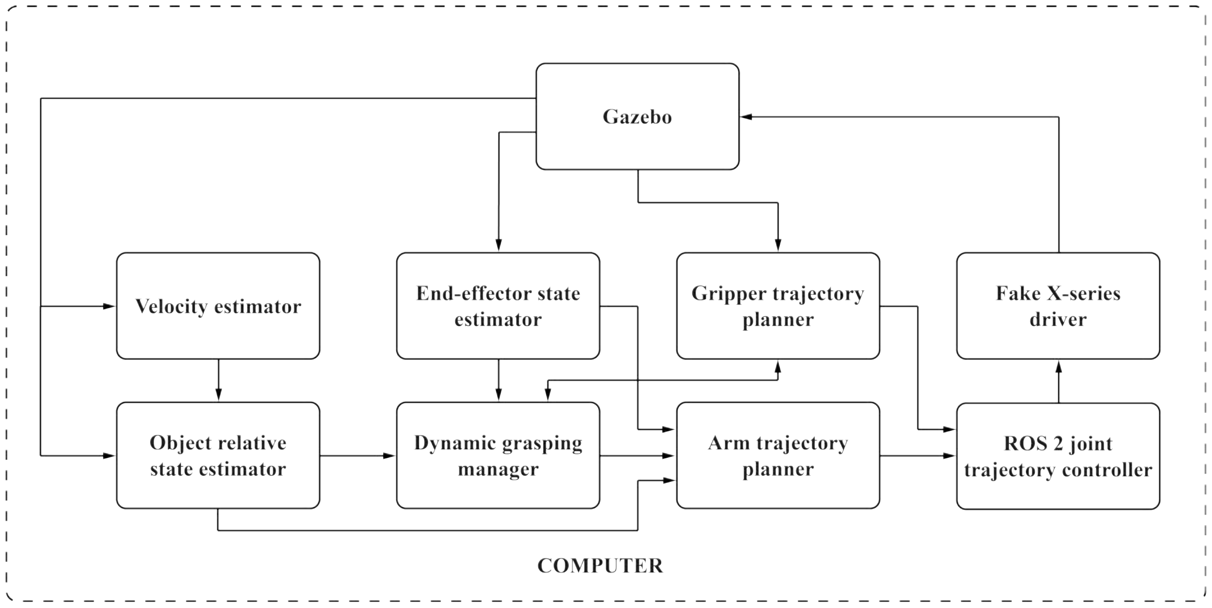

Gazebo
Velocity estimator
End-effector state estimator
Gripper trajectory planner
Fake X-series driver
Object relative state estimator
Dynamic grasping manager
Arm trajectory planner
ROS 2 joint trajectory controller
COMPUTER

Figure 10: Overall architecture of the simulated dynamic grasping system.

4.3.2 Simulation scenario for dynamic grasping

To evaluate the control pipeline under controlled yet challenging conditions, a representative simulation scenario was constructed within Gazebo. In this setup, the Interbotix WidowX 250 S robotic arm is fixed to the world frame, while the object to be grasped, a red cylinder, is placed atop a mobile cart, as shown in . This design choice, where the object is mobile and the robot is

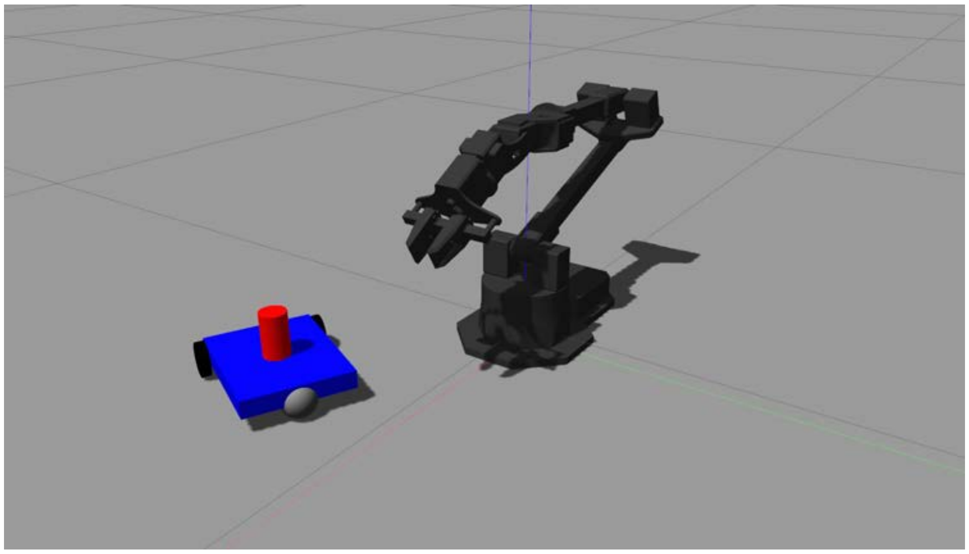


Figure 11: Simulation setup in Gazebo, showing all the rigid bodies involved in the exchange.

stationary, was made deliberately to allow for easier maneuverability and to validate the system under the demanding dynamic conditions. Importantly, this inversion of roles does not lead to issues when switching to the real hardware setup, where the robot is instead mounted on a mobile UGV. Since the control logic only relies on relative positions and velocities between the robot and the object, the simulation scenario remains fully representative of real-world use. Both in simulation and hardware experiments, the cart's (or UGV's) movement is manually commanded via a keyboard teleoperation node, offering maximum flexibility to test trajectories, velocity profiles, and disturbances.

The chosen object is a cylinder with the same physical characteristics, like dimensions and mass, as the one used in the hardware experiments. A cylindrical shape was selected because it ensures a stable base whether it is moving or not, and does not enforce a specific grasping orientation thanks to its axisymmetric nature. This simplification was intentional, as the focus of this work lies in the robustness of the control and motion coordination, rather than in addressing complex grasping strategies for irregularly shaped objects.

Notably, Gazebo streams joint states and object pose information at a frequency of 30 Hz. To remain synchronized with this simulation rate and avoid unnecessary computational overhead, the dynamic grasping manager node, in simulation, is configured to operate at the same frequency. This ensures that no data packets are missed and that the grasping logic always processes the most recent available state, preserving both responsiveness and stability.

4.3.3 Insights derived from simulation experiments

Running extensive dynamic grasping simulations within this framework yielded several valuable insights. First and foremost, they helped determine the approximate time durations required for each segment of the grasping trajectory. After the initial tests with approximate durations, these parameters were subsequently adjusted to achieve the quickest motion possible while ensuring maximum reliability and repeatability across a wide range of object velocities. These simulations also allowed to roughly estimate the upper bounds on the object velocity that the system could track reliably without compromising the success of the grasp.

From a methodological standpoint, the simulations suggested a series of refinements and improvements to the original control architecture. These included slight modifications in the way the finite-state machine handles premature grasp attempts and direction changes, as well as enhancements in the gripper timing logic based on empirical results. Furthermore, simulations provided a baseline for measuring computational performance: trajectory planning was consistently completed in approximately 0.5 ms, and full trajectory execution (from planning to actuation) required only a few milliseconds. These results confirmed the feasibility of deploying the system in real-time applications.

The practical effectiveness of the proposed approach was also confirmed visually. The sequence shown in Figure 12 demonstrates how the robot performs all the essential phases of the dynamic grasping pipeline: tracking, approaching, grasping, and retreating, all under dynamic conditions. This simulated behavior is directly comparable to the theoretical trajectory design discussed in Chapter 3 and illustrated in Figure 7. Both the simulated and theoretical representations share a consistent structure, with well-defined approach and grasp offsets (o_a and o_g), and segment transitions that preserve continuity in position and velocity. It is worth noting that, unlike in Figure 7b and Figure 7c, the end-effector orientation in this case appears flipped. This variation arises from the optimal orientation computed for the specific spatial configuration of the arm, cart, and object, as determined by the strategy outlined in Subsection 3.3.3.

Overall, this simulation setup provides a high-fidelity, fully reproducible environment that mirrors the kinematic and dynamic interactions of the real system. It enables exhaustive testing of the control strategy across a wide range of conditions while maintaining consistency with the hardware implementation pipeline.

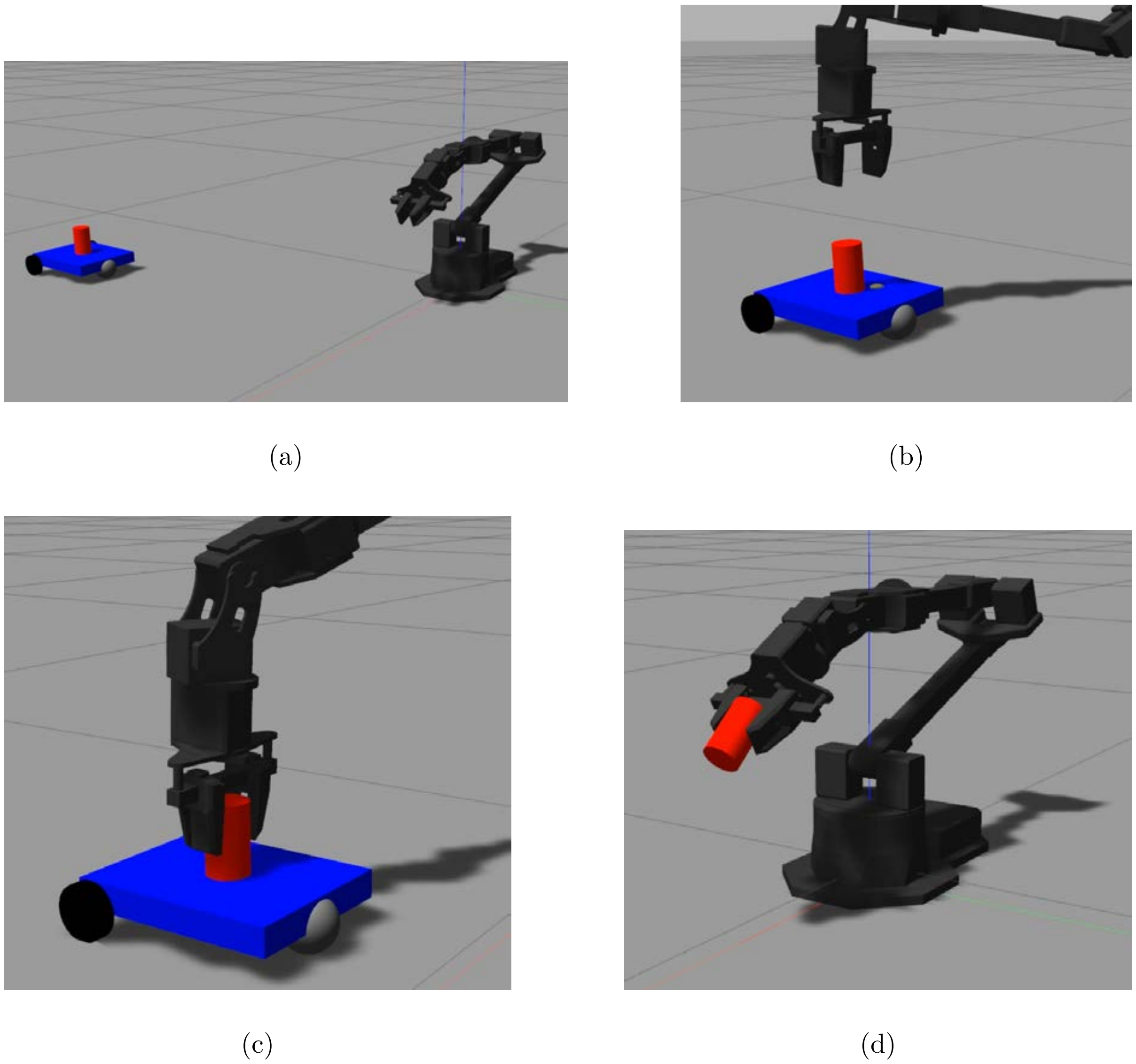


Figure 12: Main phases of the trajectory executed in simulation for dynamic object grasping.

CHAPTER 5

HARDWARE SET-UP AND EXPERIMENTS

This chapter describes the hardware configuration and experimental protocol used to assess the performance of the proposed dynamic grasping framework in a real-world setting. We begin by introducing the robotic and perception systems, then discuss their integration and calibration. Following that, we detail the experimental scenarios and the metrics employed for evaluation. Finally, we highlight key observations and insights gained through the hardware trials.

5.1 Hardware overview

5.1.1 Robotic arm

The primary manipulator in this study is the WidowX 250 S, a 6-DOF articulated arm with revolute joints, actuated by DYNAMIXEL servomotors. To ensure mechanical stability and prevent base tipping when the arm extends its links, it is rigidly mounted to a custom aluminum plate, as shown in Figure 13. This plate enlarges the contact area and shifts the resulting center of mass closer to the center of the robot base, maintaining system balance under full extension. MoCap markers are attached near the Waist joint on the base to enable accurate tracking of the robot's pose.

The end-effector comprises a two-finger gripper with 3D-printed narrow fingers, actuated via the arm's final motor stage. The gripper fingers translate along guide rails to execute grasping motions.

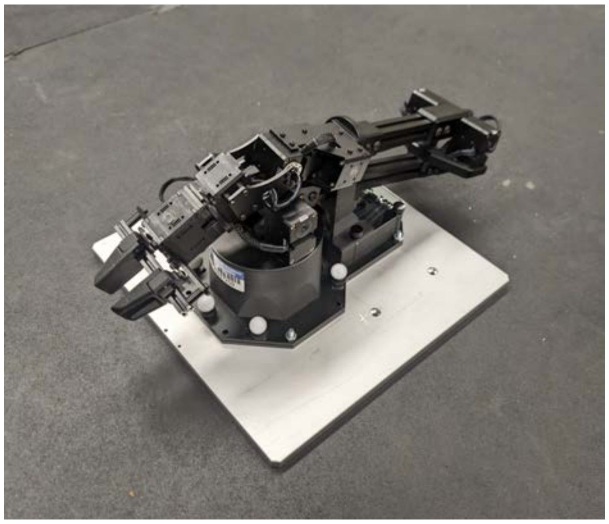
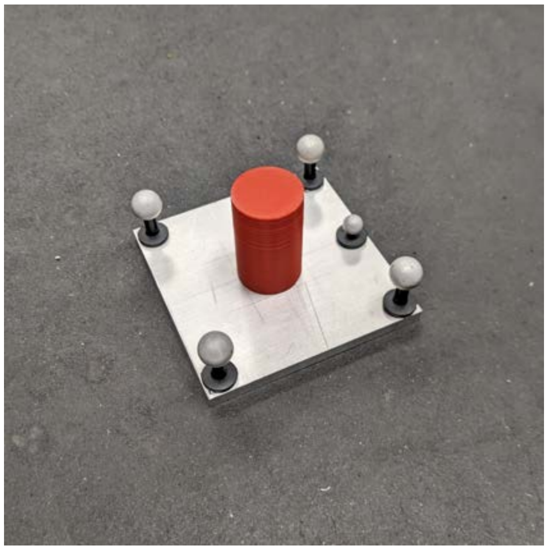


Figure 13: WidowX 250 S bolted to an aluminum base, equipped with motion-capture markers.

Internally, a DYNAMIXEL U2D2 interface board is mounted within the robot's central hub. This board provides USB-based communication between the offboard computer and the servomotors, allowing the control software to send joint commands and receive encoder feedback in real time.

5.1.2 Objects

The experiments employed simple geometric objects compatible with the WidowX gripper's limited width and simple design. Figure 14 shows representative examples: Two approaches were considered for marker placement. The first method, shown in Figure 14a, uses a cylindrical object mounted on a small marker-equipped plate. While this approach adds complexity, it guarantees reliable tracking, since MoCap requires markers to be spatially well separated. The


(a)

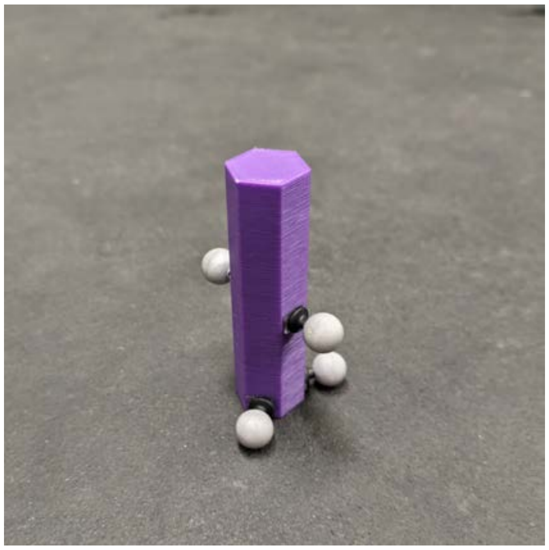

(b)

Figure 14: Sample test objects with motion-capture marker configurations.

second method (Figure 14b) places markers directly on the object. Although more similar to other works with different vision systems, where it is the object itself to be tracked at all times, this configuration may result in less reliable data from the motion capture system. In fact, the latter works best when the markers used to define a rigid body are not too close to one another. Moreover, placing them directly on the object to be grasped can lead to marker occlusion or collisions with the gripper. For the sake of completeness and performance testing, experiments were carried out using both approaches.

5.1.3 Mobile base (UGV)

To simulate relative motion scenarios during grasping, the WidowX manipulator is mounted on a Clearpath Husky A200 unmanned ground vehicle, as shown in Figure 15. The Husky is

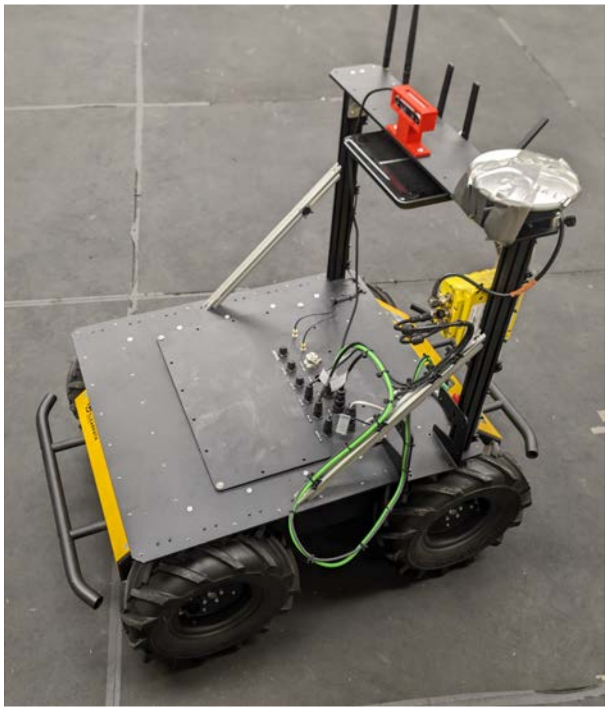


Figure 15: Clearpath Husky A200 UGV supporting the WidowX manipulator and computing platform.

well-suited for rapid prototyping in ROS 2 environments. It can be teleoperated via keyboard or accept velocity commands on ROS 2 topics, enabling the generation of controlled relative motion for dynamic grasp execution. Its deck provides a grid of threaded inserts, allowing the aluminum plate from Subsection 5.1.1 to be securely fastened. The aluminum mounting plate has been custom designed and machined to incorporate counterbore holes, allowing the screw heads to sit flush or slightly below the surface of the plate. These were measured and drilled manually to align precisely with the threaded inserts in the Husky's deck, ensuring that no screw heads protrude from either side. This flush mounting ensures a solid connection between the manipulator and the UGV.

Onboard the Husky are both the WidowX arm and the computer responsible for all real-time grasp planning and perception tasks. Once fully assembled, the integrated system, which includes the manipulator, UGV, and offboard computer, is ready to perform grasping of objects while moving, independently tracking the targets, planning and executing trajectories on the fly as the UGV is teleoperated. The complete setup is shown in Figure 16.

5.1.4 Motion capture system

As previously introduced, the vision system adopted for tracking the positions of both the robotic arm and the target object is a motion capture system developed by Optitrack. Specifically, the experimental environment is equipped with twelve Prime^{x} 13 motion capture cameras (Figure 17), which provide millimeter-level accuracy across a tracking area of approximately 36 m^2, streaming data at a native frame rate of 240 Hz [30]. These cameras operate by detecting only specially designed reflective markers, as illustrated in Figure 18.

OptiTrack's system is managed through its proprietary software, Motive, which enables the monitoring and configuration of all tracked elements, as well as access to various data

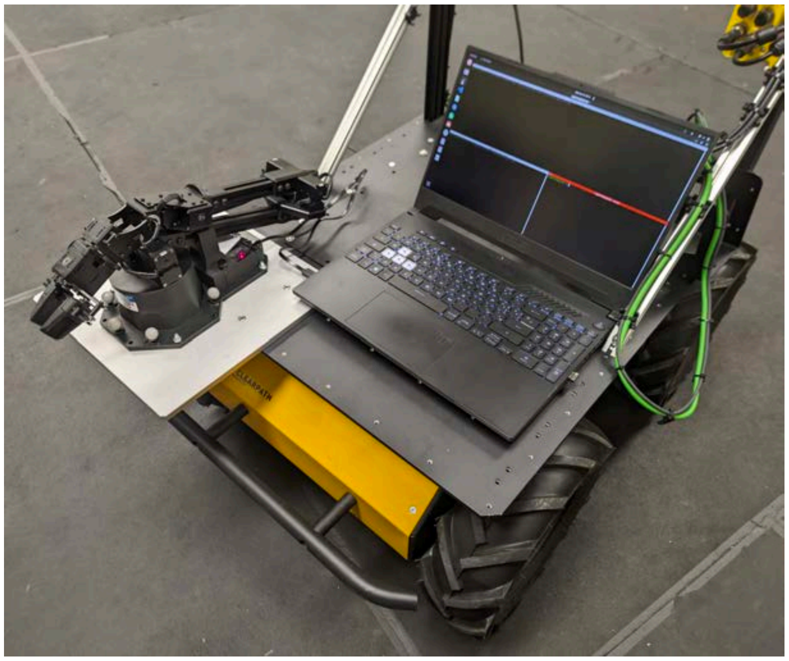


Figure 16: Integrated loco-manipulation system: WidowX 250 S, Husky A200, and offboard computer.

streaming options. Due to the high precision of the system and the collaborative nature of the camera array, any change in the position or orientation of the cameras can drastically degrade tracking accuracy and performance. Therefore, whenever the system is resumed after a period of inactivity, a full calibration is strongly recommended. This process should be carried out following OptiTrack's official guidelines and using the dedicated calibration tools. After completing this procedure, the system used in this work reported a root-mean-square error (RMSE) of approximately 0.6 *mm*. Following calibration, rigid bodies were created within

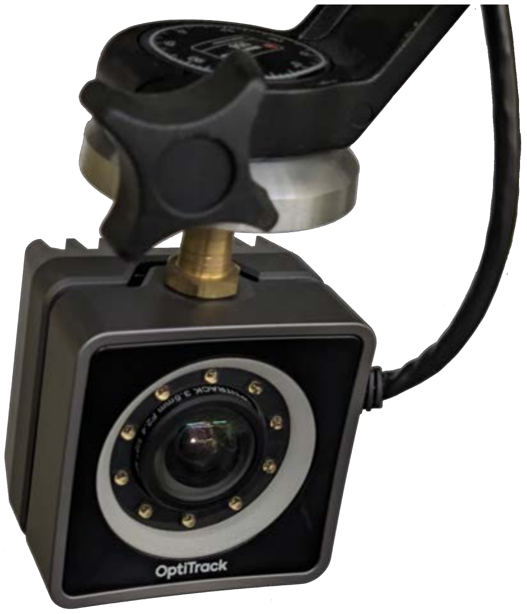


Figure 17: OptiTrack's Prime^x 13 motion capture camera.

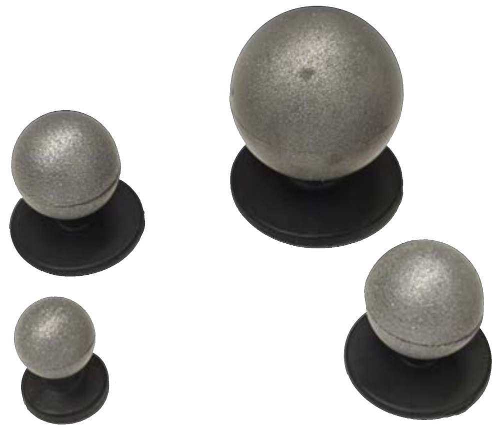


Figure 18: Motion capture markers.

the tracked environment by selecting and grouping the corresponding sets of markers. It is important to note that the reference frame initially assigned to a rigid body upon its creation does not necessarily coincide with the one initially designed during the kinematic analysis. This is particularly critical for elements such as the base of the WidowX arm. In such cases, it is necessary to manually reposition the center of mass and adjust the frame orientation to match the intended reference system. This refinement was performed using Motive's built-in tools, and the final configuration of the tracked rigid bodies is shown in Figure 19.

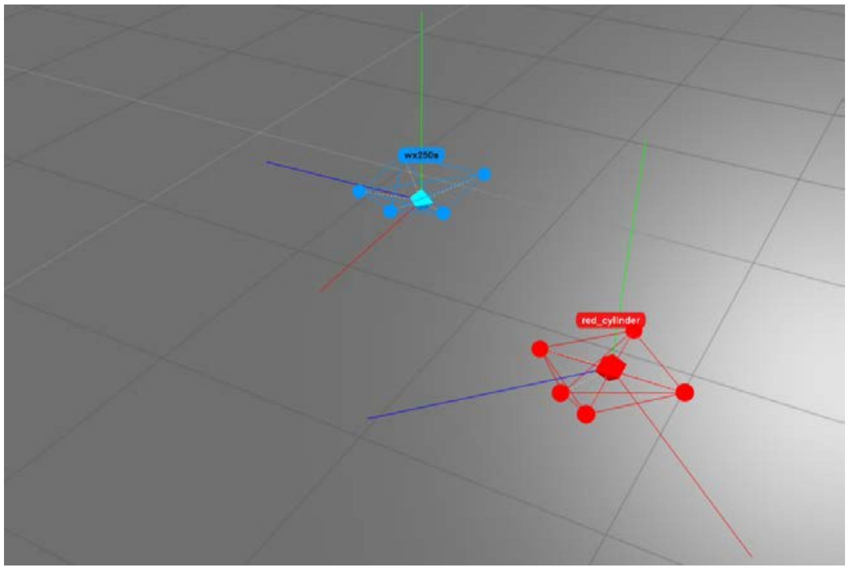


Figure 19: Rigid bodies in Motive software.

Once all components are properly configured, the Motive software is capable of streaming accurate position and orientation data for each rigid body, labeled according to the names defined within the software interface, via the NatNet 4.0 protocol to the offboard computer.

To provide a comprehensive understanding of the hardware setup and its interconnections, Figure 20 illustrates the overall architecture adopted for dynamic grasping. The diagram summarizes the main components described in this section, including the motion capture system, the robotic manipulator mounted on the mobile platform, and the computation and control modules. It highlights the flow of data and control signals, from sensing and state estimation to motion planning and actuation, showcasing how each element contributes to the execution of the grasping task.

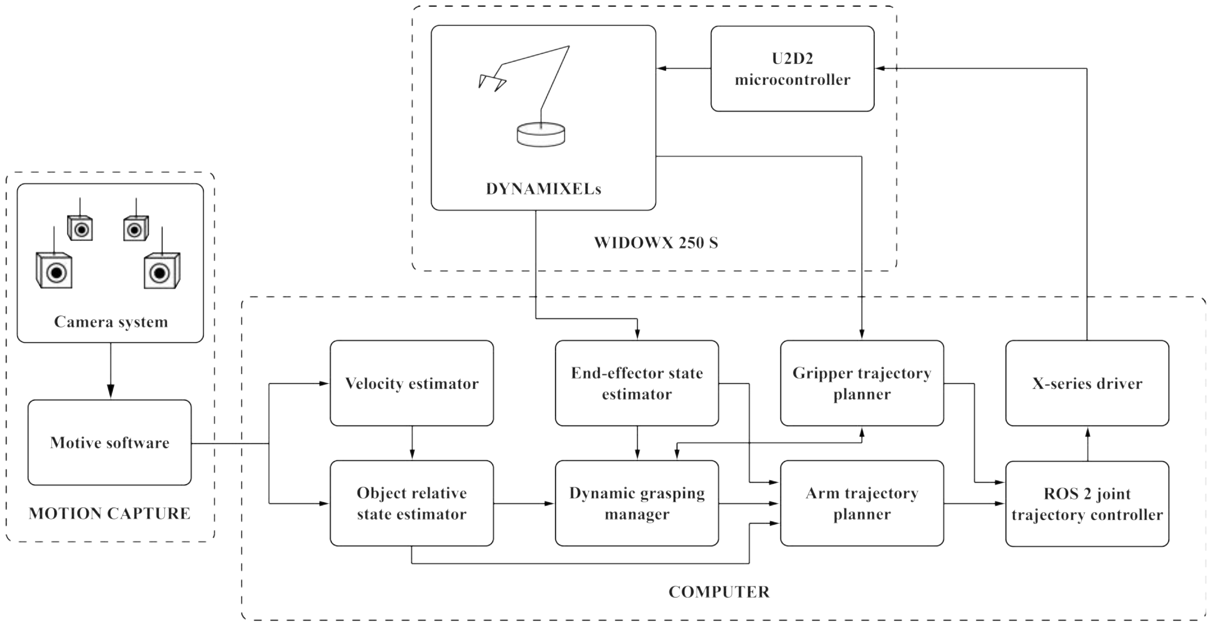


Figure 20: Overview of the complete hardware and software architecture for dynamic grasping.

5.2 DYNAMIXEL Position Controller PID Tuning

Each DYNAMIXEL actuator of the robotic arm (e.g., XM430-W350, XL430-W250) contains an onboard position controller incorporating PID control and feed-forward terms. Figure 21 depicts the internal controller pipeline of XM-series and XL-series actuators: Any instruction received from the DYNAMIXEL bus is stored as the Goal Position, which is then converted into the desired position and velocity trajectories using the specified Velocity Profile and Acceleration Profile, both of which are adjustable parameters. Standard PID feedback is then applied to the error $e = q_d - q$ (where q_d is the goal joint position and q is the current measured position), with

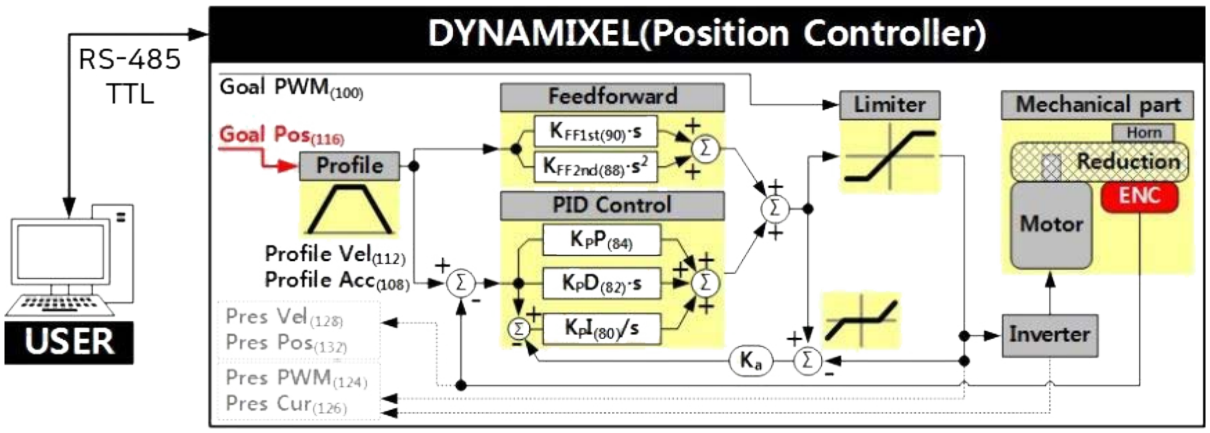


Figure 21: Block diagram of the DYNAMIXEL position controller including internal PID and feedforward (velocity and acceleration) controllers, anti-windup, and output limiter [1][2].

proportional (K_P), integral (K_I), and derivative gains (K_D). It is also possible to introduce preemptive compensation by adding feedforward terms:

$$u_{\text{FF}} = K_{FF1} \cdot \dot{q}_d + K_{FF2} \cdot \ddot{q}_d$$

where K_{FF1} and K_{FF2} scale desired velocity and acceleration, thus reducing tracking error due to dynamics. By computing a baseline command from known reference signals, feedforward terms reduce the burden on the feedback (PID) loop, sometimes resulting in faster settling times, reduced overshoot, and improved tracking accuracy. An anti-windup gain (K_a), that is not modifiable by the user, limits the integral term to prevent accumulation outside actuator capabilities. The combined PWM output is saturated when needed before being issued to the motor. Finally, the correct PWM output is sent to the motor, and the current joint position and velocity are obtained through an encoder and stored for the next cycle.

It is relevant to notice that, when performing tuning using the DYNAMIXEL Wizard 2.0 software, all the controller gains that the user can modify follow a dedicated range of values, which are scaled versions of physical ones. The scaling factor is, however, not the same for all the gains. Table IV summarizes all these details and applies to both types of servomotors in WidowX. Note that in Table IV, the symbols K'_P, K'_I, K'_D, K'_{FF1}, K'_{FF2} denote the raw,

TABLE IV: DYNAMIXEL POSITION CONTROLLER GAINS, CONVERSION EQUATIONS, AND RANGES [1][2].

Gain	**Address**	**Conversion Equation**	**Range**
Position D gain	80	$K'_D = 16\ K_D$	0–16383
Position I gain	82	$K'_I = 65536\ K_I$	0–16383
Position P gain	84	$K'_P = 128\ K_P$	0–16383
Feedforward 1st gain (velocity)	90	$K'_{FF1} = 4\ K_{FF1}$	0–16383
Feedforward 2nd gain (acceleration)	88	$K'_{FF2} = 4\ K_{FF2}$	0–16383

non-physical gain values that must be assigned to the corresponding control table addresses within the actuator firmware.

All the DYNAMIXELs of WidowX come with factory settings. The XM430-W350 servomotors have only a default proportional gain, while the XM430-W350 servomotors have both proportional and derivative gains assigned. This was not optimal, and after a few tests, even in stationary conditions, the robot arm clearly showed poor accuracy, suggesting that tuning was needed.

We performed gain tuning using the official DYNAMIXEL Wizard 2.0 software by ROBOTIS, a very versatile GUI tool that allows users to isolate individual motors, modify maximum velocity and acceleration profile values, and adjust PID and feedforward parameters in real time, observing the actuator's responses on live graphs during step tests. The adopted manual tuning process involved:

- Torquing on all the motors to make the whole kinematic chain solid.
- Enabling single-motor control via Wizard.
- Applying step position commands and observing the response curves.
- Iteratively adjusting the proportional (K_P), integral (K_I), derivative (K_D), and, if necessary, feedforward gains (K_{FF1}, K_{FF2}) too.
- Aiming for fast yet stable joint responses, without excessive overshoot or steady-state error.

The results are illustrated in Table V. In Table V, the second feedforward gain (acceleration

TABLE V: FIRMWARE AND PHYSICAL CONTROL-LOOP GAINS FOR EACH JOINT OF WIDOWX 250 S.

Joint	**Firmware gains ($\boldsymbol{K'}$)**				**Physical gains ($\boldsymbol{K}$)**			
	$\boldsymbol{K'_P}$	$\boldsymbol{K'_I}$	$\boldsymbol{K'_D}$	$\boldsymbol{K'_{FF1}}$	$\boldsymbol{K_P}$	$\boldsymbol{K_I}$	$\boldsymbol{K_D}$	$\boldsymbol{K_{FF1}}$
Waist	1000	10	0	0	7.8	$1.5{\cdot}10^{-4}$	0.0	0.0
Shoulder	2800	3500	8700	700	21.9	$5.3{\cdot}10^{-2}$	543.8	175.0
Elbow	2000	3200	6000	600	15.6	$4.9{\cdot}10^{-2}$	375.0	150.0
Forearm roll	800	16	0	0	6.2	$2.4{\cdot}10^{-4}$	0.0	0.0
Wrist angle	1200	20	0	0	9.4	$3.1{\cdot}10^{-4}$	0.0	0.0
Wrist	640	0	3600	0	5.0	0.0	225.0	0.0
Gripper	640	0	3600	0	5.0	0.0	225.0	0.0

term) is omitted because none of the joints required it. As expected, the Shoulder and Elbow joints, responsible for carrying and positioning the entire kinematic chain, demanded the most nuanced tuning. Including a velocity feedforward term ensures control effort is proportional to speed, enhancing the arm's ability to track its desired trajectory through space. The integral gain also plays a crucial role: tiny angular errors at these joints translate into significant end-effector displacement, so a well-tuned K_I swiftly eliminates steady-state errors for precision positioning. Finally, the derivative gain is employed to dampen any oscillatory behavior introduced by higher P and I gains, helping the system converge smoothly.

Manual tuning was chosen because robotic manipulators exhibit strongly configuration-dependent dynamics; in fact, variations in inertia, gravity loading, friction, and coupling occur every time the arm moves. As a result, a set of gains that works in one pose may perform poorly in another. Although model-based or analytical tuning methods can achieve higher theoretical precision, they require accurate dynamic models and often involve significant online computation. In contrast, manual (trial and error) tuning is model-free, relies solely on observed behavior, and allows for rapid, pose-specific adjustment. To achieve optimal performance in the dynamic grasping tasks of this study, each joint's PID calibration was conducted arranging the other joints to configurations that mimic those achieved in such tasks and working around that point, one joint at a time. It is, in fact, better to ensure precision in configurations achieved at the end of grasping trajectories, where the correct positioning and orientation of the end-effector are crucial.

5.3 Experiment protocols and evaluation metrics

Experiments were conducted in a controlled laboratory environment with a floor covered in rubber tiles to minimize mechanical vibrations. To evaluate the performance of the proposed dynamic grasping algorithm under various conditions, the relative placement of the robotic arm, UGV, and object was intentionally varied between trials. Consistent positioning was preserved only during tests designed to study performance degradation over a range of relative speeds, so as to isolate the effect of velocity while minimizing other sources of variability. This approach ensured a fair and objective assessment of the sensitivity of the system to changes in motion dynamics.

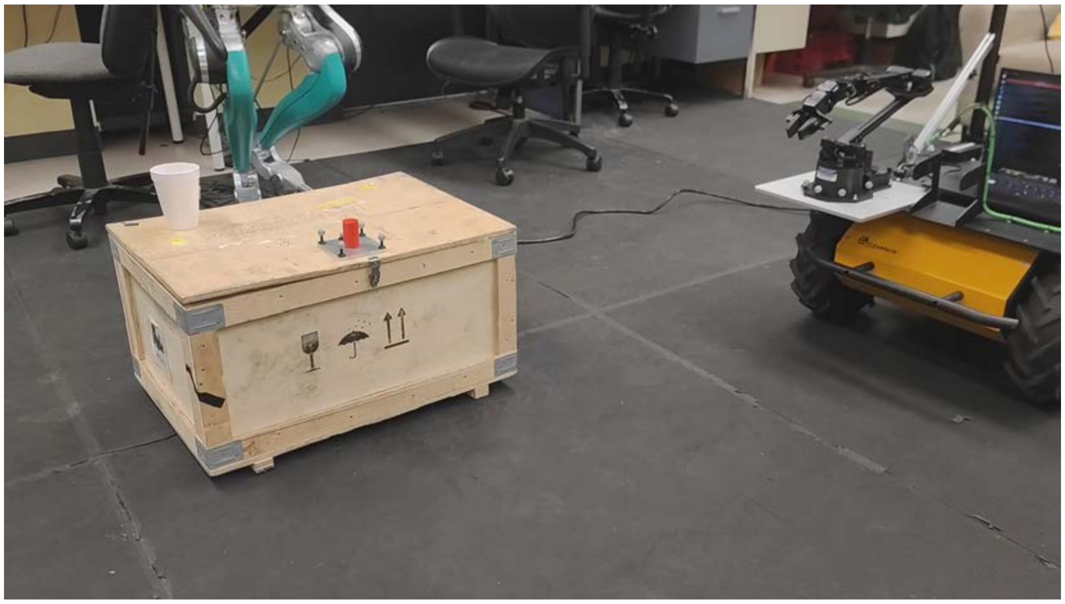
The following subsections detail the rationale behind each experimental scenario, the methodology adopted, and the types of data collected to assess system performance.

5.3.1 Testing at different speeds

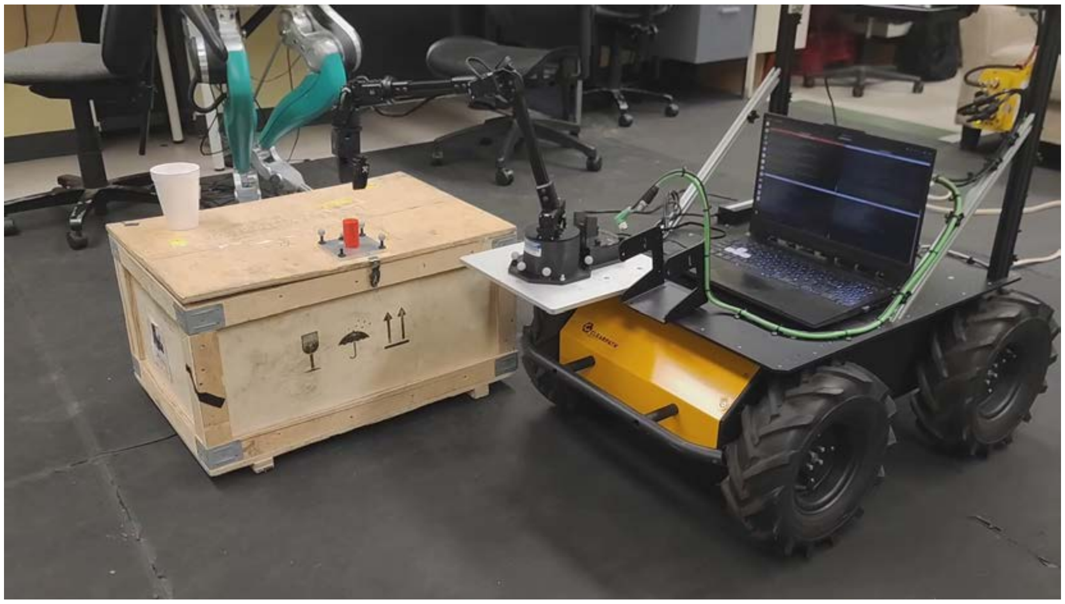
To examine how performance deteriorates with increasing relative speed between the object and the end-effector, a series of trials was performed at different constant velocities. These trials shared an identical spatial setup: from the perspective of the robot's base frame, the object consistently entered the reachable workspace from the same location and followed a fixed direction. This consistency ensured that the primary variable affecting performance was the relative velocity, which directly influenced the time the object remained within the reachable workspace ($\Delta\tau$ in Subsection 3.6.1).

For all runs, the robot arm was initialized from a position sufficiently far from the object to allow the grasp sequence to begin immediately once the conditions described in Subsection 3.6.1 were met. This maximized the time available for WidowX to perform the grasp, thereby improving the likelihood of success.

Figure 22 presents three snapshots from a representative trial, depicting a successful grasp as the system follows a linear trajectory alongside the object. It can be seen that the wooden box holding the target object was consistently positioned along the lateral path of the moving Husky–WidowX system. Thus, from the manipulator's base frame, the object always approached from the same side. Figure 22a shows the moment just before the robot initiates its

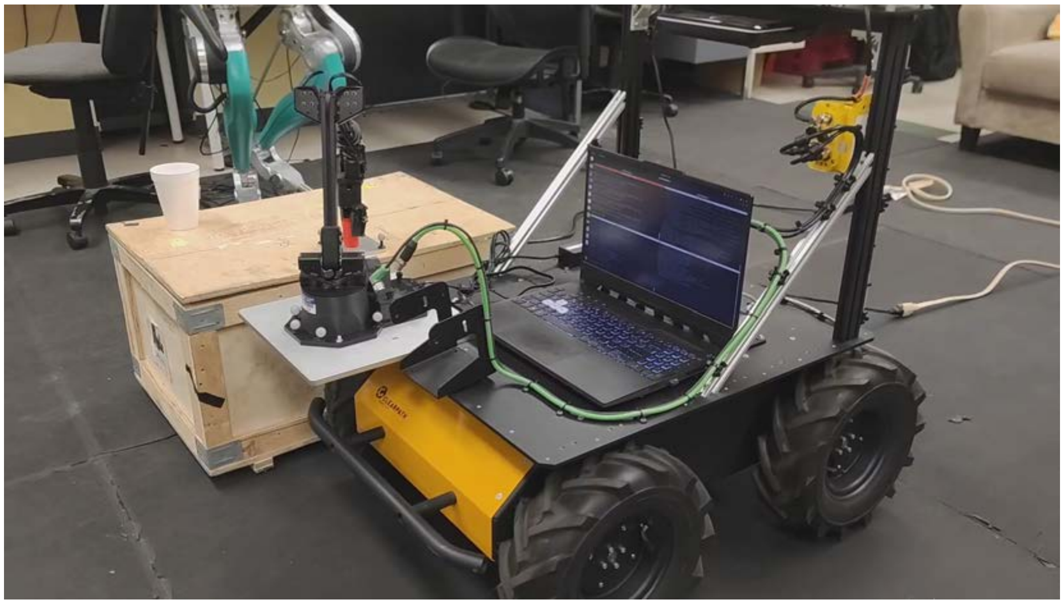


(a)



(b)

Figure 22: Sequence illustrating a successful grasp as the Husky–WidowX system moves linearly at 25 cm/s beside the target object.



(c)

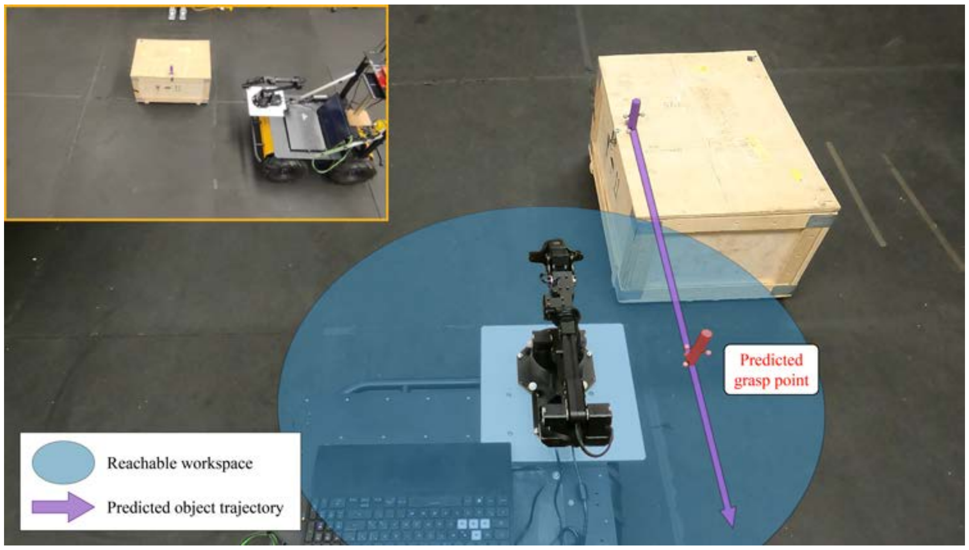
Figure 22: Sequence illustrating a successful grasp as the Husky–WidowX system moves linearly at 25 cm/s beside the target object.

motion toward the object (similar to the configurations shown in Figure 7a and Figure 12a). A few instants later, as shown in Figure 22b, the end-effector manages to get over the object, at the predefined approach distance $o_a = 10\ cm$, a configuration also visible in Figure 7b and the simulation frame in Figure 12b. Finally, Figure 22c captures the exact moment of grasp, when the gripper closes around the object at the grasp offset $o_g = 1\ cm$. As discussed earlier, the grasp offset represents the distance between the object's center of mass and the tool center point of the end-effector, and must be adjusted according to the object's size and shape.

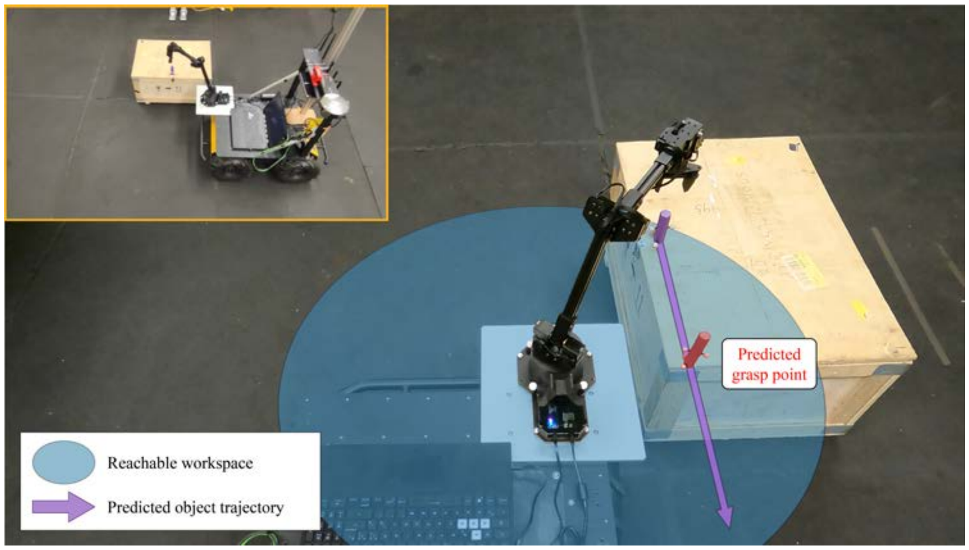
The same type of test, where the relative velocity between the robot's base and the object to be captured remains constant, can also be viewed from the perspective of the mobile base. This is illustrated in Figure 23, which highlights the critical steps involved in capturing the object as shown in Figure 22, but from a different viewpoint. These images also depict the reachable workspace of the robot arm for an object that, from the point of view of the robot arm, travels on that imaginary plane parallel to the xy-plane of the manipulator's base frame, and a portion of predicted object trajectory at that instant, in order to better highlight the space that object had to travel before entering the workspace (Figure 23a) and, when already inside the workspace, the space that it has left inside it (Figure 23b); these are in fact critical details that are needed to establish if to start the grasp sequence or to continue it after a change in the object's relative velocity is detected.

The set of speed values tested ranged from 5 cm/s to 35 cm/s in increments of 5 cm/s. During each trial, joint states, end-effector poses, and object positions were recorded with timestamps from ROS 2 topics using a custom data logger. Both the planned and actual trajectories were captured, preserving the full temporal progression of each event.

This data was used to generate plots comparing the planned and the executed joint and end-effector trajectories, as well as depicting the spatial paths of the object and end-effector from the start of the grasp sequence to the moment of grasp. Additionally, the relative distance between the two over time was analyzed, all referenced to the base frame of the manipulator.

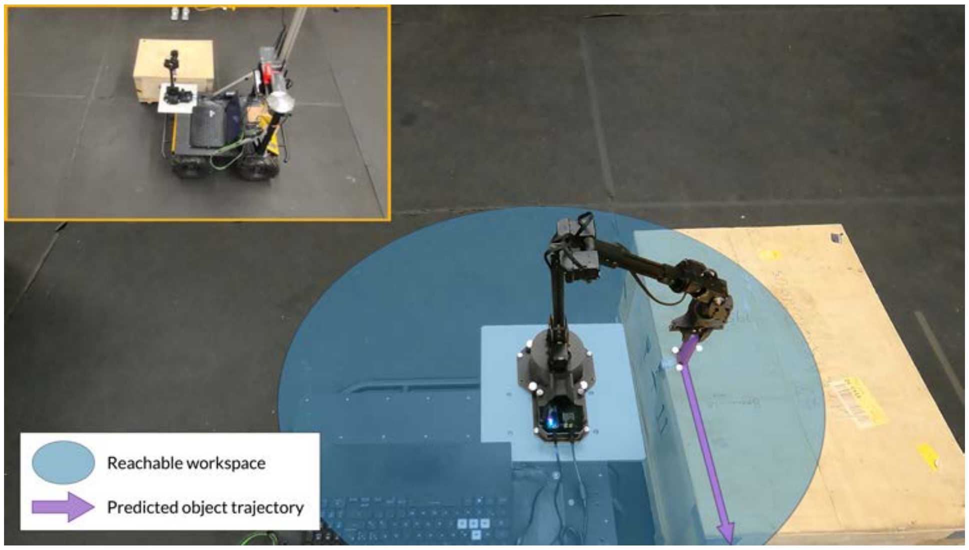


(a)



(b)

Figure 23: Sequence illustrating a successful grasp at 18 cm/s from the point of view of the mobile system.



(c)

Figure 23: Sequence illustrating a successful grasp at 18 cm/s from the point of view of the mobile system.

5.3.2 Testing variations in velocity magnitude

Given that the proposed control algorithm was designed to function in dynamic and changing environments, further testing was conducted to assess its robustness against variations in the magnitude of the object's relative velocity, including sudden stops. These tests reused the same spatial setup described in Subsection 5.3.1, with one key distinction: during each run, the Husky initially moved at a constant velocity that was deliberately altered mid-trial, either by slowing down or accelerating, after the arm began its motion. This destabilized the originally planned trajectory and triggered real-time replanning by the control system.

Figure 24 depicts the final stage of one such trial, where the red cylinder is successfully grasped after the system detected that the object was no longer moving relative to the base frame. The data collected in these trials mirrored that of the fixed-speed experiments. In

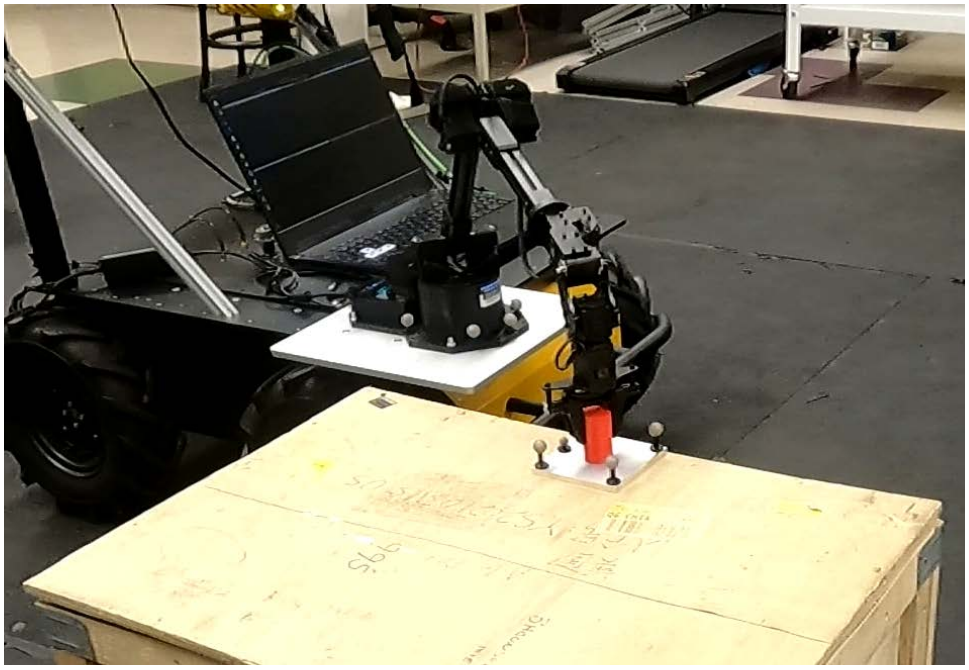


Figure 24: WidowX grasping a stationary red cylinder after a sudden stop of Husky.

addition to visual confirmation of grasp success, the analysis focused on the 3D trajectories of the object and end-effector over time and the time evolution of their relative distance in space.

5.3.3 Testing changes in direction

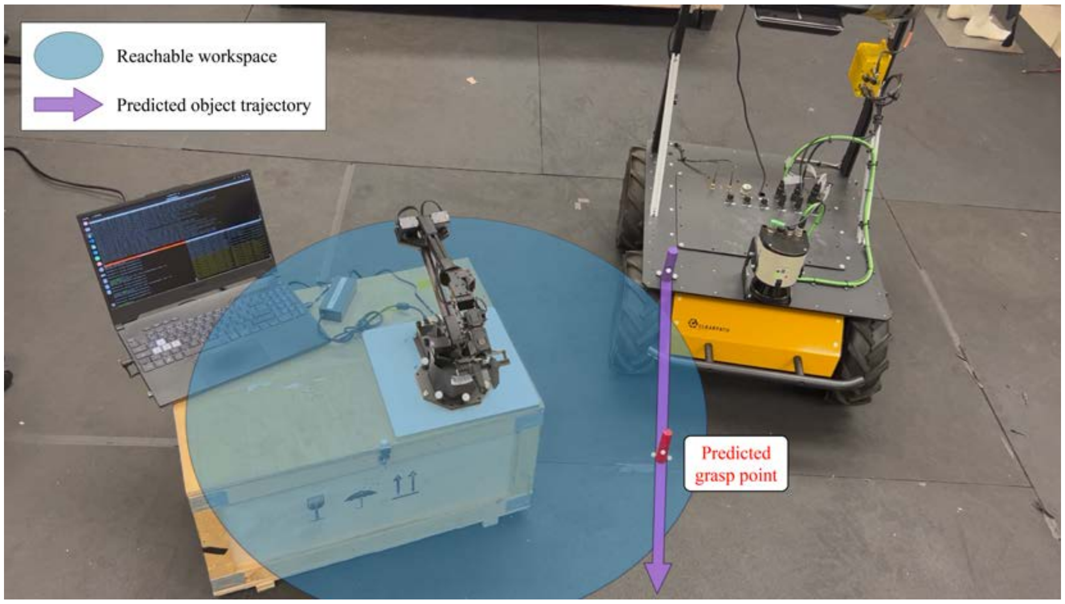
The final experimental condition addressed the system's response to changes in the direction of the object's motion. Although, from a control perspective, this scenario is treated similarly to the one involving velocity magnitude variations, since the algorithm tracks the velocity vector and only initiates grasping when linear motion is stable, a direction change imposes an additional requirement: the orientation of the planned trajectory must be updated. Because the object is no longer moving along the same line, this adjustment typically results in a longer time to successfully complete the grasp.

To evaluate the system under these circumstances, tests were conducted in two configurations: one with WidowX mounted on the moving Husky while the object remained stationary, and another where the roles were reversed: WidowX was placed on a box and the object was placed on the Husky. This second configuration was essential because Husky can only maintain a constant linear velocity along the longitudinal axis, and given the orientation of the robot arm on its platform, any direction change would still result in the object moving in a similar straight path relative to the arm, just at a shifted lateral position but with the same direction. Therefore, to genuinely test direction changes, the object had to move with respect to the base frame in varying directions, which was achieved by mounting it on Husky while keeping the robot arm fixed.

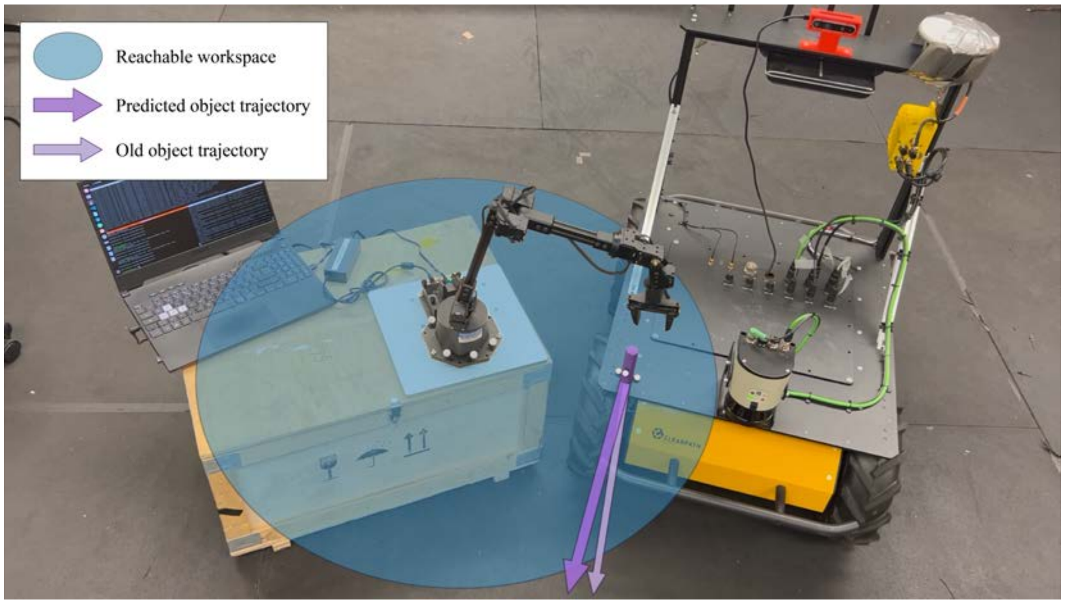
Figure 25 illustrates a test where Husky changed direction three times while moving at a constant speed of approximately 10 cm/s, with the object on board. Like in Figure 23, these images show the reachable workspace of the robot arm and a portion of the predicted object trajectory at the represented time instant. In this situation, after a change of direction, the length of the portion of reachable workspace that the object has to travel and the time in which it is predicted to travel it are paramount in determining whether to continue the grasp or to stop the task.

Figure 25a captures an instant before the object enters the workspace, with the robot arm still at rest. After a change in direction occurs, the control system detects that the velocity of the target received during the current cycle of the dynamic grasping manager no longer matches the previously known velocity. This discrepancy means that the predicted ”old” and ”new” trajectories differ, as illustrated in Figure 25b. Consequently, the dynamic grasping manager sends a command to the motion execution pipeline to stop the arm. Figure 25c instead illustrates the end-effector pose and the object position right after the last change in direction. It is clear that there was a much smaller portion of space inside the reachable workspace available and consequently also the time constraint became tighter, as the object would have soon exited the reachable workspace. However, thanks to the fact that end-effector had already started the capture sequence and got close to the object, it had time to start right after and manage to grasp the object right before it got out of its reachable workspace, as shown in Figure 25d.

For these tests, the data collected and the validation criteria are the same explained and adopted for the tests described in Subsection 5.3.1.

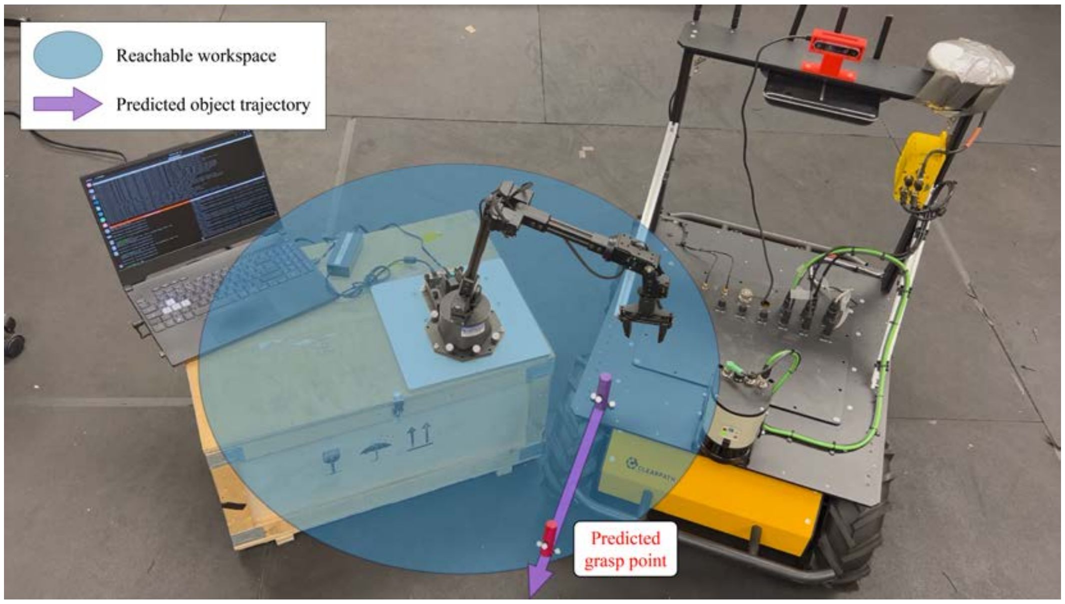


(a)

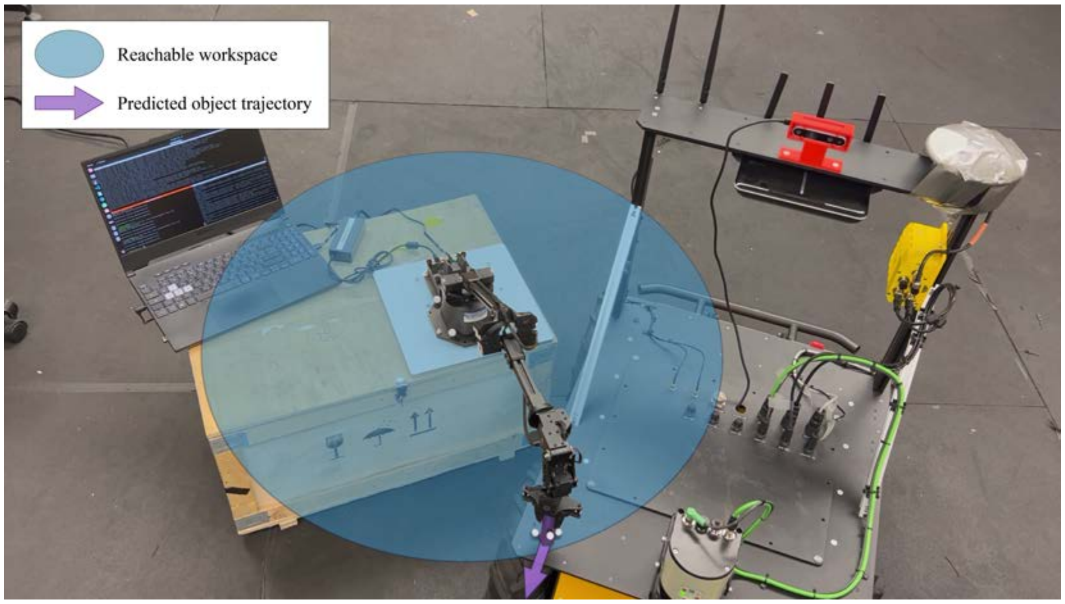


(b)

Figure 25: Sequence illustrating a successful grasp after direction changes of the object.



(c)



(d)

Figure 25: Sequence illustrating a successful grasp after direction changes of the object.

CHAPTER 6

EXPERIMENTAL RESULTS

In this chapter we present the results of the experiments and tests carried out to evaluate the performance of the proposed dynamic grasping algorithm and control system.

6.1 Computational performance of the dynamic grasping control system

The experimental results provide comprehensive insights into both the robot system's data retrieval capabilities and the computational performance of the implemented algorithms. In hardware experiments, the dynamic grasping manager node successfully handles and updates the entire system state at a frequency of 50 Hz. The trajectory planner emerges as the highest-performing component, generating complete Cartesian trajectories from start to finish in approximately 100 μs and converting all Cartesian position and velocity waypoints to joint space waypoints in under 300 μs. The total processing time, from the moment the trajectory generation command is issued until the planner forwards the trajectory to the position controller, averages approximately 500 μs and consistently remains below 1 ms. This exceptional performance is achieved through Ruckig's capability to generate time-optimal trajectories within extremely short timeframes, combined with the analytical inverse kinematic function that enables rapid Cartesian-to-joint space conversions in a few microseconds.

6.2 Performance and accuracy across different relative velocities

Let us now move to the actual outcomes of experiments. The following figures correspond to a test conducted using the basic setup described in Subsection 5.3.1, where the relative speed between the object and the end-effector was approximately $25, cm/s$. They compare the ideal and actual trajectories of the robot arm in terms of joint positions, joint velocities, and end-effector pose. These results are representative of the range of relative velocities for which the dynamic grasping algorithm has proven to be successful. In each figure, the red arrow marks the first moment when the gripper is fully closed and the grasp sequence is considered complete. This time also defines the final instant shown in all time-based plots.

Figure 26 is clearly the one in which the actual trajectory best follows the ideal one. This is a direct consequence of the fact the joint configuration is part of the closed control loop of the motors' PID controllers, which therefore try to correct the error between the measured joint positions and the target ones at each step. As a result, accurate tracking at the joint level, particularly towards the end of the trajectory, leads to relatively precise end-effector motion in Cartesian space. It is worth noting, however, that Cartesian motion is not directly regulated in closed loop, but rather arises from the forward kinematics of the joint configuration, and is therefore effectively controlled in open loop. For what concerns the position components of the pose, it is evident that the most critical trajectory to follow is the one along the z direction. This can be explained by the presence of gravity, which from the point of view of the position controller is considered a pure (unknown) disturbance that cannot be avoided. Among the Cartesian position components, the most challenging trajectory to follow appears to be along the z-axis. This discrepancy can be attributed to the influence of gravity, which acts as an unmodeled disturbance from the perspective of the position controllers and is not explicitly compensated for during motion execution.

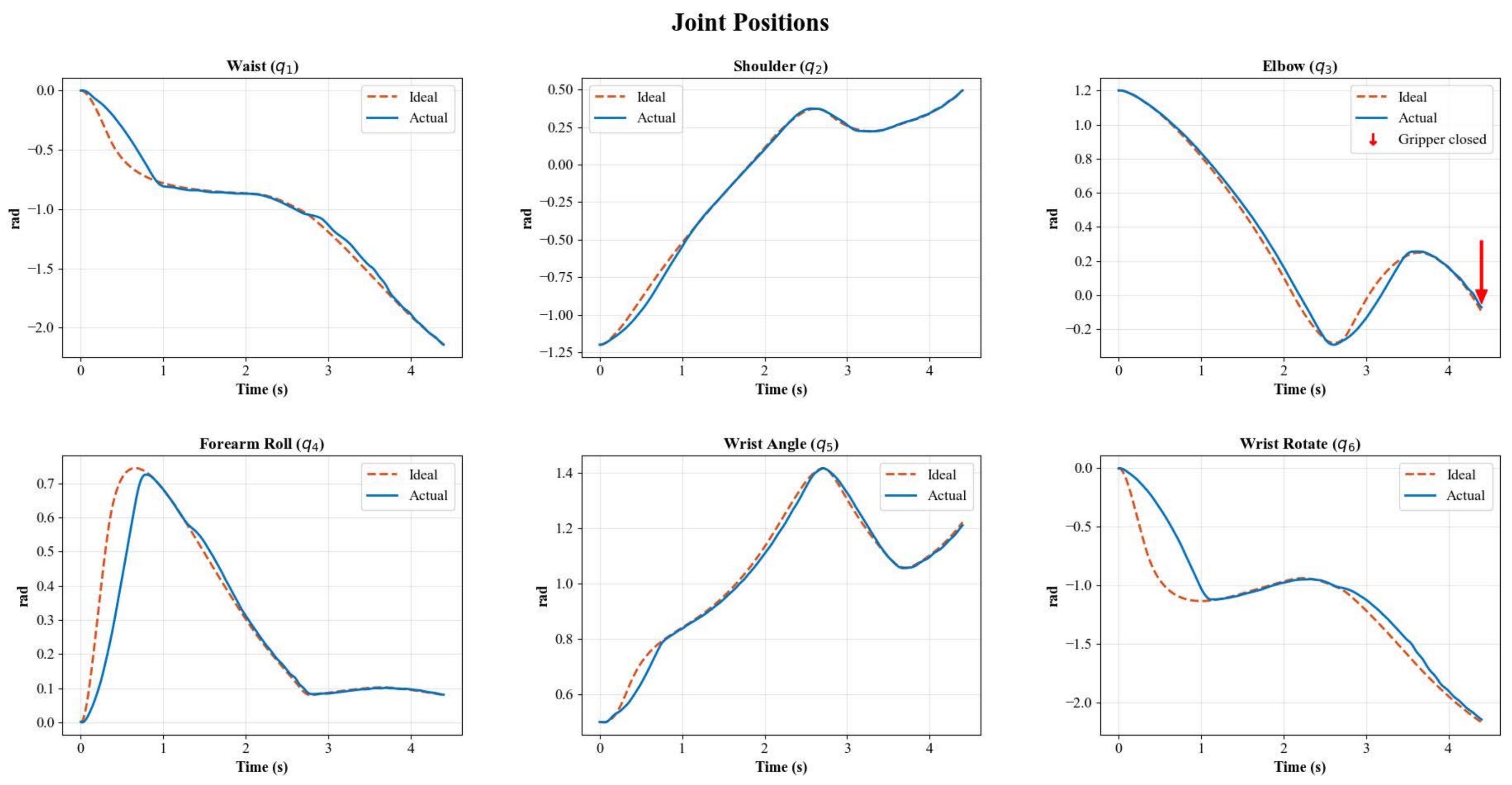


Figure 26: Ideal (dashed) vs. actual (solid) joint position trajectories for the six arm joints during dynamic grasping at a relative object speed of $0.25\,m/s$. The joint-space RMSE is $0.254\,rad$.

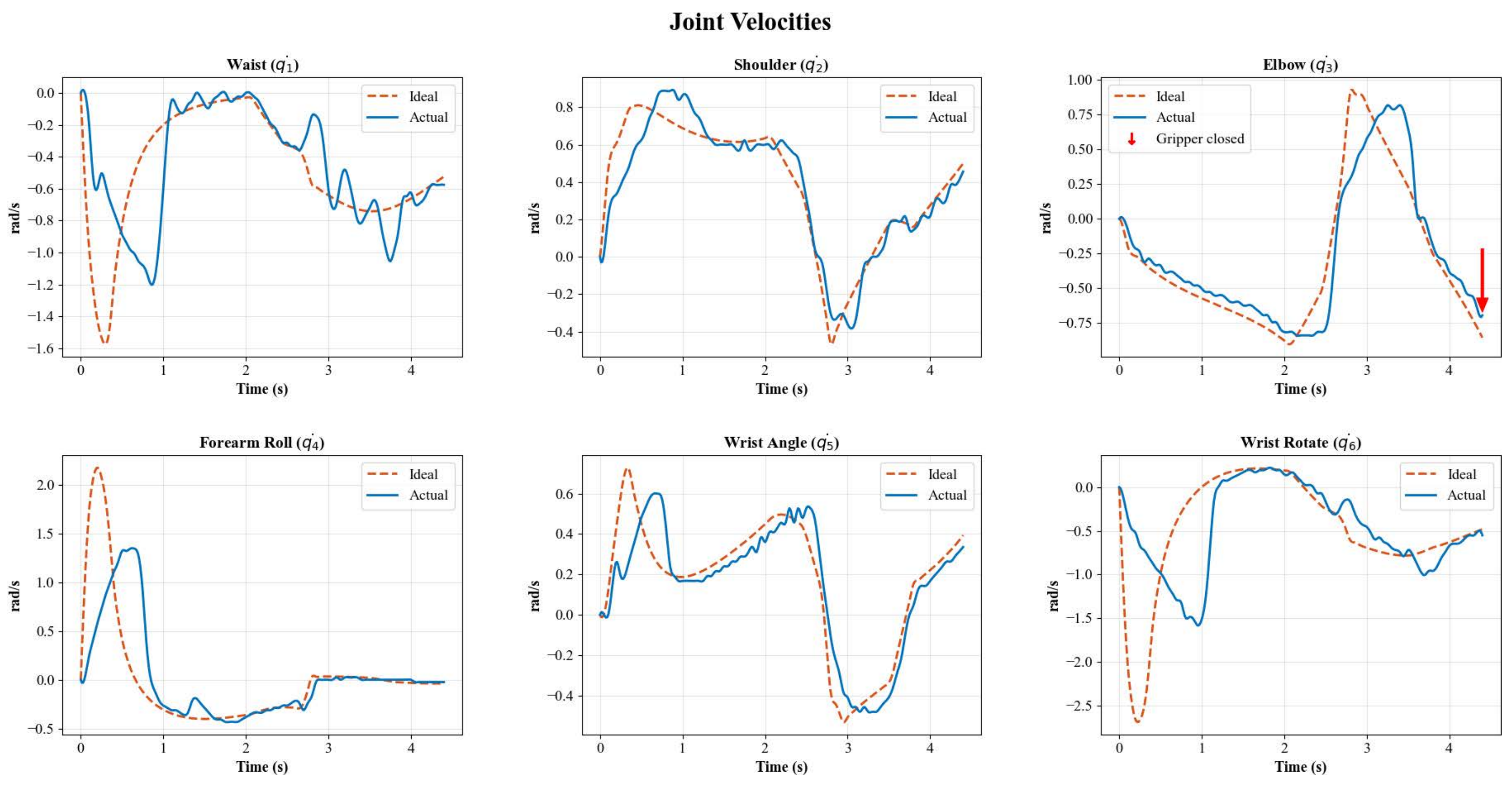


Figure 27: Ideal (dashed) vs. actual (solid) joint velocity trajectories for the six arm joints during dynamic grasping at a relative object speed of 0.25 m/s. The joint-space velocity RMSE is 0.9285 rad.

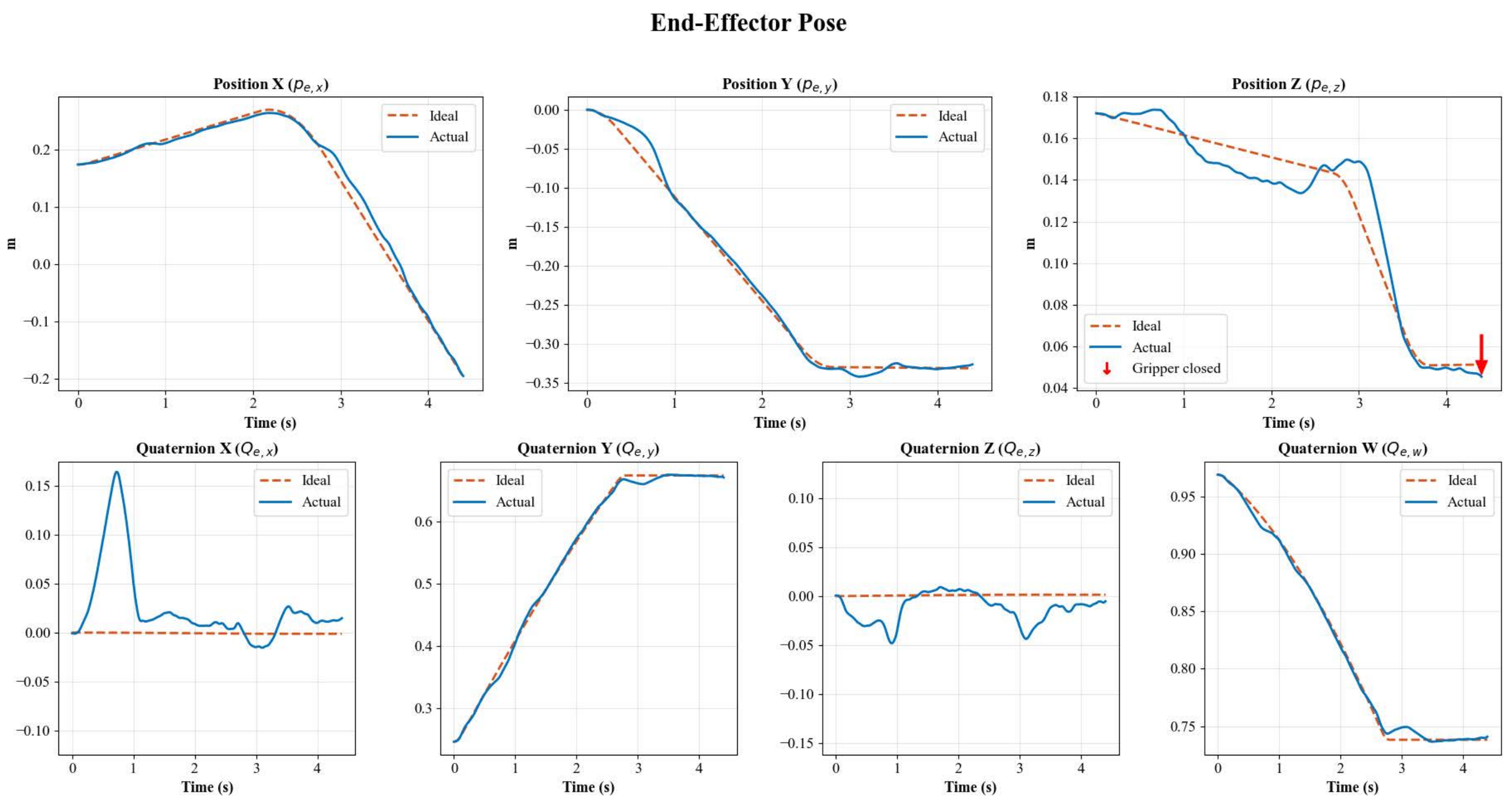


Figure 28: Ideal (dashed) vs. actual (solid) end-effector pose trajectories at a relative object speed of 0.25 m/s. The RMSE is 0.0186 m for position and 0.052 (unitless) for orientation in quaternion space.

Concerning the quaternion component of the pose (used to describe the orientation of the end-effector with respect to the base frame of the manipulator), after a preliminary analysis, at approximately $t = 0.75\ s$ a spike is visible in the $Q_{e,x}$ component (see Figure 28). To verify whether this spike corresponds to a meaningful orientation error, the two quaternions are first converted into rotation matrices; their relative rotation is then analyzed and finally expressed in Euler angles. The actual and ideal quaternions at that instant are

$$\boldsymbol{q}_{act} = (0.1624,\ 0.3511, -0.02611,\ 0.9217), \tag{6.1}$$

$$\boldsymbol{q}_{id} = (0.0002,\ 0.3663,\ 0.0005,\ 0.9305), \tag{6.2}$$

where the ordering is (q_x, q_y, q_z, q_w) and $\|\boldsymbol{q}\| = 1$. Recall that for a unit quaternion $\boldsymbol{q} = (q_x, q_y, q_z, q_w)$, the corresponding rotation matrix is

$$\boldsymbol{R}(\boldsymbol{q}) = \begin{bmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y - q_zq_w) & 2(q_xq_z + q_yq_w) \\ 2(q_xq_y + q_zq_w) & 1-2(q_x^2+q_z^2) & 2(q_yq_z - q_xq_w) \\ 2(q_xq_z - q_yq_w) & 2(q_yq_z + q_xq_w) & 1-2(q_x^2+q_y^2) \end{bmatrix}. \tag{6.3}$$

Applying (Equation 6.3) yields

$$\boldsymbol{R}_{act} = \begin{bmatrix} 0.7521 & 0.1622 & 0.6387 \\ 0.0659 & 0.9459 & -0.3177 \\ -0.6557 & 0.2810 & 0.7007 \end{bmatrix}, \qquad \boldsymbol{R}_{id} = \begin{bmatrix} 0.7316 & -0.0008 & 0.6817 \\ 0.0011 & 1.0000 & -0.0000 \\ -0.6817 & 0.0007 & 0.7316 \end{bmatrix}. \tag{6.4}$$

Then, the relative rotation between ideal and actual frames at that specific time instant is obtained as

$$\boldsymbol{R}_{rel} = \boldsymbol{R}_{act}\ \boldsymbol{R}_{id}^{\top} = \begin{bmatrix} 0.9856 & 0.1630 & -0.0452 \\ -0.1691 & 0.9460 & -0.2767 \\ -0.0023 & 0.2803 & 0.9599 \end{bmatrix}. \tag{6.5}$$

Had the two quaternions been identical, $\boldsymbol{R}_{rel}$ would be exactly the identity matrix; the small off-diagonal terms therefore quantify the misalignment. Using the ZYX convention, the error angles $(\phi_{err}, \theta_{err}, \psi_{err})$ are obtained from (Equation 6.5) as

$$\begin{aligned} (\phi_{err}, \theta_{err}, \psi_{err}) &= (\text{atan2}(R_{32}, R_{33}),\ -\arcsin(R_{31}),\ \text{atan2}(R_{21}, R_{11})) \\ &= (0.284,\ 0.0023,\ -0.170)\ rad \\ &\approx (16.3^\circ,\ 0.1^\circ,\ -9.7^\circ). \end{aligned} \tag{6.6}$$

Despite the visible spike in the $Q_{e,x}$ plot, these values indicate a moderate deviation. Furthermore, this deviation occurred only for a brief period and not at the end of the trajectory. Overall, it did not undermine the success of the grasping motion for the task being studied.

The group of plots in Figure 27 is the one that shows the most evident deviations between the ideal and actual trajectories, and it is about joint velocities. This outcome is expected, as joint velocity is not explicitly controlled by the position controller, which regulates only joint positions via PID loops. As a result, the velocity profiles do not precisely track the ideal trajectories, especially during the initial phase of motion where rapid accelerations are required. Nonetheless, it can be observed that velocity tracking improves toward the end of the trajectory, where accelerations are smaller. This suggests that the joint actuators may have limitations in producing high acceleration commands, possibly due to torque or current constraints. Interestingly, the system appears more capable of decelerating than accelerating, as actual velocities tend to converge more quickly when decreasing than when ramping up. Despite the discrepancies observed at the beginning of the motion, the final joint velocities closely match the reference values, indicating a successful completion of the planned motion and supporting the effectiveness of the grasp execution. Two additional plots that help characterize the

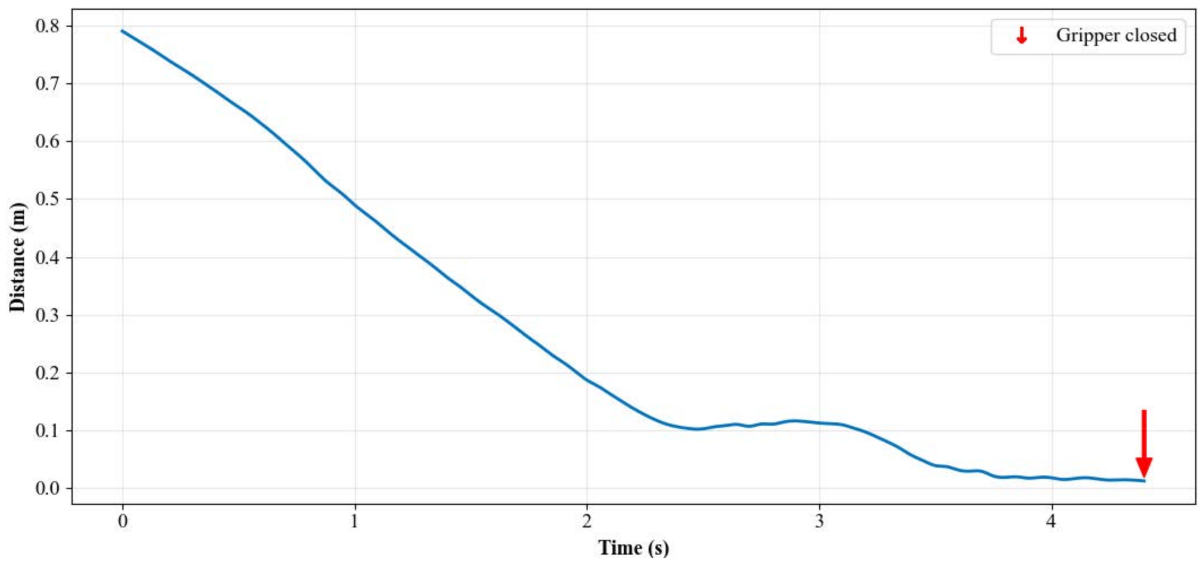


Figure 29: Euclidean distance between the end-effector and the object over time during a test at a relative object speed of 0.25 m/s.

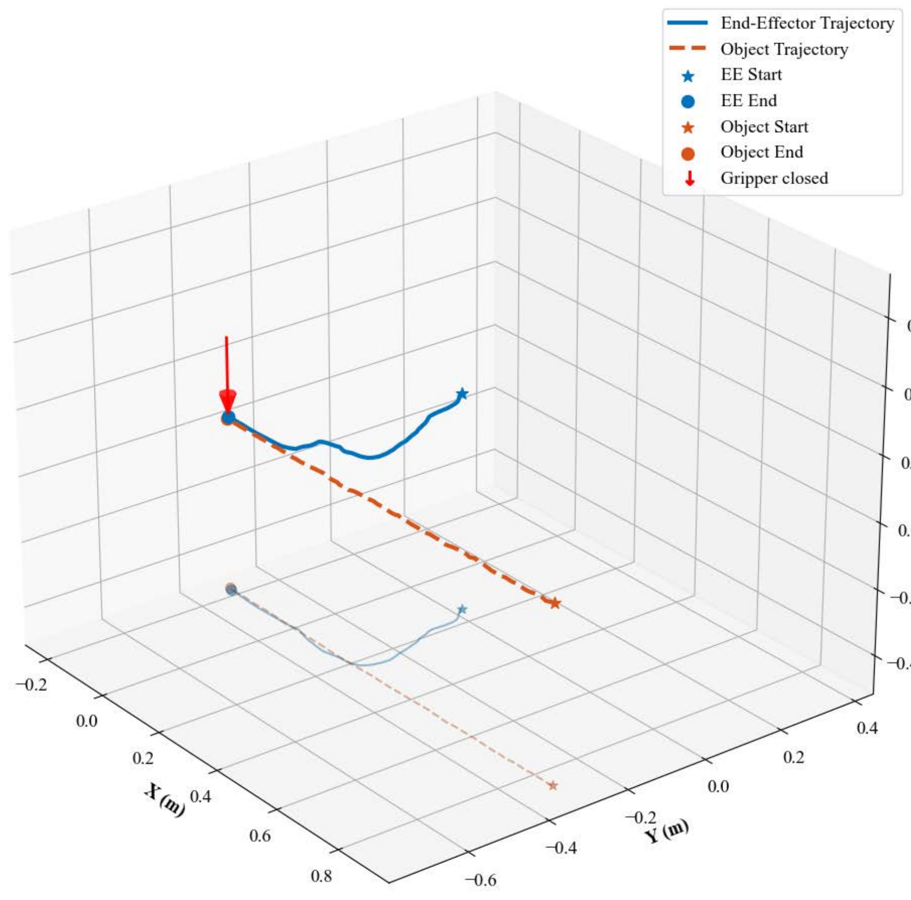


Figure 30: Actual trajectories of the end-effector (solid) and the object (dashed) in the base frame of WidowX 250 S at a relative object speed of 0.25 m/s.

end-effector's behavior (as described in Subsection 3.3.4) during the grasp sequence are shown in Figure 29 and Figure 30. The first presents the evolution in time of Euclidean distance between end-effector and target object, specifically measured between the tool center point and the object's center of mass, from the moment the manipulator starts the grasp sequence until it the gripper is fully closed and the object is grasped. In the figure it is possible to observe a short, almost flat portion of the curve where the distance appears to be constant: this is the point where the robot arm gets to place its end-effector over the object and ”inverts” its motion to match the object’s velocity. This phase is also evident in the spatial trajectory plot of Figure 30, where the end-effector’s path exhibits a change in direction, aligning more closely with the trajectory of the moving object. Toward the end of the sequence, the end-effector continues to follow the object at close range, maintaining alignment until the gripper is fully closed and the object is secured. Additional results of the same type of test (successful grasp) at a lower speed of 0.05 m/s are illustrated in Appendix B.

Let us now analyze the outcomes of an unsuccessful experiment with the usual setup of Subsection 5.3.1, in which Husky moved forward at a speed of 35 cm/s, which is approximately the point at which the robotic arm no longer manages to perform a successful capture of the target object. The test failure in this case can be explained by several interconnected factors. First, at this relative speed, the object crosses the reachable workspace in a very short time. The largest diameter of the workspace, achieved only when the arm is fully extended (including wrist joints), is approximately 130 cm (see Table II). Realistically, when the wrist is bent to grasp the object as in any of the described trajectories for grasping moving objects, the span reduces to approximately 100 cm. This still refers to the maximum diameter, which is not entirely crossable since it passes through the robot base. Therefore, the portion of reachable workspace traversed by the object forms a chord whose length is at least 10-15% smaller than the maximum diameter, reducing the distance traveled within the graspable area to 85 cm

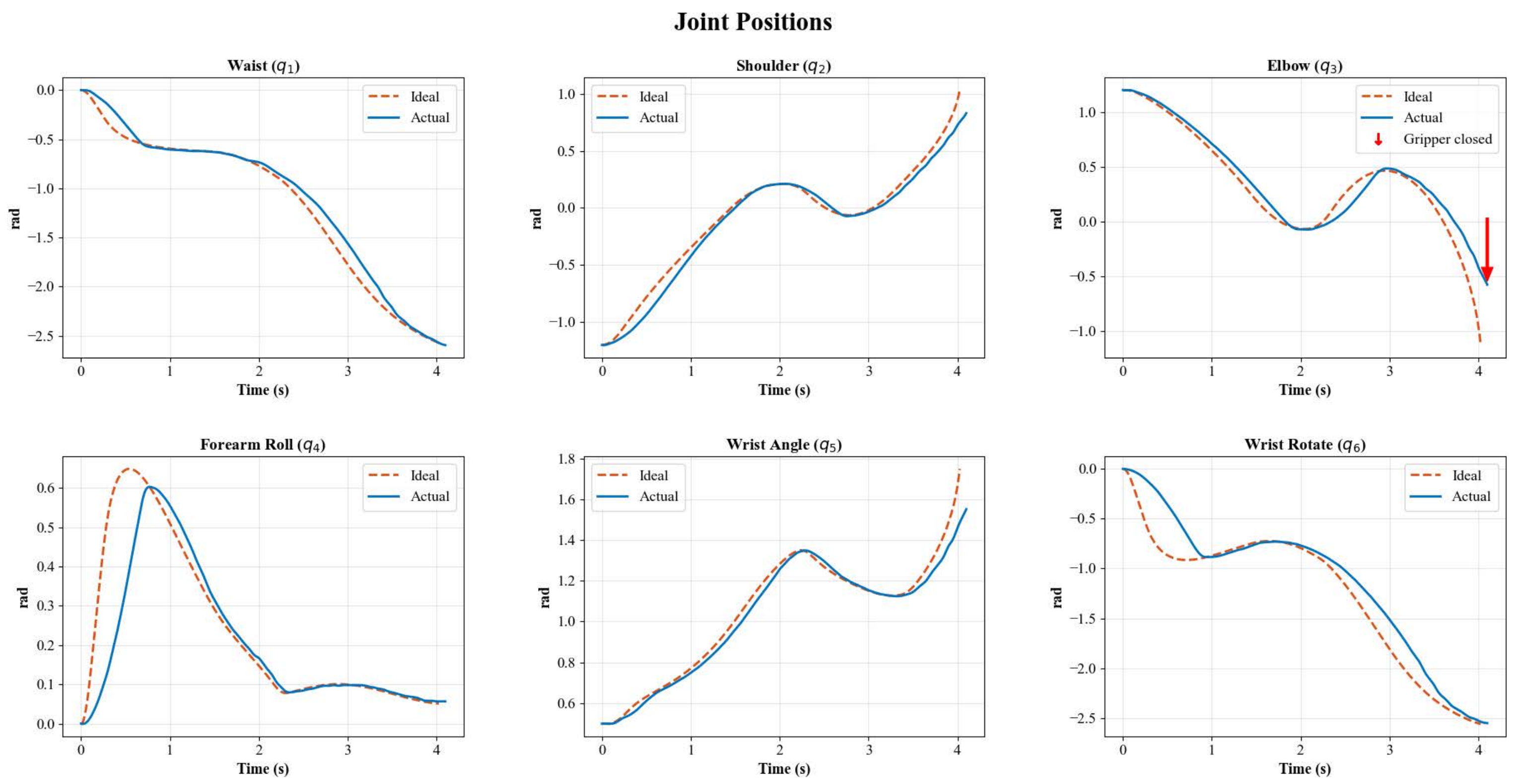


Figure 31: Ideal (dashed) vs. actual (solid) joint position trajectories for the six arm joints during dynamic grasping at a relative object speed of $0.35\,m/s$. The joint-space RMSE is $0.3059\,rad$.

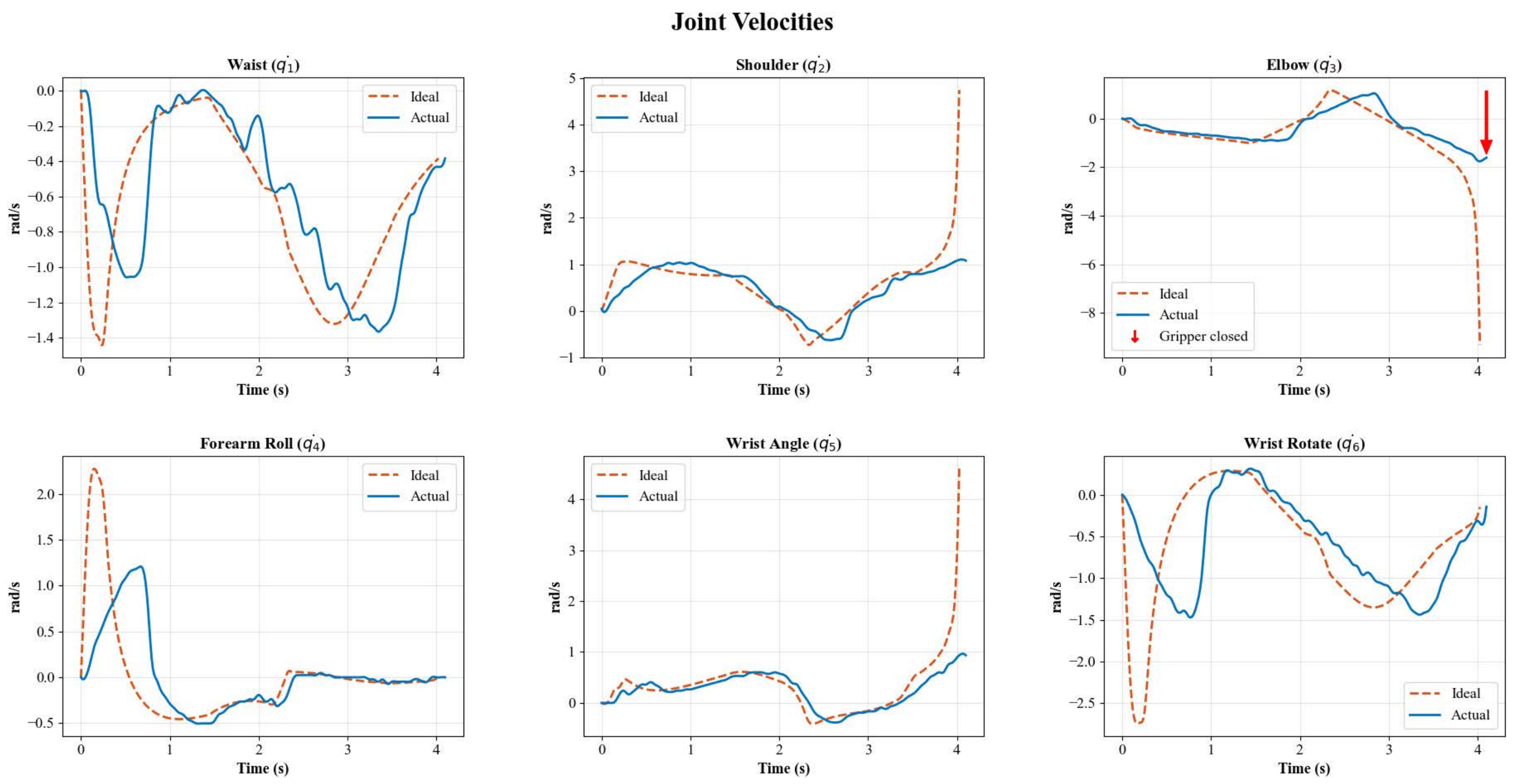


Figure 32: Ideal (dashed) vs. actual (solid) joint velocity trajectories for the six arm joints during dynamic grasping at a relative object speed of 0.35 m/s. The joint-space velocity RMSE is 1.302 rad.

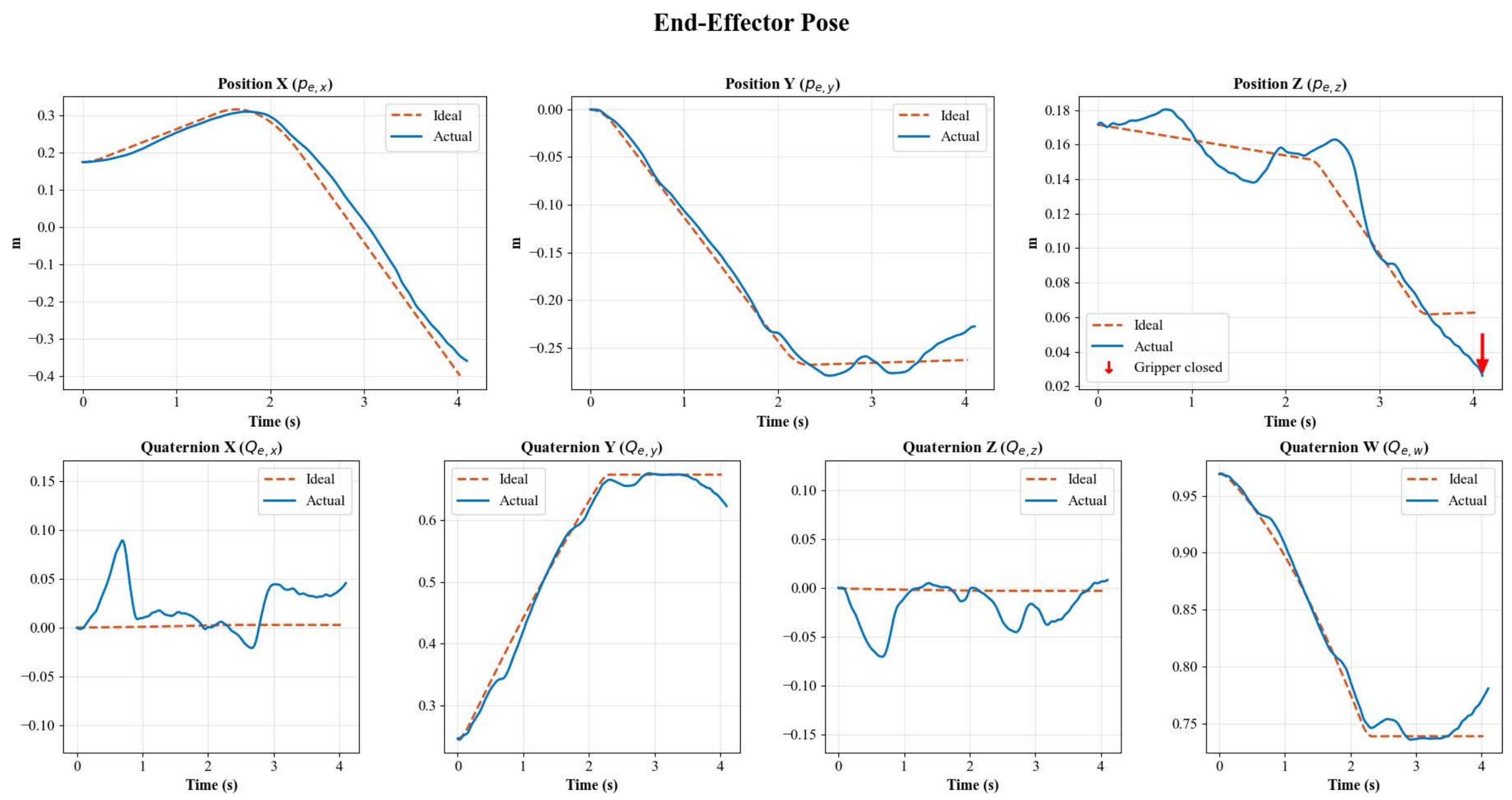


Figure 33: Ideal (dashed) vs. actual (solid) end-effector pose trajectories at a relative object speed of 0.35 m/s. The RMSE is 0.0353 m for position and 0.0441 (unitless) for orientation in quaternion space.

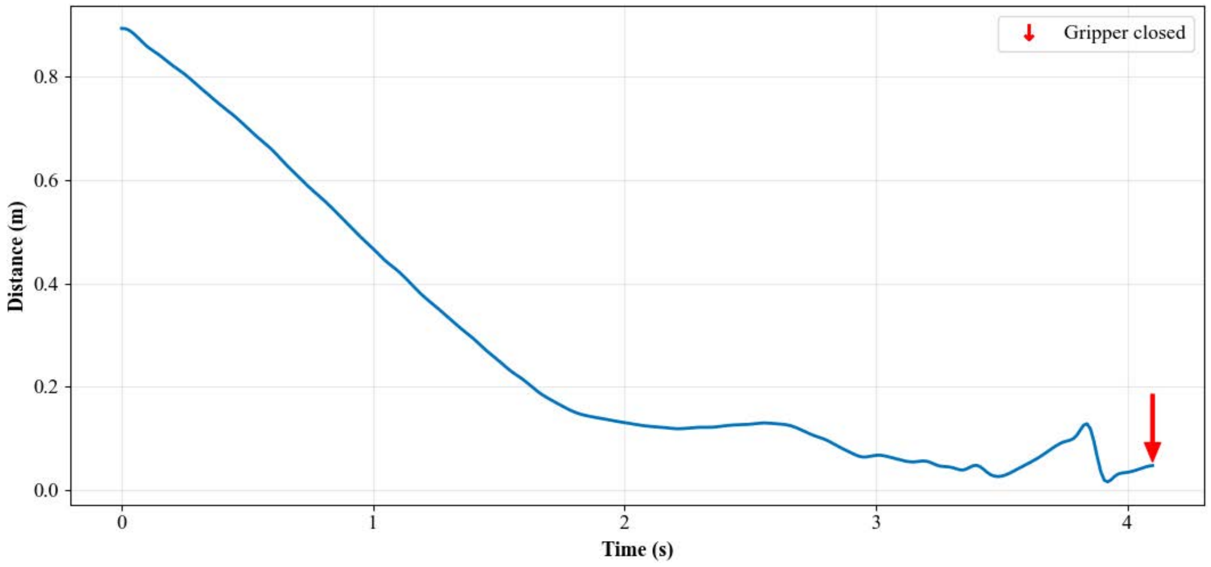


Figure 34: Euclidean distance between the end-effector and the object over time during a test at a relative object speed of 0.35 m/s.

or less. At 35 cm/s, this distance is covered in slightly more than 2 s, which represents the minimum time required for the dynamic grasping manager to activate the manipulator's motion, as described in Section 3.6. This implies that even if the robot arm attempts a grasp, it occurs at the workspace edge just before the target exits, creating instability as the robot arm loses much of its mobility when fully extended. When coupled with joint velocity and acceleration limits and the fact that the PID position controller struggles to counteract errors due to dynamic effects and gravity at higher speeds, these factors provide a complete explanation for why the dynamic grasping algorithm fails at relative speeds around 35 cm/s and above.

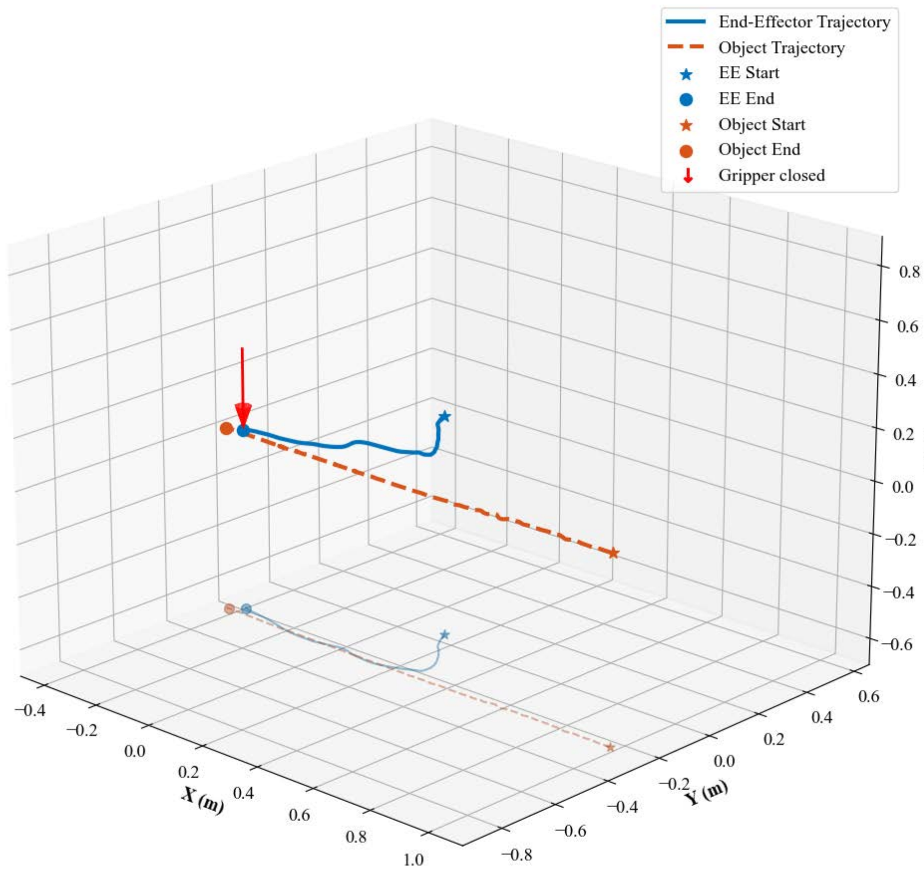


Figure 35: Actual trajectories of the end-effector (solid) and the object (dashed) in the base frame of WidowX 250 S at a relative object speed of 0.35 m/s.

The end-effector's proximity to the reachable workspace border is evident in Figure 31 and Figure 32. For joints responsible for full arm extension, such as Shoulder and Elbow, the planned joint velocities spike to extremely high values toward the end, clearly indicating full arm extension during those time instants. This behavior significantly impacted trajectory following, as shown in the joint position plots where the same joints experiencing velocity limit issues failed to track the desired trajectory, resulting in unstable robot movement as demonstrated in Figure 34. Overall, while the end-effector came remarkably close to capturing the object, the attempt failed due to degraded position controller performance and instability caused by operating at the workspace boundary.

TABLE VI: TRACKING ERROR VERSUS RELATIVE SPEED FOR JOINT POSITIONS (rad) AND JOINT VELOCITIES (rad/s).

Rel. speed (m/s)	**Joint positions**		**Joint velocities**	
	RMSE	**ℓ_∞ -norm**	**RMSE**	**ℓ_∞-norm**
0.05	0.2112	0.6638	0.9373	2.9083
0.10	0.2070	0.6578	0.8925	2.8144
0.15	0.2108	0.6919	0.8972	2.7582
0.20	0.2018	0.6380	0.8367	2.6553
0.25	0.2540	0.7523	0.9285	2.8111
0.30	0.2638	0.7193	0.9275	2.9237
0.35	0.3059	0.7162	1.3020	5.8519

As shown in Table VI and Table VII, the average and maximum deviations from the ideal trajectory, represented by RMSE and ℓ_∞-norm respectively, remain remarkably consistent across

TABLE VII: TRACKING ERROR VERSUS RELATIVE SPEED FOR END-EFFECTOR POSITION (m) AND END-EFFECTOR ORIENTATION (QUATERNION ERROR).

Rel. speed (m/s)	**End-effector position**		**End-effector orientation**	
	RMSE	**ℓ_∞-norm**	**RMSE**	**ℓ_∞-norm**
0.05	0.0126	0.0279	0.0282	0.0955
0.10	0.0136	0.0315	0.0281	0.0877
0.15	0.0128	0.0359	0.0329	0.1049
0.20	0.0143	0.0424	0.0405	0.1488
0.25	0.0186	0.0419	0.0520	0.1658
0.30	0.0254	0.0567	0.0275	0.0551
0.35	0.0353	0.0647	0.0441	0.1172

the relative speed range of 5 cm/s to 25 cm/s. This consistency is particularly pronounced for end-effector position, which represents the most critical element of the analysis, since it directly relates to our Cartesian space objectives. The average end-effector position error for the trial conducted at Husky's speed of 35 cm/s is nearly twice the average position error observed for speeds within the 5 cm/s to 25 cm/s range. Similarly, elevated values appear in the joint velocities section of Table VI, attributed to the significant trajectory misalignment towards the end of the motion when the manipulator operates at the boundary of reachable workspace.

As summarized in Figure Figure 36, the success rate of dynamic grasping experiments remains consistently high in simulation across all tested speeds, with 100% successful captures even at the highest relative speed; this happens since in Gazebo, joint position commands

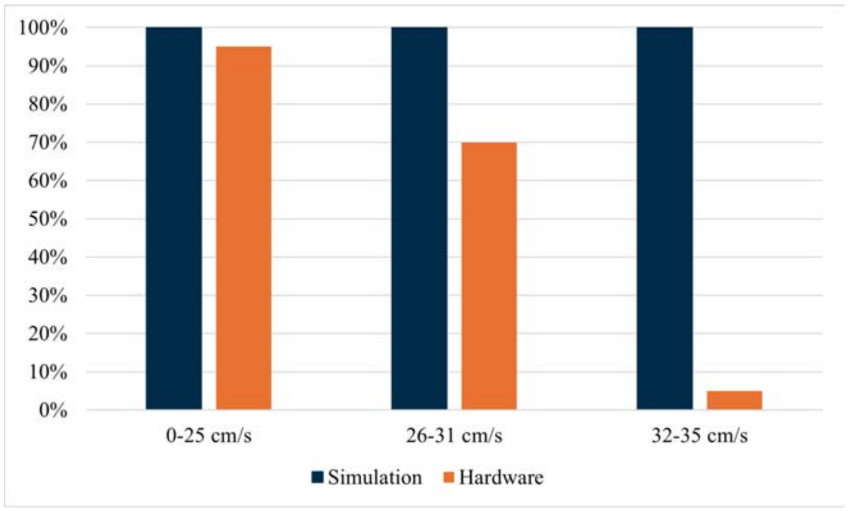


Figure 36: Success rate of the dynamic grasping control system as a function of relative object speed, comparing results obtained in simulation and on hardware.

are executed by moving the joint smoothly toward the target using simulated controllers and physics, which often results in more precise movement than on the real robot due to the absence of real-world imperfections. In fact, hardware tests reveal a noticeable decline in performance as the relative object speed increases. While relative speeds up to 25 cm/s achieve success rates of 95%, performance starts to drop significantly beyond this threshold. At speeds of about $26-31$ cm/s, the hardware success rate falls to approximately 70%, primarily due to the more relevant effect of dynamic disturbances on the position controllers and the limited time the object remains within the reachable workspace, particularly when its trajectory crosses the workspace far from the center. At $32-35$ cm/s, successful grasps are basically almost never guaranteed. These results highlight the impact of physical limitations such as actuator dynamics and unmodeled disturbances, which are absent in the idealized simulation environment. Note that the infrequent failures at speeds within 25 cm/s are mainly not speed-related, but rather stem from undesired disturbances such as higher mechanical vibrations of the entire system caused by Husky's motion or erroneous data acquisition from the vision system.

6.3 Performance and accuracy in dynamic scenarios

Given the way the employed trajectory planner and control system operate, there is an intrinsic difficulty in quantitatively assessing the difference between nominal and executed trajectories in tests involving changes in the object's relative motion. In such scenarios, the manipulator must dynamically adapt its plan, alternating periods of constant motion with other time intervals during which it is stopped and does not have a planned trajectory to follow, making it challenging to establish a consistent reference trajectory. Consequently, performance and accuracy were primarily evaluated through visual inspection, verifying whether the end-effector was able to successfully intercept the target.

Through this visual assessment approach, the experimental results demonstrate that successful target capture after a motion change is highly dependent on the target's speed range. At low speeds (about 0-10 cm/s), the system maintains a very high capture likelihood even when subjected to two rapid direction changes, indicating robust tracking capabilities within this operating range. Performance degrades significantly as target speed increases, with moderate success (about 10-15 cm/s) only achievable when a single quick change occurs within the workspace, and low success rates (15-18 cm/s) requiring that direction changes happen before workspace entry or shortly thereafter. Beyond 18 cm/s, the system shows very low to no success in target capture, even with optimal timing of directional changes. These findings confirm that the system's responsiveness is fundamentally limited by target velocity, with the optimal operating range for what concerns the starting velocity of 0-25cm/s, providing the best conditions for the end-effector to initiate motion and align with moving targets. The results underscore the critical importance of maintaining target speeds within the low-to-medium velocity ranges to ensure reliable capture performance in dynamic tracking scenarios, where a motion change is expected.

To complement this visual evaluation, recorded joint data allowed for a-posteriori analysis, enabling the generation of two complementary plots: a 3D animation of the actual end-effector and object trajectories in the robot base frame, and a plot showing the evolution of their Euclidean distance over time. The animation is presented through a figure displaying six successive snapshots capturing the most critical moments of the interaction. These figures provide additional insight into the system's behavior under dynamic conditions.

Figure 38 directly refers to the test described in Subsection 5.3.3 and illustrates an object trajectory featuring three changes in the direction of motion, executed at a speed of 10 cm/s. In the initial portion of the trajectory, the end-effector attempted to move over the object and align with its original path. This smooth phase concludes with a full stop, coinciding with a sharp bend in the end-effector trajectory. From that point onward, the manipulator attempted to replan its motion; however, slightly after a new trajectory was computed and dispatched, another change occurred, requiring the process to be repeated. The grasp sequence successfully concluded just before the object exited the reachable workspace, as shown in Figure 25c. The

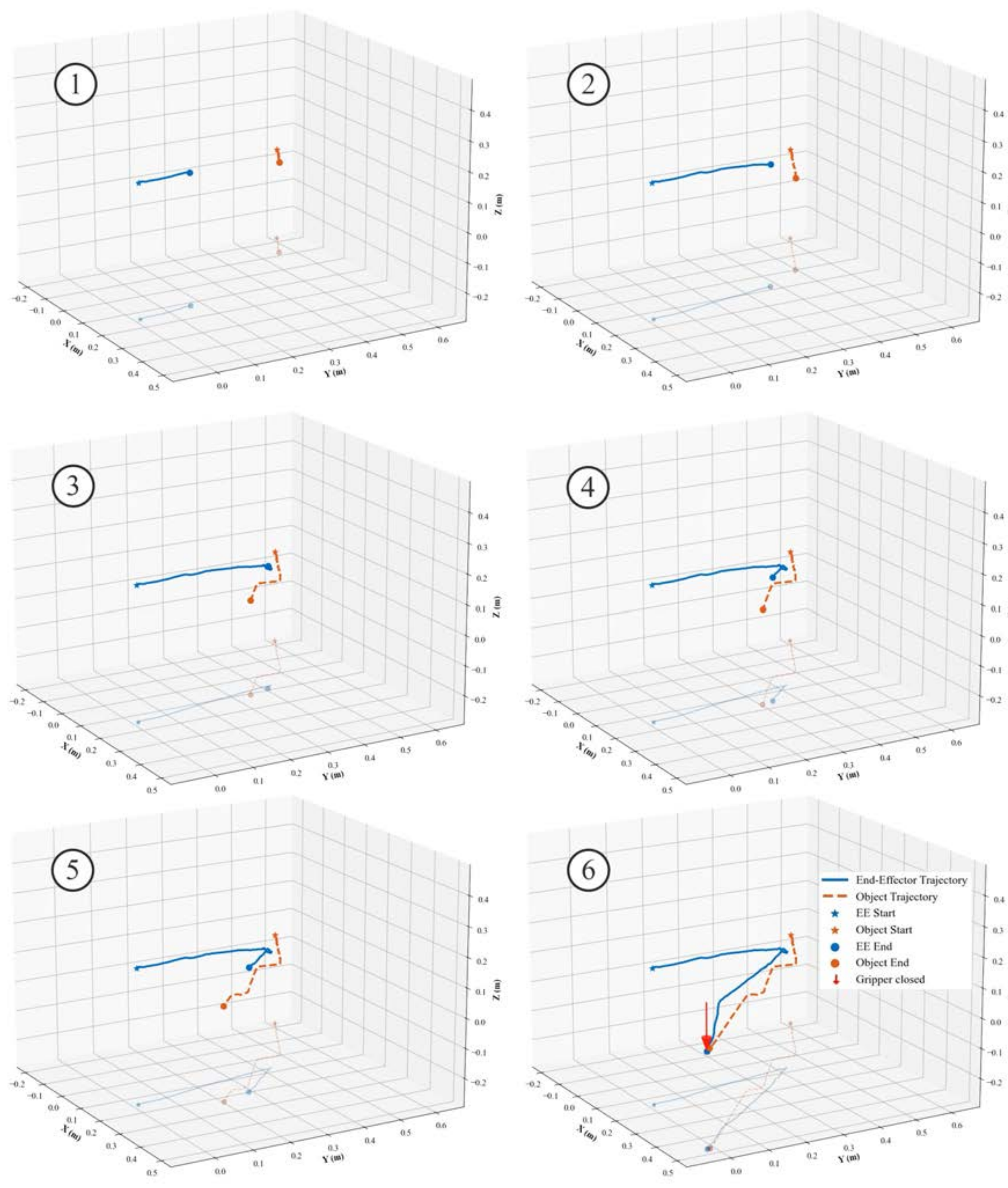


Figure 37: End-effector (solid) and object (dashed) 3D trajectories in the direction-change test performed at a relative speed of 0.10 m/s.

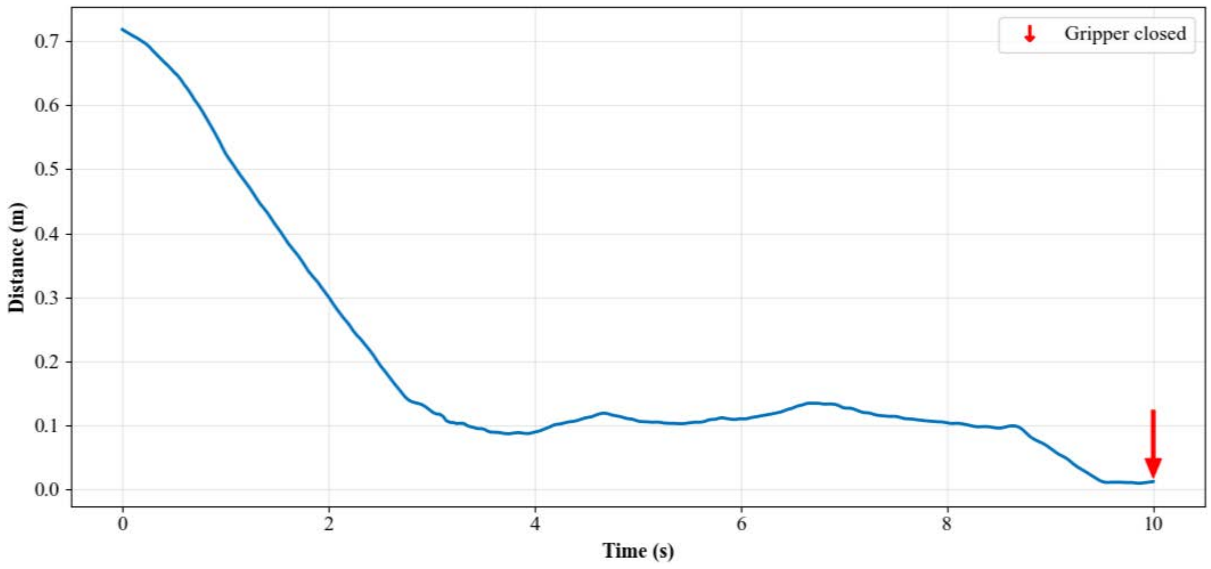


Figure 38: Euclidean distance between end-effector and object over time during the direction-change test performed at a relative speed of 0.10 m/s.

instant at which the robot arm halts to reprogram itself can be identified in Figure 38 by examining the nearly flat segment of the curve. Following the initial decrease in distance, two small bumps appear around 4.7 s and 6.7 s. These correspond to moments when the distance temporarily increased, immediately after the manipulator paused to replan its trajectory. In fact, while the end-effector remained stationary during replanning, the object continued to move along its new direction, momentarily increasing the separation.

Another notable detail is the time required for the end-effector to reprogram its trajectory once the object's velocity stabilizes. As expected, experimental results indicate this interval lasts approximately 0.5 s. This duration is primarily attributed to latency introduced by the velocity estimator of Subsection 3.5.2, which, by design, employs a window of 120 samples. Consequently, when the target changes direction and reaches a new constant velocity, the estimator needs about half a second to reflect the updated motion based on position data. An additional delay follows, as the dynamic grasping manager and control system recognize the new motion stability and generate the corresponding commands for execution.

Similar observations can be drawn from the results of other tests including deceleration phases after the manipulator already started it motion or scenarios in which Husky was brought to a complete stop, leading to a new simplified trajectory for stationary objects, that only includes the first two segments: in this case the end-effector does not necessitate to track the object, since it is not moving relative to its base frame. Additional plots from these experiments are shown in Appendix C.

CHAPTER 7

CONCLUSIONS

This thesis has presented a comprehensive framework for dynamic object grasping using a robotic manipulator, addressing the challenges of real-time motion prediction and adaptive control in non-static scenarios. The system developed throughout this work enables a 6-DOF robotic arm to successfully identify, track, and capture objects moving with a constant linear velocity relative to it, under a variety of dynamic scenarios. This capability was made possible through the integration of motion estimation, predictive trajectory planning, and adaptive grasp execution, all coordinated via a finite-state machine that updates the system state at a 50 Hz frequency.

At the core of the architecture is an efficient real-time motion planner that, together with a custom analytic inverse kinematics solver, enables the robot to generate and execute time-optimal end-effector trajectories in under a millisecond. This computational efficiency was a key enabler for maintaining the reactivity required for dynamic grasping. The design also accounted for practical constraints such as gripper closure delays and object geometry, using motion-aware orientation strategies to increase grasp robustness.

Experimental results confirmed that the system is capable of successfully grasping objects moving at relative speeds up to roughly 30 cm/s, with a very high success rate in this range. Beyond this threshold, performance began to degrade due to kinematic and actuation limitations, as the time during which the object remains in the robot's reachable workspace becomes insufficient for safe execution. The ability to adapt to changes in the object's motion, both in magnitude and direction, was also demonstrated. In such scenarios, the system reacts by pausing its current motion, monitoring for trajectory stabilization, and replanning as soon as the tracked relative motion becomes steady (constant and linear) again. These capabilities were validated through both realistic simulations in Gazebo and hardware experiments, employing different setups, with the robotic arm mounted on a UGV, or stationary on a support.

A key contribution of this work lies in the dynamic grasp triggering logic, which moves beyond static workspace boundaries by leveraging real-time kinematic feasibility checks and velocity-based trajectory extrapolation. This adaptive grasp prediction strategy allows the robot to make informed decisions about when and how to engage in a grasp attempt, rather than relying solely on geometric proximity. Additionally, the modular ROS 2-based software design ensures that the developed framework can be extended or repurposed in broader loco-manipulation contexts, including multi-robot cooperation and field robotics.

Nevertheless, a number of practical constraints emerged during implementation and testing, highlighting specific areas for improvement:

- Servo tracking at high dynamics: the DYNAMIXEL position controller shows reduced tracking accuracy at elevated speeds and accelerations, as gravity and other dynamic forces are not currently compensated.
- Dependence on an external vision system: while the MoCap system delivers excellent precision for position data at very high rates, it restricts deployment to controlled environments and is not always a viable solution for certain applications. However, it can

still be employed in indoor applications where there is the possibility to install and use a motion capture system.

- Trajectory complexity limitations: objects following non-linear paths, such as circular motion or constant acceleration, fall outside the grasp manager's predictive capabilities.
- Gripper geometry constraints: the small and simple gripper design limits the system to handling only objects that match its basic geometry and finger opening range.

From these observations, it is possible to describe what could be some future improvements:

- Unified WidowX–UGV control architecture: host the control logic on the UGV's on-board Jetson using ROS 2 and powering the arm directly from the robot vehicle, enabling fully autonomous operation.
- Vision-based tracking: transition from motion capture to RGB cameras with depth sensors. Though this entails addressing a reduced field of view (potentially requiring multiple cameras), vibration compensation, lower data rates, and coarser position estimates, it would significantly increase applicability in varied environments.
- Support for non-linear trajectories: enhance the grasp manager to recognize and handle additional simple trajectory types, such as circular or uniformly accelerated motion.
- Advanced control methods: replace position PID control with inverse dynamics-based controller to actively counter dynamic disturbances, leveraging a higher control refresh rate.

- Dynamic planning fallback: for more complex motion that cannot be described through simple models, the system could temporarily switch from predictive grasp logic to real-time trajectory planning, leveraging Ruckig's online solver and possibly the inverse dynamics-based controller, that works at higher rates. This would allow the end effector to closely follow the object, minimizing waiting periods until a stable grasp window emerges.

Ultimately, with its blend of predictive planning, responsive control, and modular architecture, this framework establishes a solid foundation for responsive robot interaction with moving objects. Addressing the outlined limitations and pursuing the proposed improvements will move this work decisively toward operational autonomy in complex environments.

APPENDICES

Appendix A

GRASP TIMING AND DECISION LOGIC EXAMPLE

To better understand the grasp triggering condition involving the predicted entry time $\tau_{obj,in}$, consider the following illustrative scenario.

Suppose the robot requires $\tau_e = 3\ s$ to reach the predicted entry point, and the object is expected to remain inside the reachable workspace for a total duration of $\tau_{obj,in} = 3\ s$. Since the minimum required grasp window is 2 seconds, the object exceeds this minimum by 1 s, meaning it provides some timing flexibility.

In this case, the robot can initiate the grasp sequence even if the object is predicted to arrive at the entry point as early as $\tau_{obj,out} = \tau_e - (\tau_{obj,in} - 2) = 2\ s$. This works because the robot will arrive 1 second later, while the object will still be inside the workspace for 2 more seconds, satisfying the minimal dwell-time requirement.

If instead $\tau_{obj,out}$ were less than this threshold (e.g., 1.5 seconds), the object would have already been in the workspace for approximately 1.5 seconds when the robot arrives, leaving only 1.5 seconds to complete the grasp. This would violate the safety margin encoded in the duration condition, and thus the grasp would not be triggered.

Appendix B

ADDITIONAL RESULTS AT A LOWER RELATIVE SPEED

This appendix reports the results of the same dynamic grasping experiment discussed in Subsection 5.3.1, but performed with a lower relative speed between the end-effector and the object, set to approximately 5 cm/s. The structure of the plots is identical to those presented in Section 6.2: joint positions, joint velocities, end-effector pose, Euclidean distance to the object, and the 3D spatial trajectories are all included for completeness. These results are provided to illustrate the system's behavior in slower motion scenarios, although no significant deviations from the expected grasping performance are observed at this speed. It can be seen that the RMSE is generally lower, especially for position and orientation in the Cartesian space (see Figure 41): a lower relative speed of the object to track and capture relaxes the work the position PID controller which can therefore follow the desired trajectory with a higher lever of accuracy.

Appendix B (continued)

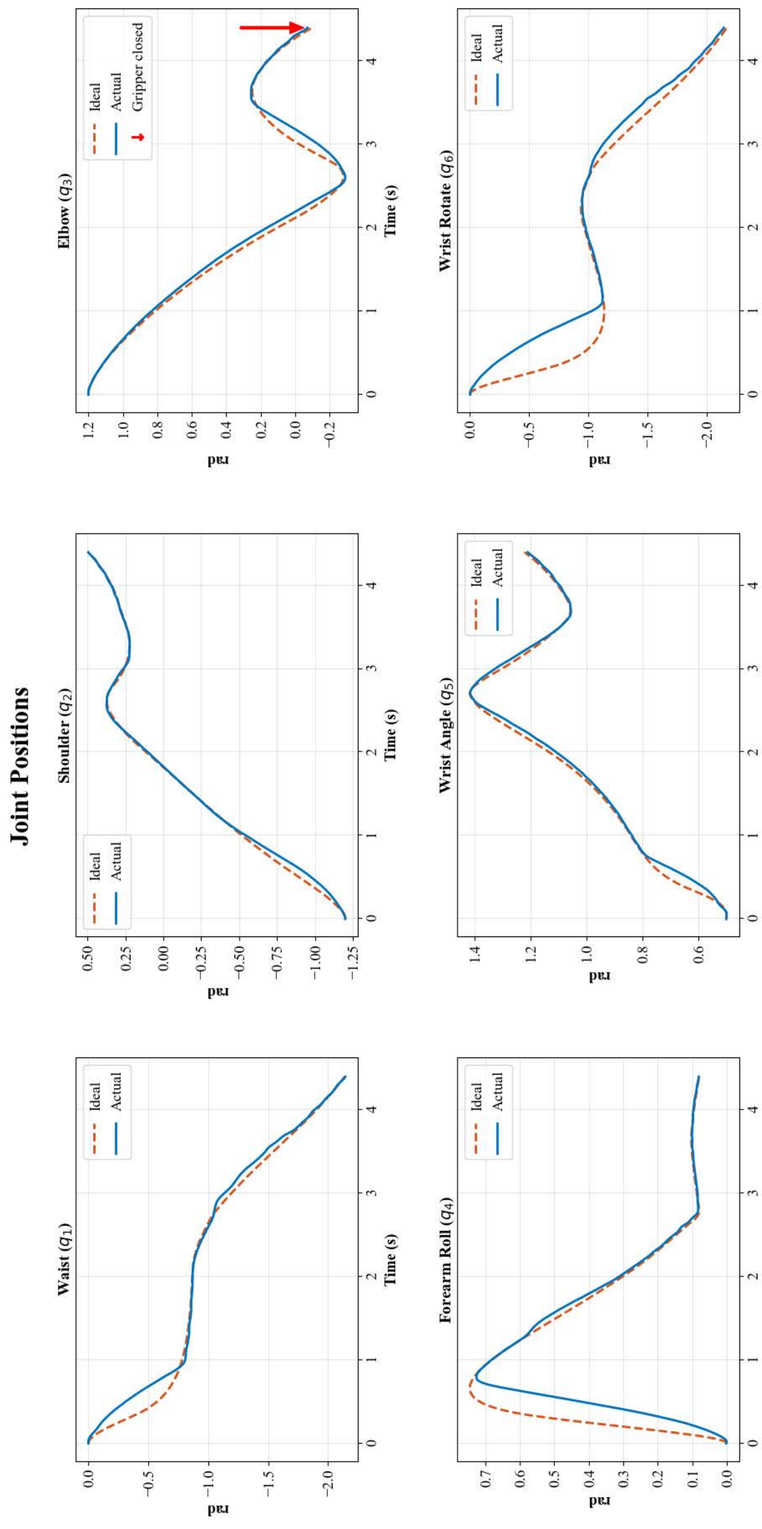


Figure 39: Ideal (dashed) vs. actual (solid) joint position trajectories for the six arm joints during dynamic grasping at a relative object speed of $0.05\,m/s$. The joint-space RMSE is 0.2112 *rad*.

Appendix B (continued)

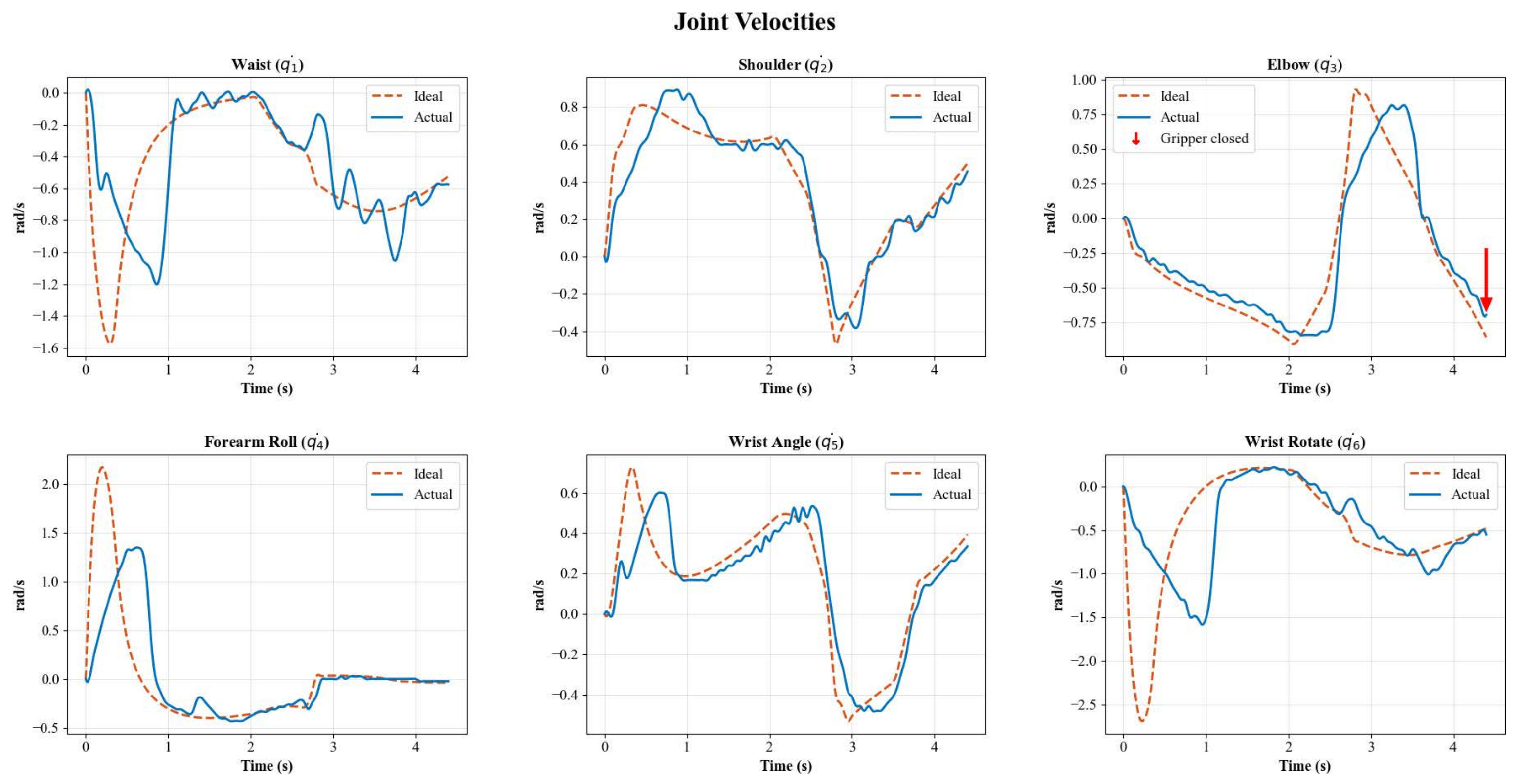


Figure 40: Ideal (dashed) vs. actual (solid) joint velocity trajectories for the six arm joints during dynamic grasping at a relative object speed of 0.25 m/s. The joint-space velocity RMSE is 0.9373 rad.

Appendix B (continued)

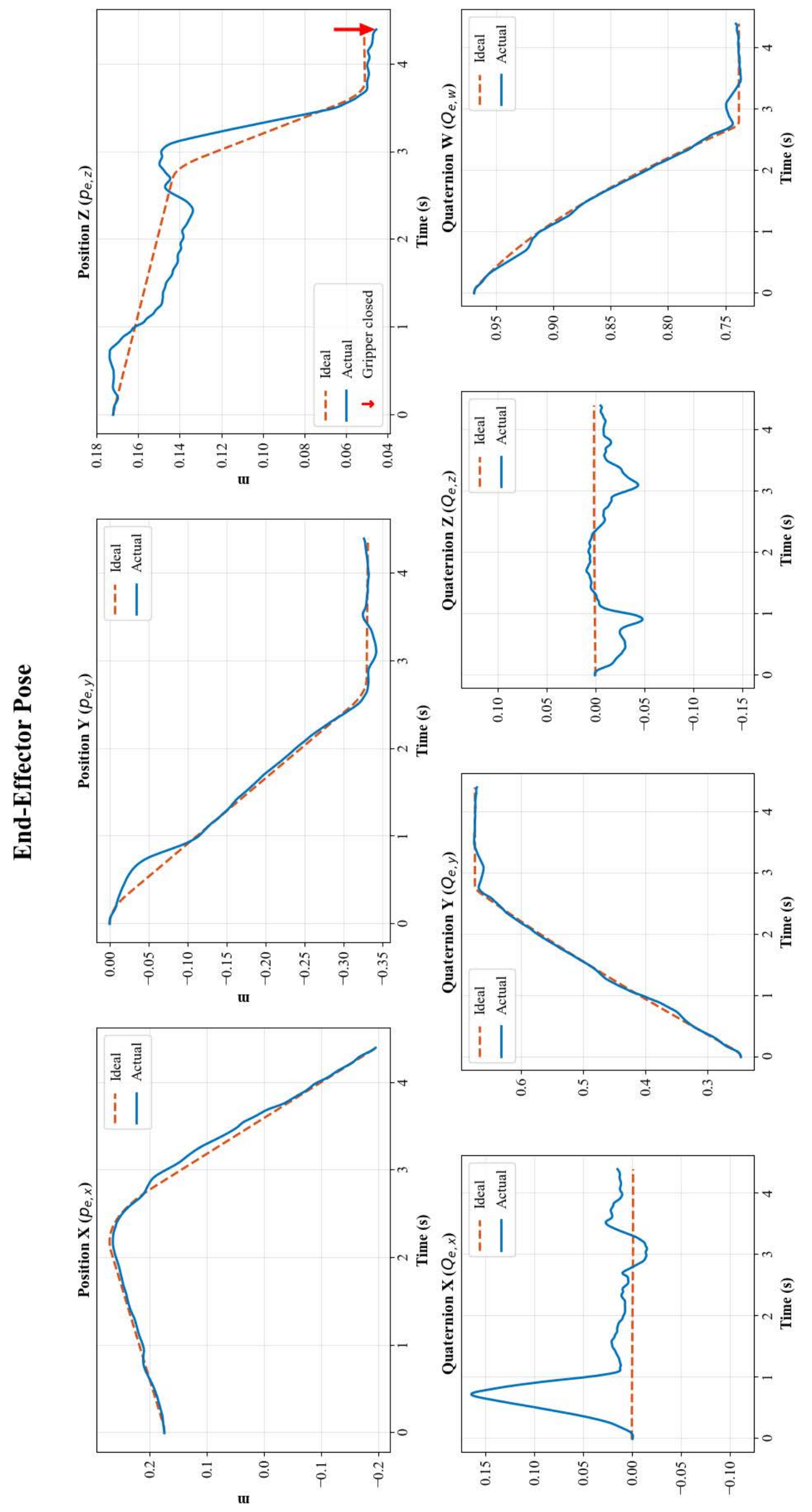


Figure 41: Ideal (dashed) vs. actual (solid) end-effector pose trajectories at a relative object speed of 0.25 m/s. The RMSE is 0.0126 m for position and 0.0282 (unitless) for orientation in quaternion space.

Appendix B (continued)

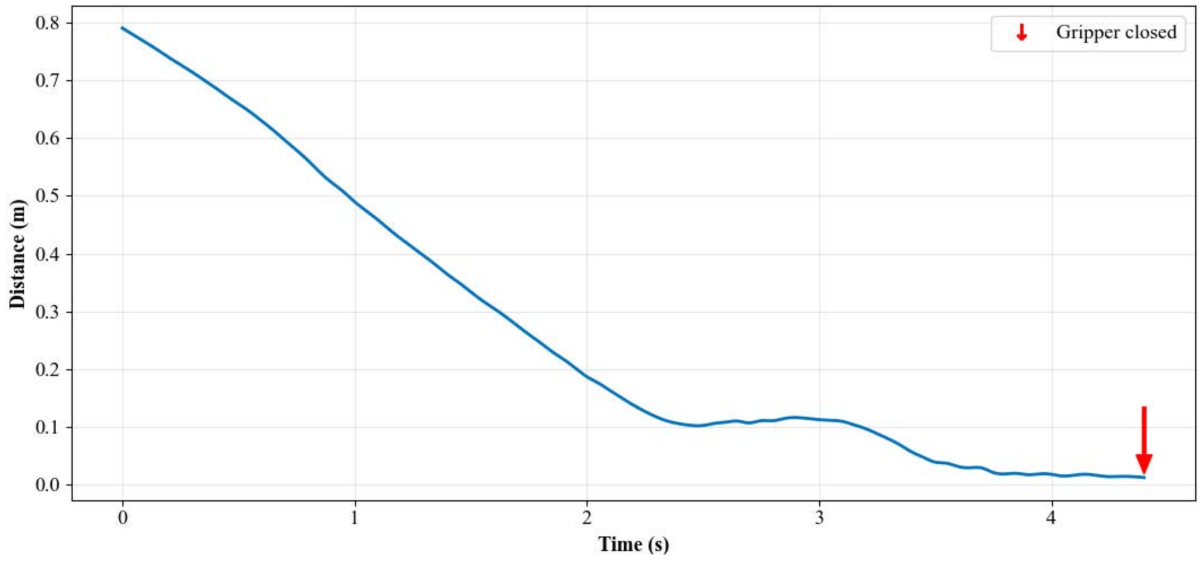


Figure 42: Euclidean distance between the end-effector and the object over time during a test at a relative object speed of 0.05 m/s.

Appendix B (continued)

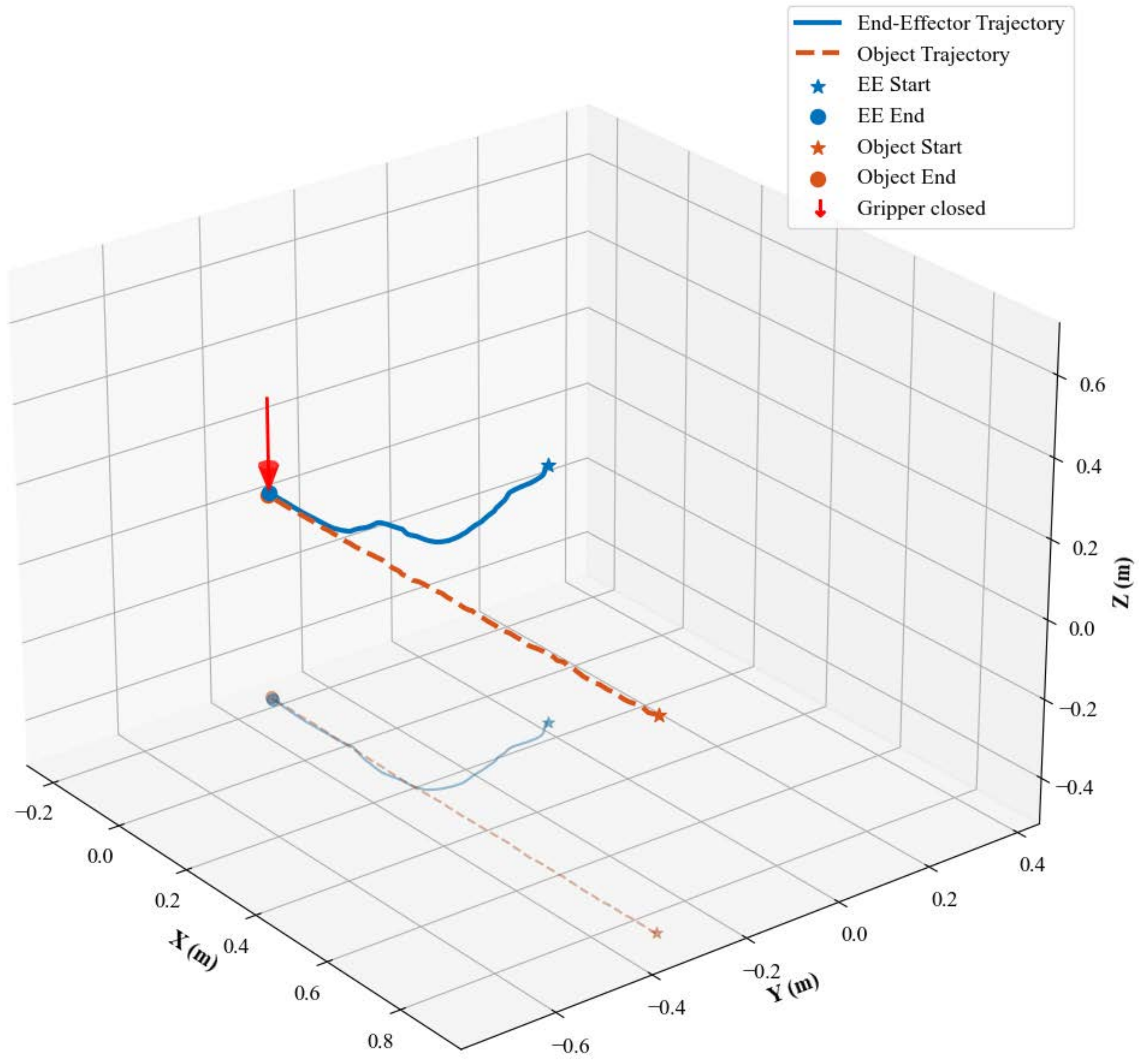


Figure 43: Actual trajectories of the end-effector (solid) and the object (dashed) in the base frame of WidowX 250 S at a relative object speed of 0.05 m/s.

Appendix C

ADDITIONAL RESULTS OF SCENARIOS WITH CHANGING VELOCITY

This appendix reports additional plots of experiments done in scenarios where the object velocity was changing either in magnitude or direction from the point of view of WidowX's base frame (not necessarily stationary).

In Figure 44 and Figure 46, it is possible to identify the point in space where the end-effector stopped, allowing the manipulator to reprogram a new trajectory. This point appears as a knot-like feature along the trajectory, characterized by small displacements of the tool center point. The irregularity is attributed to mechanical vibrations induced by the motion of the Husky platform during the stop.

Appendix C (continued)

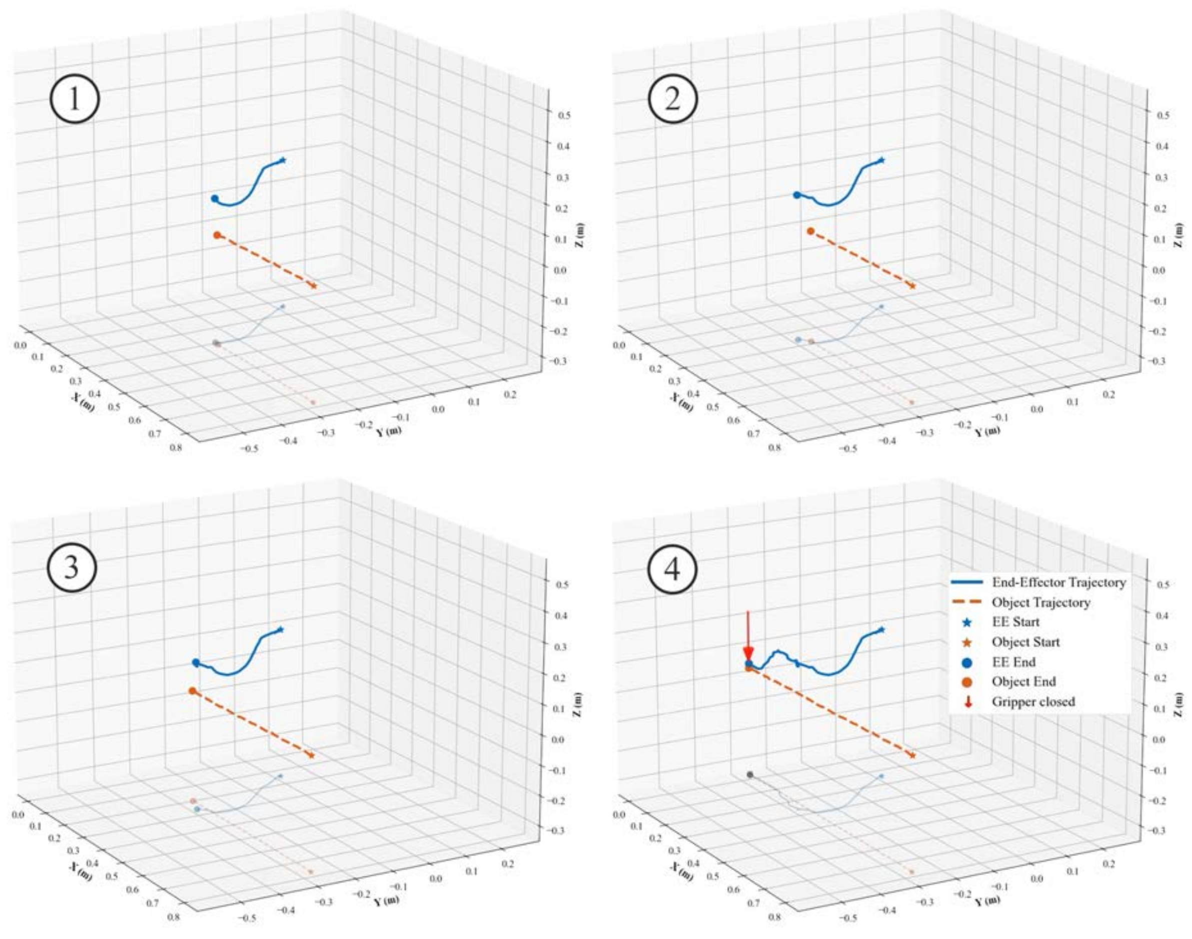


Figure 44: End-effector (solid) and object (dashed) 3D trajectories in a test during which the relative velocity was lowered from 0.20 m/s to 0.10 m/s.

Appendix C (continued)

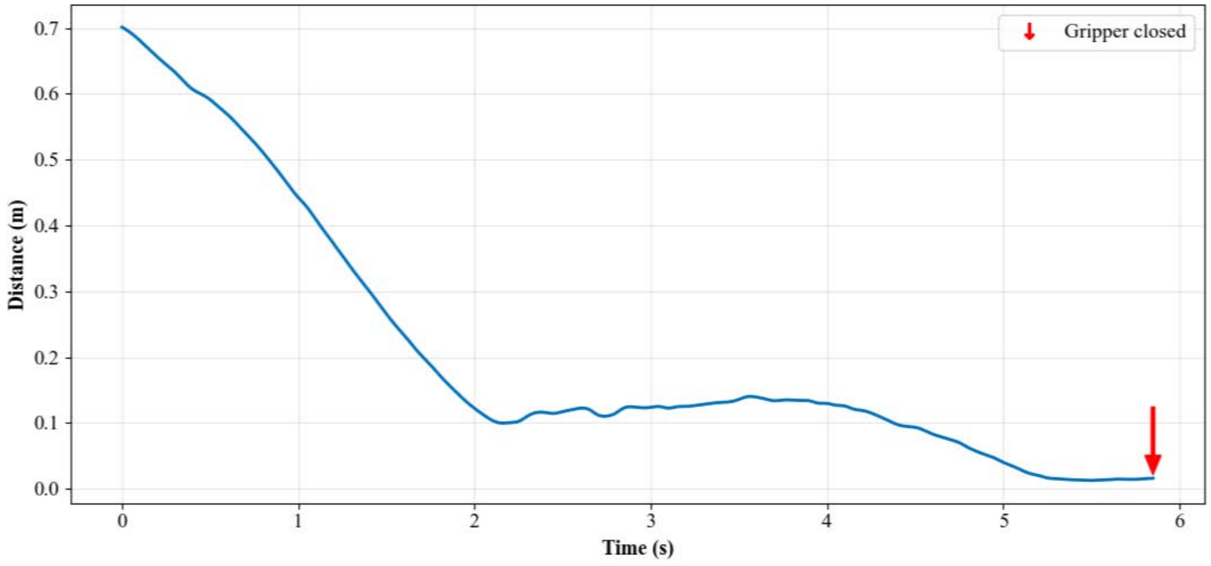


Figure 45: Euclidean distance between end-effector and object over time in a test during which the relative velocity was lowered from 0.20 m/s to 0.10 m/s.

Appendix C (continued)

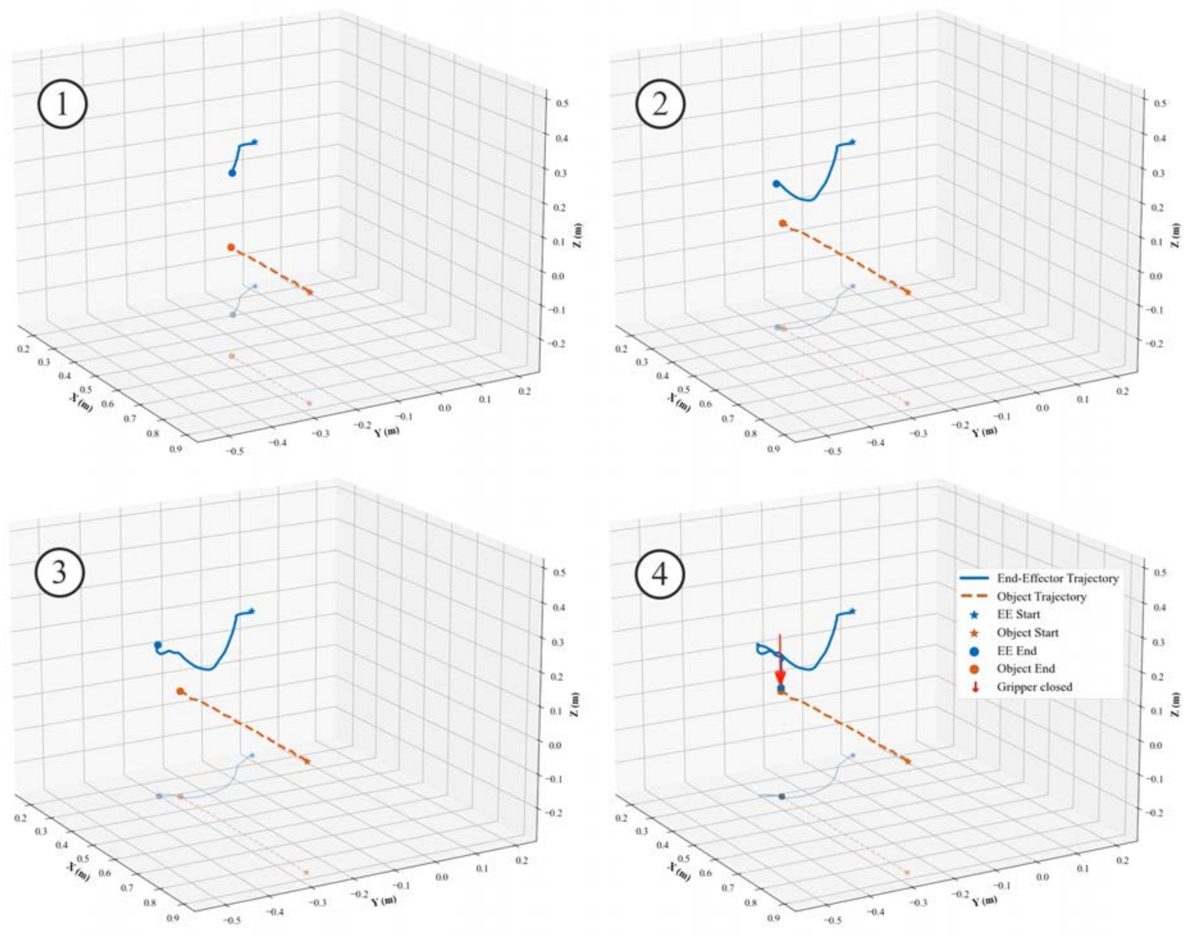


Figure 46: End-effector (solid) and object (dashed) 3D trajectories in a test during which the relative velocity was dropped to zero.

Appendix C (continued)

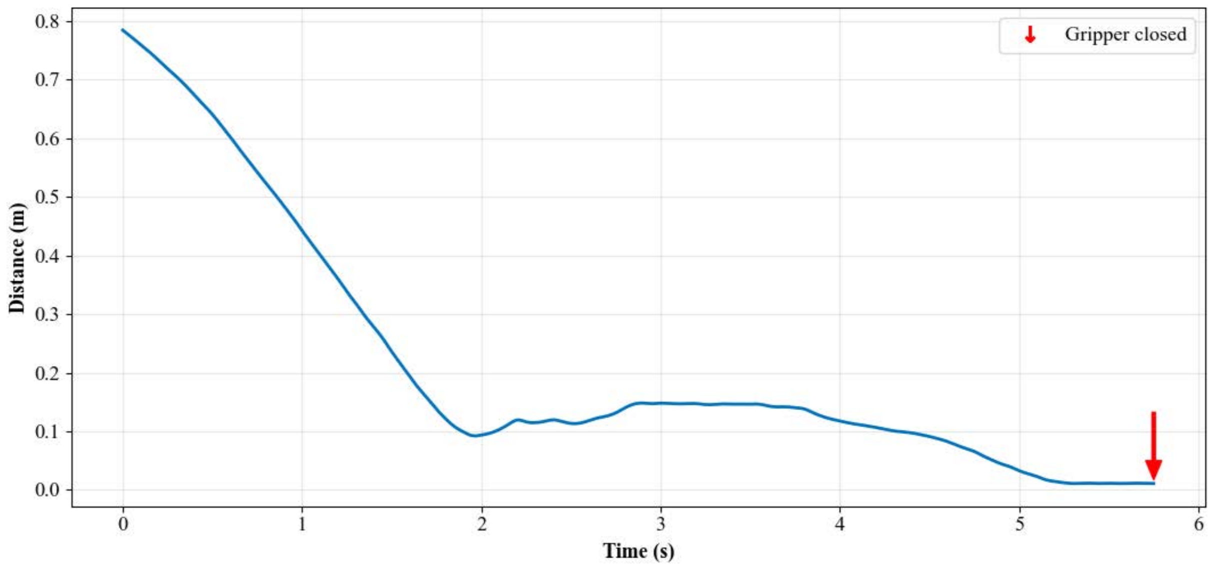


Figure 47: Euclidean distance between end-effector and object over time in a test during which the relative velocity was dropped to zero.

CITED LITERATURE

1. ROBOTIS: DYNAMIXEL XM430-W350 Position PID Controller. https://emanual.robotis.com/docs/en/dxl/x/xm430-w350/, 2025. Accessed: 2025-06-19.

2. ROBOTIS: DYNAMIXEL XL430-W250 Position PID Controller. https://emanual.robotis.com/docs/en/dxl/x/xl430-w250/, 2025. Accessed: 2025-06-19.

3. Trossen Robotics: WidowX 250 S Product Photograph. https://www.trossenrobotics.com/widowx-250. [Online; accessed June 4, 2025].

4. Trossen Robotics: WidowX 250 S Technical Drawing. https://www.trossenrobotics.com/widowx-250. [Online; accessed June 4, 2025].

5. Miller, A. and Allen, P.: Graspit! a versatile simulator for robotic grasping. IEEE Robotics and Automation Magazine, 11(4):110–122, 2004.

6. Ferrari, C. and Canny, J.: Planning optimal grasps. In Proceedings 1992 IEEE International Conference on Robotics and Automation, pages 2290–2295 vol.3, 1992.

7. Houshangi, N.: Control of a robotic manipulator to grasp a moving target using vision. In Proceedings., IEEE International Conference on Robotics and Automation, pages 604–609 vol.1, 1990.

8. Allen, P., Timcenko, A., Yoshimi, B., and Michelman, P.: Automated tracking and grasping of a moving object with a robotic hand-eye system. IEEE Transactions on Robotics and Automation, 9(2):152–165, 1993.

9. XiaoYang, Z., Ayub, M. A., Ruslan, F. A., Abdul-Rahman, S., and Abdullah, S. C.: Flexible target grasping strategy of industrial robot based on eye-in-hand 3d machine vision. In 2025 IEEE International Conference on Robotics and Technologies for Industrial Automation (ROBOTHIA), pages 1–6, 2025.

10. Zhang, Y., Wang, R., and Chen, X.: Dynamic behavior cloning with temporal feature prediction: Enhancing robotic arm manipulation in moving object tasks. IEEE Robotics and Automation Letters, 10(6):5209–5216, 2025.

CITED LITERATURE (continued)

11. Nguyen, H. H., Vu, M. N., Beck, F., Ebmer, G., Nguyen, A., Kemmetmueller, W., and Kugi, A.: Language-driven closed-loop grasping with model-predictive trajectory optimization. Mechatronics, 109:103335, 2025.

12. Sleiman, J.-P., Farshidian, F., and Hutter, M.: Versatile multicontact planning and control for legged loco-manipulation. Science Robotics, 8(81):eadg5014, 2023.

13. Ferrolho, H., Ivan, V., Merkt, W., et al.: Roloma: Robust loco-manipulation for quadruped robots with arms. Autonomous Robots, 47:1463–1481, 2023. Published: 15 October 2023.

14. Jiang, K., Fu, Z., Guo, J., Zhang, W., and Chen, H.: Learning whole-body loco-manipulation for omni-directional task space pose tracking with a wheeled-quadrupedal-manipulator. IEEE Robotics and Automation Letters, 10(2):1481–1488, 2025.

15. Trossen Robotics: WidowX 250 S – 6-DOF Research Manipulator Arm. https://www.trossenrobotics.com/widowx-250. [Online; accessed June 4, 2025].

16. Trossen Robotics: WidowX 250 S – X-Series Arms Documentation. https://docs.trossenrobotics.com/interbotix_xsarms_docs/specifications/wx250s.html. [Online; accessed June 4, 2025].

17. Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G.: Robotics: Modelling, Planning and Control. Advanced Textbooks in Control and Signal Processing. London, Springer, 2009.

18. Berscheid, L. and Kröger, T.: Jerk-limited real-time trajectory generation with arbitrary target states. Robotics: Science and Systems XVII, 2021.

19. Griffig Robotics GmbH: Ruckig - Motion Generation for Robots and Machines. https://ruckig.com/#about. [Online; accessed March 28, 2025].

20. OptiTrack: Mocap4ROS2 Setup Guide, 2023. [Online; accessed April 16, 2025].

21. Open Robotics: ROS 2 Humble Hawksbill Release (LTS). https://docs.ros.org/en/humble/Releases.html, 2022. [Online; accessed June 14, 2025].

CITED LITERATURE (continued)

22. Open Robotics: ROS 2 Humble: Stability, Performance and Developer Ergonomics. https://www.openrobotics.org/blog/2022/5/24/ros-2-humble-hawksbill-release, 2022. [Online; accessed June 14, 2025].

23. Open Robotics: Unconfigured DDS considered harmful to networks. https://discourse.ros.org/t/unconfigured-dds-considered-harmful-to-networks/25689, 2021. [Online; accessed June 14, 2025].

24. Trossen Robotics: Interbotix X-Series ROS 2 Packages Documentation. https://docs.trossenrobotics.com/interbotix_xsarms_docs, 2024. [Online; accessed June 15, 2025].

25. ros2_control Maintainers: joint_trajectory_controller – User Documentation. https://control.ros.org/master/doc/ros2_controllers/joint_trajectory_controller/doc/userdoc.html, 2025. [Online; accessed June 14, 2025].

26. ros2_control Maintainers: Trajectory Representation in joint_trajectory_controller. https://control.ros.org/master/doc/ros2_controllers/joint_trajectory_controller/doc/trajectory.html, 2025. [Online; accessed June 14, 2025].

27. ROS Maintainers: Understanding trajectory replacement (joint_trajectory_controller). https://wiki.ros.org/joint_trajectory_controller/UnderstandingTrajectoryReplacement, 2025. [Online; accessed June 14, 2025].

28. Trossen Robotics: Interbotix X-Series SDK – ROS 2 Interface Overview. https://github.com/TrossenRobotics/interbotix_xsarms_docs/blob/main/docs/ros_interface/ros2/overview/xs_sdk.rst, 2025. [Online; accessed June 14, 2025].

29. Open Source Robotics Foundation: Gazebo simulation environment. https://classic.gazebosim.org, 2014. [Online; accessed June 15, 2025].

30. NaturalPoint, Inc. (dba OptiTrack): Prime^x 13 Motion Capture Camera. https://www.optitrack.com/cameras/primex-13, 2025. [Online; accessed June 20, 2025].

VITA

NAME	Simone Ughetto
EDUCATION	
	M.S., Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, Illinois, United States of America, Expected July, 2025.
	M.S., Mechatronic Engineering, Politecnico di Torino, Turin, Italy, Expected July, 2025.
	B.S., Mechanical Engineering, Politecnico di Torino, Turin, Italy, July 2023.
LANGUAGE SKILLS	
Italian	Native speaker.
English	Full working proficiency.
	2023 - TOEFL examination (109/120).
	A.Y. 2024/2025 One Year of study abroad in Chicago, Illinois.
	A.Y. 2020/2024. Lectures and exams attended exclusively in English.
SCHOLARSHIPS	
Fall 2024	Italian scholarship for TOP-UIC students.