ENSTA Campus Brest

Internship Report

Control of a Robotic Manipulator to Assist People with Disabilities

Internship carried out at:
LAAS-CNRS

Student: Khaled CHATAH

Tutor: Martin MUJICA

Supervisor: Luc JAULIN

2025 – 2026

Contents

Introduction **1**

1 Overview of the Problem and State of the Art **2**
1.1 Challenges of Assistive Manipulation 2
1.2 Assistive Robotics and Disability 4
1.3 Robotic Arms Used in Assistive Robotics 4
1.4 Methods for Controlling the Robotic Manipulator 5
1.5 Object Detection, Object Pose Detection and 3D Vision 7
1.6 Robotic Grasping 8
1.7 Conclusion 8

2 Developed Approach **10**
2.1 High-Level Pipeline 11
2.2 Assistive Robotic Platform 11
2.3 User Interaction 12
2.3.1 Joystick-Based Supervision 12
2.3.2 Speed and Approach Angle Control 12
2.4 State Machine 13
2.4.1 State Description 13
2.5 Control Architecture 15
2.5.1 Kinematic Model 15
2.5.2 Evolution Function 15
2.5.3 General MPC Formulation 16
2.6 Vision System 24
2.6.1 Object Detection and Segmentation 24
2.6.2 Object Position Estimation 25
2.6.3 Mouth Detection 26

3 Experimental Setup and Results **28**
3.1 Experimental Setup 28
3.1.1 Orthopus Explorer Robotic Arm 29
3.1.2 Joystick Interface 30

3.1.3 Object Selection Using the Joystick . . . 31
3.1.4 Implementation of the Joystick Command Strategies . . . 32
3.1.5 Input Smoothing, Normalization, and Saturation . . . 34
3.1.6 Software Framework and Libraries . . . 36
3.1.7 ROS2 Framework . . . 37
3.2 MPC Parameter Tuning . . . 37
3.2.1 Arbitration Coefficients . . . 38
3.2.2 Internal Weighting Matrices . . . 38
3.2.3 State and Input Bounds . . . 39
3.3 Experimental Results . . . 40
3.3.1 Experiment 1: Influence of the Vertical Alignment Cost . . . 40
3.3.2 Experiment 2: Complete Semi-Autonomous Manipulation Task . . . 42
3.3.3 Demonstration . . . 45
3.4 Discussion . . . 46

4 Conclusion and Perspectives **47**
4.1 Conclusion . . . 47
4.2 Perspectives . . . 47

Abstract

This internship focused on the development of a semi-autonomous control architecture for an assistive robotic manipulator mounted on a wheelchair. The objective was to help users with motor disabilities perform manipulation tasks, such as reaching, grasping, moving, and pouring an object, while keeping the user in the control loop.

The proposed system combines visual perception, user interaction through a joystick, a high-level state machine, and Nonlinear Model Predictive Control. The vision module detects the selected object and estimates its position using YOLO11-seg segmentation and RGB-D data. The user's mouth is detected using MediaPipe in order to generate a target for the drinking assistance phase. The state machine organizes the task into successive phases, while the NMPC controller generates smooth and constrained robot motions according to the current objective.

The control formulation is based on a kinematic model of the robot. The NMPC cost function includes pose tracking, velocity regularization, end-effector alignment, and posture costs. These terms allow the robot to reach the desired target while avoiding abrupt movements and favoring safer configurations around the wheelchair. The user can supervise the robot using the joystick by selecting the object, controlling the progression speed, adjusting the approach direction, and validating key actions.

Experimental tests showed that the proposed approach is functional and suitable for the considered assistive manipulation scenario. The generated motions were smooth, predictable, and easy to supervise. A demonstration was also carried out at ISIR (Institut des Systèmes Intelligents et de Robotique), Sorbonne Université, confirming the relevance of the developed architecture in a real experimental context.

Introduction

Assistive robotics has become an increasingly important field in modern healthcare and intelligent systems. Robotic manipulators can provide valuable assistance to people with motor disabilities by helping them perform daily activities such as grasping objects, moving items, or interacting with their environment. These technologies aim to improve autonomy, quality of life, and accessibility for people in situations of physical impairment.

Despite the significant progress achieved in robotics and artificial intelligence, developing a robotic assistant capable of operating efficiently in real-world environments remains a complex challenge. Such systems must be able to perceive their surroundings, detect and localize objects, estimate object poses in three-dimensional space, and safely control robotic movements in real time. In addition, assistive robotic systems must remain intuitive, robust, and safe for human interaction.

One of the major challenges in assistive manipulation lies in the integration of perception and control. A robotic manipulator must continuously process sensory information from cameras and sensors while generating appropriate commands to interact with dynamic and uncertain environments. This requires the combination of several domains, including computer vision, robotic control, motion planning, and grasping strategies.

The objective of this internship is to study and develop methods for controlling a robotic manipulator dedicated to assistive robotics applications. The work focuses on robotic arm control using Model Predictive Control (MPC), object detection and pose estimation using RGB-D vision systems, and the integration of these modules into a complete robotic architecture based on ROS2 (Robot Operating System).

This report is organized as follows. The first chapter presents the state of the art related to assistive robotic manipulators, robotic control methods, object detection, pose estimation, and robotic grasping. The following chapters describe the architecture of the developed system, the implemented perception and control methods, and the experimental results obtained during the internship.

Chapter 1

Overview of the Problem and State of the Art

Assistive robotic manipulation aims to help people with motor disabilities perform everyday tasks that may otherwise be difficult or impossible to accomplish independently. Such tasks can include object retrieval, object transportation, feeding assistance, drinking assistance, and interaction with the surrounding environment.

Designing a robotic assistant capable of performing these tasks is a challenging problem that involves multiple domains of robotics, including perception, manipulation, motion planning, grasping, and human-robot interaction. In order to successfully complete a task, the robotic system must be capable of understanding its environment, identifying relevant objects, interacting safely with humans, and executing appropriate manipulation actions.

Furthermore, assistive robotic systems operate in environments that are often dynamic and partially unknown. Objects may appear at different locations, obstacles may be present, and the robot must continuously adapt its behavior while maintaining safe operation.

The objective of this work is therefore to study the challenges associated with assistive robotic manipulation and to develop a robotic system capable of helping a user perform daily manipulation tasks while ensuring safety, reliability, and usability.

1.1 Challenges of Assistive Manipulation

Assistive robotic manipulation involves several challenges that must be addressed in order to obtain a system that is safe, reliable, and usable in daily-life situations. The main challenges considered in this work are the following:

- **The Environment Challenge.** The robotic manipulator operates in cluttered environments where multiple obstacles may be present. During task execution, the robot must avoid collisions with the user, the surrounding environment, and

potentially with itself. Since the robot is intended to operate close to the user, and possibly from a wheelchair-mounted configuration, ensuring safe manipulation becomes a critical challenge.

- **The Grasping Challenge.** Objects encountered in daily life can differ significantly in shape, size, weight, texture, and fragility. Consequently, a grasping strategy that is suitable for one object may not necessarily be appropriate for another. The robotic system must therefore be capable of adapting its manipulation strategy to the characteristics of the selected object in order to ensure a reliable grasp and successful task execution.

- **The Perception Challenge.** Reliable manipulation requires accurate perception of the surrounding environment. The robot must be capable of detecting and localizing objects of interest while distinguishing them from other elements present in the scene, such as the table surface, the user, or potential obstacles. Perception errors may directly affect the quality of the manipulation process and can lead to unsuccessful grasps or unsafe robot motions.

- **The Semi-Autonomous Challenge.** The robotic system follows a semi-autonomous paradigm in which the user remains involved in the control loop. Although autonomous functionalities can assist the user during manipulation tasks, the user must retain control over the overall process at all times. The system must therefore balance autonomous decision-making with user supervision, allowing the user to influence, modify, or interrupt the robot behavior whenever necessary.

- **The Stability Challenge.** Assistive robotic manipulation requires the robot to remain stable throughout task execution. Abrupt motions, excessive vibrations, or unstable trajectories may compromise the successful completion of the task and reduce user confidence in the robotic system. Furthermore, many assistive tasks involve transporting objects over relatively long distances while maintaining a desired orientation. In such situations, unstable robot motions may affect the quality of the manipulation process and reduce the overall reliability of the robotic system. Generating smooth and stable trajectories therefore represents an important challenge in assistive robotics.

- **The Human-Like Behavior Challenge.** Beyond safety and task completion requirements, it is desirable for the robotic manipulator to exhibit smooth and human-like motions. Producing more natural arm behaviors can improve user comfort, increase the predictability of the robot's actions, and facilitate human-robot interaction in assistive contexts.

1.2 Assistive Robotics and Disability

Assistive robotics aims to improve the autonomy and quality of life of people with disabilities by helping them accomplish daily tasks such as object manipulation, navigation, and interaction with their environment. Several studies have focused on integrating robotic manipulators with human-centered control interfaces adapted to users with reduced mobility or severe motor impairments.

Muelling et al. proposed a shared-control teleoperation framework for assistive robotic manipulation using Brain-Computer Interfaces (BCIs) [2]. Their system combines computer vision, user intent inference, and autonomous robotic assistance in order to compensate for noisy and low-dimensional BCI commands. The authors introduced adjustable levels of autonomy to preserve the user's feeling of control while improving task success rates. Different user interfaces are discussed in the paper, including intracortical BCIs, joysticks, graphical interfaces, and eye-tracking-inspired intention prediction approaches. The system was validated on manipulation tasks such as grasping, door opening, and liquid pouring.

The work highlights the importance of shared autonomy in assistive robotics, especially for users with severe physical impairments. Instead of relying solely on direct teleoperation, the robot assists the user through autonomous behaviors inferred from the user's intentions. The paper also emphasizes the importance of compliant and safe robot control during human-robot interaction. [2]

1.3 Robotic Arms Used in Assistive Robotics

Different robotic manipulators are used in the literature depending on the targeted application domains. Robotic arms used in assistive robotics can be integrated in different ways depending on the target application. Some manipulators are mounted on fixed platforms or mobile bases, while others are directly attached to a wheelchair in order to remain close to the user and to follow the user in daily environments. When these robots are designed to work in close interaction with humans, they are often referred to as collaborative robots, or cobots. In assistive robotics, this collaborative aspect is important because the robot operates close to the user and interacts with objects in the environment in order to support the user during manipulation tasks.

Muelling et al. used a seven-degree-of-freedom (7-DOF) robotic manipulator as a neuroprosthetic arm controlled through BCI-based teleoperation [2]. The manipulator was designed for assistive tasks and integrated within a shared-control architecture allowing autonomous assistance during object manipulation.

Ciocarlie et al. employed the PR2 robot platform for household object manipulation tasks [3]. The PR2 robot is equipped with two 7-DOF robotic arms and parallel jaw grippers integrating tactile fingertip sensor arrays. The robot also contains several per-

ception sensors including stereo cameras, laser scanners, and an IMU. The platform was specifically designed for autonomous service robotics in indoor environments.

Gebrayel et al. used a Franka Emika Panda 7-DOF manipulator equipped with an eye-in-hand stereo camera for vine pruning applications [4]. The manipulator was used in combination with visual servoing techniques and point cloud alignment algorithms for precise positioning in agricultural environments.

1.4 Methods for Controlling the Robotic Manipulator

Different control strategies are proposed in the reviewed papers depending on the complexity of the environment and the targeted tasks.

Muelling et al. proposed a shared-control teleoperation framework combining user commands with autonomous robotic assistance [2]. Their control architecture integrates:

- user intent inference,
- human-robot arbitration,
- compliant control,
- autonomous motion assistance.

The system blends user commands and autonomous robot actions through an arbitration function. The level of assistance depends on the confidence of the inferred user intention. The paper also presents compliant control methods to ensure safe interactions between the robotic arm and the environment.

Luo et al. proposed an image-based visual servoing (IBVS) grasping framework driven by deep-learning-based keypoint detection [1]. Their method directly computes robot joint velocities from visual feature errors between current and desired image features. The approach combines:

- object detection,
- keypoint extraction,
- visual servoing control.

The controller uses detected image-plane keypoints as feedback features and generates reactive velocity commands without explicit motion planning.

Gebrayel et al. proposed a position-based visual servoing (PBVS) framework combined with Iterative Closest Point (ICP) point-cloud alignment [4]. Several ICP variants were evaluated:

- standard ICP,

- Levenberg-Marquardt ICP,
- Point-to-Plane ICP,
- Symmetric ICP.

The ICP algorithm estimates the relative pose between the current and reference point clouds, while PBVS computes the camera velocity screw required to minimize pose errors. One advantage of this approach is that it does not require explicit feature extraction from the image. Instead of detecting and tracking visual features or keypoints, the method directly uses the geometry of the 3D point clouds. This can make the approach more robust when reliable visual features are difficult to extract. However, this advantage comes at the cost of higher computational complexity, since point cloud registration and pose estimation require processing a large amount of 3D data.

Ciocarlie et al. integrated collision-aware inverse kinematics, motion planning, and tactile feedback control for reliable grasp execution [3]. Their framework combines grasp planning, collision-free trajectory generation, and tactile error correction during manipulation.

Fabio Luz studied the application of MPC to a six-degree-of-freedom assistive robotic manipulator designed for Activities of Daily Living (ADL) assistance such as feeding tasks [5]. The work highlights several limitations of classical PID control approaches for robotic manipulation, particularly their difficulty in handling nonlinear constraints, multi-variable systems, and real-time trajectory adaptation.

The proposed MPC framework uses a dynamic model of the robotic manipulator in order to predict the future behavior of the system and compute optimal control commands over a prediction horizon. The controller generates trajectories allowing the end-effector to reach a desired target position while respecting physical limitations such as joint constraints and velocity bounds.

One important aspect emphasized in the thesis is the ability of MPC to adapt in real time to changing target positions and external perturbations. The simulation results demonstrate that the robotic manipulator can successfully follow fixed, variable, and even circular reference trajectories while maintaining relatively smooth joint motions.

The author also explains that the MPC cost function includes both the end-effector position error and the joint velocities in order to improve motion smoothness and reduce abrupt robot movements.

Another advantage discussed in the work is the capability of MPC to explicitly integrate system constraints into the optimization problem, including:

- joint position limits,
- control input bounds,
- trajectory smoothness constraints.

The thesis concludes that MPC represents a powerful approach for assistive robotic manipulation because it allows smooth and adaptive trajectory generation while considering the physical limitations of the robotic system.

1.5 Object Detection, Object Pose Detection and 3D Vision

Several perception approaches are presented in the reviewed works for object detection, pose estimation, and 3D scene understanding.

Luo et al. proposed a perception pipeline combining object detection and keypoint detection for robotic grasping [1]. The framework uses:

- **RTMDet** for object detection. RTMDet is used to locate the target object in the image by predicting its bounding box. This first step identifies where the object is located in the camera image.
- **RTMPose4VSG** for keypoint detection. RTMPose4VSG is used to estimate specific visual keypoints on the detected object. These keypoints are then used by the visual servoing controller to guide the robot motion during grasping.

The method performs real-time inference of image-plane keypoints used for visual servoing. Their system was designed to remain robust under low lighting, occlusions, overexposure, and cluttered environments. The authors also used domain randomization to generate a synthetic dataset for training.

Ciocarlie et al. used 3D point cloud perception from stereo cameras and laser range sensors [3]. Their perception pipeline includes:

- planar surface extraction,
- Euclidean clustering,
- object segmentation,
- ICP-based object recognition,

The system combines semantic perception with 3D occupancy mapping in order to support safe grasp planning and motion planning.

Muelling et al. used depth-image template matching and ICP registration for object localization [2]. Their perception module removes the robot arm from the depth image, segments potential objects, and compares segmented depth images against pre-generated templates of known objects. ICP refinement is then used for final pose estimation.

Gebrayel et al. focused on point-cloud alignment for visual servoing in agricultural robotics [4]. Their method uses stereo vision and ICP registration to estimate the relative pose between current and reference vine point clouds. The authors emphasize the challenges caused by irregular vine geometries, occlusions, and varying lighting conditions.

1.6 Robotic Grasping

Different robotic grasping approaches are discussed in the reviewed papers.

Luo et al. proposed a grasping framework based on deep-learning keypoint detection and image-based visual servoing [1]. Instead of relying on explicit hand-eye calibration and motion planning, the robot continuously tracks visual keypoints and generates velocity commands for reactive grasping. Their method achieved successful grasping in dynamic environments and under partial occlusions. However, one limitation of this approach is its dependence on a dataset used to train the detection and keypoint estimation models. Since the grasping strategy relies on learned keypoints, the robot may not be able to grasp a completely random object that was not represented in the training data. This limits the generalization ability of the method when facing unknown objects in an unconstrained environment.

Ciocarlie et al. developed a complete grasping architecture for both known and unknown household objects [3]. For known objects, grasp poses were precomputed using the GraspIt! simulator and stored in a database. For unknown objects, grasps were generated online using heuristics based on object geometry and principal axes. Their framework also integrates tactile sensing for grasp failure detection and correction during execution.

Muelling et al. introduced grasp inference using capture envelopes within a shared-control framework [2]. Their system estimates the intended grasp pose of the user based on the robot end-effector trajectory and predefined grasp sets associated with known object models. The framework combines intent prediction with autonomous grasp assistance.

The reviewed papers demonstrate that robotic grasping increasingly relies on the integration of perception, motion planning, visual servoing, tactile sensing, and learning-based approaches in order to achieve robust performance in unstructured environments.

1.7 Conclusion

The study of assistive robotics highlights the importance of developing robotic systems capable of helping people with motor disabilities perform daily tasks more autonomously. Semi-autonomous robotic manipulators represent a promising solution since they combine human guidance with intelligent robotic assistance.

The system considered in this project relies on the Orthopus Explorer robotic arm, which can be mounted on a wheelchair and controlled through a joystick and a smart visual interface. This shared-control approach allows the user to remain involved in the manipulation process while benefiting from robotic assistance for perception and motion execution.

From a control perspective, Model Predictive Control (MPC) appears to be particularly suitable for assistive robotic manipulation. MPC enables the robot to generate smooth and safe motions while respecting physical and environmental constraints. In this

project, the controller aims to guide the robotic arm toward a desired target pose while maintaining the target object inside the camera field of view to improve grasping reliability. A lower-level PID controller is also considered for simpler tasks such as initializing the robotic arm to a home position.

Regarding perception, modern object detection and 3D vision techniques play a central role in robotic manipulation. The use of YOLO segmentation combined with RGB-D vision (RealSense D435 camera) allows more precise object localization by extracting segmentation masks and estimating object centroids in three-dimensional space. These visual perception methods contribute to improving end-effector positioning and increasing the reliability of grasping tasks.

Overall, the state of the art demonstrates that the combination of semi-autonomous interaction, advanced robotic control, and modern computer vision techniques provides a strong foundation for the development of intelligent assistive robotic systems.

Chapter 2

Developed Approach

This chapter presents the complete approach developed during the internship. The objective is to describe how the assistive robotic system combines perception, user supervision, task-level decision making, and NMPC-based control to perform a semi-autonomous manipulation task.

The system is composed of a robotic manipulator placed beside a wheelchair, an eye-in-hand RGB-D camera, a joystick interface, a perception module based on object detection and segmentation, a state machine, and a nonlinear model predictive controller. These components do not operate independently. Instead, they continuously interact with each other in order to generate a motion adapted to the current task phase, the user commands, and the perceived environment.

The sensors and input devices provide information to the system. The RGB-D camera observes the scene and provides visual information used to detect and localize objects. The joystick provides user commands that modify the behavior of the robot. In particular, the joystick input is used to change the speed modulation parameter of the robot motion, to modify the approach direction around the object, and to validate key actions during the task.

The state machine acts as the high-level supervisor of the system. It determines the current phase of the manipulation task and generates the corresponding references and activation coefficients for the NMPC controller. These coefficients define which cost terms are active at each state, such as position tracking, orientation alignment, posture regulation, or velocity regularization. Therefore, the MPC behavior changes depending on the current state of the task.

The NMPC controller then uses the current robot state, the task-dependent reference, and the activation coefficients generated by the state machine to compute the joint velocity command. This command is sent to the robot through the internal controller, which executes the motion.

2.1 High-Level Pipeline

The developed system follows a semi-autonomous manipulation pipeline. The robot does not execute a fully autonomous task from start to finish. Instead, the user provides high-level commands through the joystick, while the robot automatically computes the motions required to execute each phase of the task.

The high-level pipeline can be summarized as follows:

1. The RGB-D camera observes the scene from the robot end-effector.
2. The perception system detects and segments the objects present in the image.
3. The detected objects are displayed to the user through the visual interface.
4. The user selects the target object using the joystick.
5. The object position is estimated in 3D from the segmentation mask and the depth image.
6. The joystick provides user commands to modulate the robot behavior, such as the speed modulation coefficient and the approach direction.
7. The state machine defines the current phase of the manipulation task.
8. According to the current state, the state machine generates the desired reference and the activation coefficients of the NMPC cost function.
9. The NMPC controller computes the joint velocity command using the current robot state, the desired reference, and the state-dependent coefficients.
10. The internal controller applies the generated command to the robot.
11. The robot moves toward the target while the user supervises the motion.
12. The user validates the grasping action.
13. The robot lifts the object, moves toward the user, executes the drinking assistance phase, and puts the object back.

2.2 Assistive Robotic Platform

The robot is intended to be mounted beside a wheelchair in order to assist a user during manipulation tasks. During the experimental phase, the robot was tested on a fixed support, but the considered use case remains a wheelchair-mounted assistive manipulation scenario.

The Orthopus Explorer is a 6-DOF robotic manipulator equipped with a two-finger gripper. The arm is responsible for reaching, grasping, transporting, pouring, and putting back the selected object. The camera is mounted close to the end-effector, which gives the system an eye-in-hand perception configuration. The robot is mounted close to the user.

The use of an eye-in-hand camera allows the perception system to observe the region close to the gripper. This is useful for object detection and position estimation, especially during the reaching and grasping phases.

2.3 User Interaction

The system follows a semi-autonomous approach in which the user remains responsible for supervising the task. The joystick is not used to directly control all robot joints. Instead, it provides high-level commands that influence the behavior of the autonomous controller.

2.3.1 Joystick-Based Supervision

The user supervises the robot using a 2D joystick and one button to validate the robot's action. The joystick allows the user to select the object, control the robot's progression speed, modify the approach angle, and validate the grasping command.

The joystick is used to:

- select the target object;
- control the progression speed of the robot;
- adjust the approach angle around the object;
- validate the grasping command.

2.3.2 Speed and Approach Angle Control

During the task, the user controls the progression speed of the robot motion. The trajectory is computed by the controller, but the user determines how fast the robot progresses. If no speed command is provided, the robot does not continue moving in the current phase.

The joystick is also used to adjust the approach angle around the selected object. This allows the user to choose a suitable grasping direction without directly controlling the full end-effector trajectory.

2.4 State Machine

The high-level behavior of the system is organized through a state machine. The state machine defines the sequence of phases required to complete the manipulation task, from object selection to object return.

The full state machine is shown in Figure 2.1.

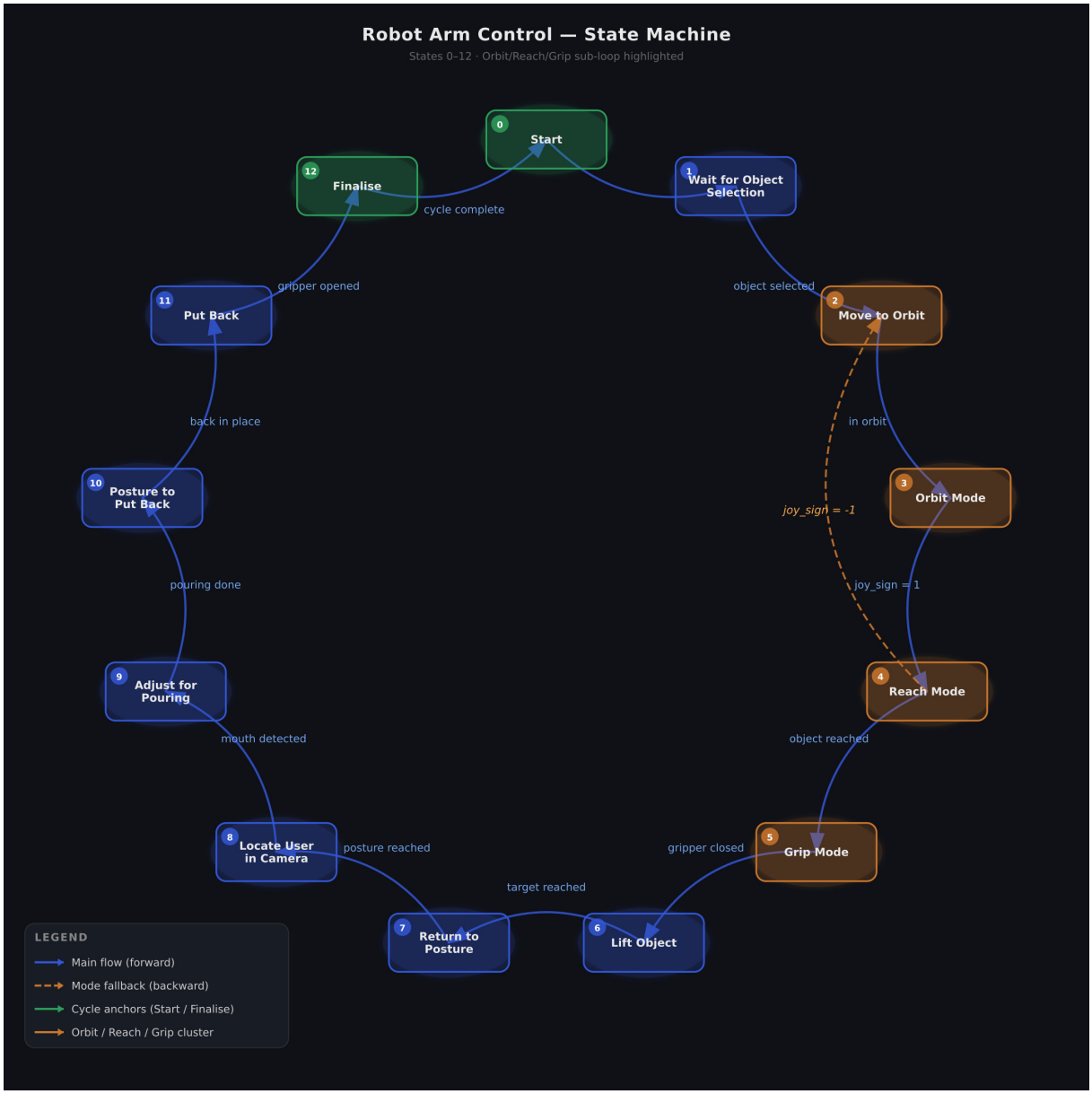


Figure 2.1: State machine of the semi-autonomous manipulation task

2.4.1 State Description

It is crucial for this type of handling that the object remains stable during the manipulation process. Indeed, from the *Reach Mode* until the object is put back, the end-effector orientation is controlled so that the gripper remains horizontal. This constraint

helps maintain the object in a stable configuration during the approach, grasping, lifting, transport, pouring, and return phases.

Although the task is organized as a sequence of states, the system should not be interpreted as fully autonomous. The state machine defines the order of the manipulation phases, while the execution of some actions remains dependent on the user. In particular, the user selects the target object, controls the progression speed of the robot motion, adjusts the approach angle before reaching the object, and validates the grasping action.

A detailed explanation of the states is presented in the following section:

- **Start:** initializes the system variables, resets the object and mouth poses, and prepares the system for a new manipulation cycle.
- **Wait for Object Selection:** the system waits until the user selects an object from the detected objects displayed on the visual interface. The user is also able to change the field of view of the robot by rotating it around its base link. Once an object is selected, the system transitions to the next state.
- **Move to Orbit:** the robot moves toward an initial configuration around the selected object. This state allows the robot to reach a suitable position before entering the orbiting behavior.
- **Orbit Mode:** the user can control the robot motion around the selected object in order to choose a suitable approach direction. This phase gives the user control over the relative positioning of the end-effector with respect to the object.
- **Reach Mode:** once the approach direction is selected, the robot moves the end-effector toward the selected object. During this phase, the selected object is locked so that the target remains fixed.
- **Grip Mode:** when the object is reached, the robot enters the grasping phase. The gripper is controlled to close around the object, and the transition to the next state occurs once the grasp is validated.
- **Lift Object:** after grasping, the robot lifts the object from its initial position. This reduces the risk of collision with the table or surrounding objects.
- **Intermediate Posture Back:** the robot moves to an intermediate posture after lifting the object. This state prepares the robot for the next phase of the task and avoids direct abrupt motions.
- **User in Field of View:** the robot adjusts its posture so that the user becomes visible to the camera. This step is required before detecting the user's mouth.
- **Adjust Posture for Pouring:** once the mouth is detected, the robot adjusts the end-effector pose to reach a suitable pouring configuration. The user can interrupt this phase and request the robot to return the object.

- **Intermediate Posture Put Back:** before putting the object back, the robot moves through an intermediate posture. This avoids sudden transitions and helps maintain a safer trajectory.
- **Put Back:** the robot moves the object back toward its original position and opens the gripper once the placement pose is reached.
- **Finalise:** the system completes the manipulation cycle, returns to a safe configuration, and resets the state machine to the initial state.

2.5 Control Architecture

The control architecture is responsible for generating the robot motion according to the current task state and the desired target pose. The selected control strategy is Nonlinear Model Predictive Control.

2.5.1 Kinematic Model

The robot is controlled using a kinematic model. The state of the system is the joint configuration:

$$x_k = q_k,$$

and the control input is the joint velocity command:

$$u_k = \dot{q}_k.$$

The end-effector pose is obtained through forward kinematics:

$$T_{ee}(q) = \begin{bmatrix} R_{ee}(q) & p_{ee}(q) \\ 0 & 1 \end{bmatrix},$$

where $p_{ee}(q)$ is the end-effector position and $R_{ee}(q)$ is the end-effector orientation.

2.5.2 Evolution Function

The continuous-time kinematic model is:

$$\dot{x}(t) = f(x(t), u(t)) = u(t).$$

With a constant input over one sampling interval, the discrete-time evolution becomes:

$$x_{k+1} = x_k + \Delta t u_k.$$

Since $x = q$, this gives:

$$q_{k+1} = q_k + \Delta t \dot{q}_k.$$

This model is used by the NMPC solver to predict the future joint configurations of the robot over the prediction horizon.

2.5.3 General MPC Formulation

Model Predictive Control is based on solving an optimization problem repeatedly over a finite prediction horizon. At each control iteration k, the controller predicts the future evolution of the system and computes a sequence of future control inputs:

$$U_k = [u_k, u_{k+1}, \ldots, u_{k+N-1}],$$

where N is the prediction horizon.

The objective of the controller is to minimize a cost function $L(k)$. This cost function is generally composed of several weighted terms, each one corresponding to a desired behavior of the system, such as target tracking, control effort minimization, smoothness, or constraint satisfaction. A terminal cost can also be added to penalize the predicted final state at the end of the horizon.

The optimization problem can be written as:

$$U_k^\star = \arg\min_{U_k} L(k),$$

with:

$$L(k) = \sum_{i=0}^{N-1} \left(\sum_{j=1}^{m} w_j J_j(x_{k+i}, u_{k+i}) \right) + J_f(x_{k+N}),$$

where J_j represents the j-th cost term, w_j is its associated weight, and J_f is the terminal cost.

The minimization is performed under the system evolution model:

$$x_{k+i+1} = f(x_{k+i}, u_{k+i}), \qquad i = 0, \ldots, N-1,$$

and under possible state and input constraints:

$$x_{\min} \leq x_{k+i} \leq x_{\max},$$

$$u_{\min} \leq u_{k+i} \leq u_{\max}.$$

The minimization is also subject to physical or spatial constraints:

$$g_r(x_{k+i}, u_{k+i}) \leq 0, \qquad r = 1, \ldots, m.$$

where m is the number of constraint functions.

These functions define the admissible region of the optimization problem. They can represent physical constraints, such as joint limits and velocity limits, or task-specific constraints, such as obstacle avoidance, field-of-view requirements, or safety conditions.

After the optimization, only the first command $u_k^\star$ is applied to the system. At the next iteration, the state is measured again, the prediction horizon is shifted forward, and a new optimization problem is solved. This principle is known as the receding horizon strategy.

Figure 2.2 illustrates the basic principle of Model Predictive Control.

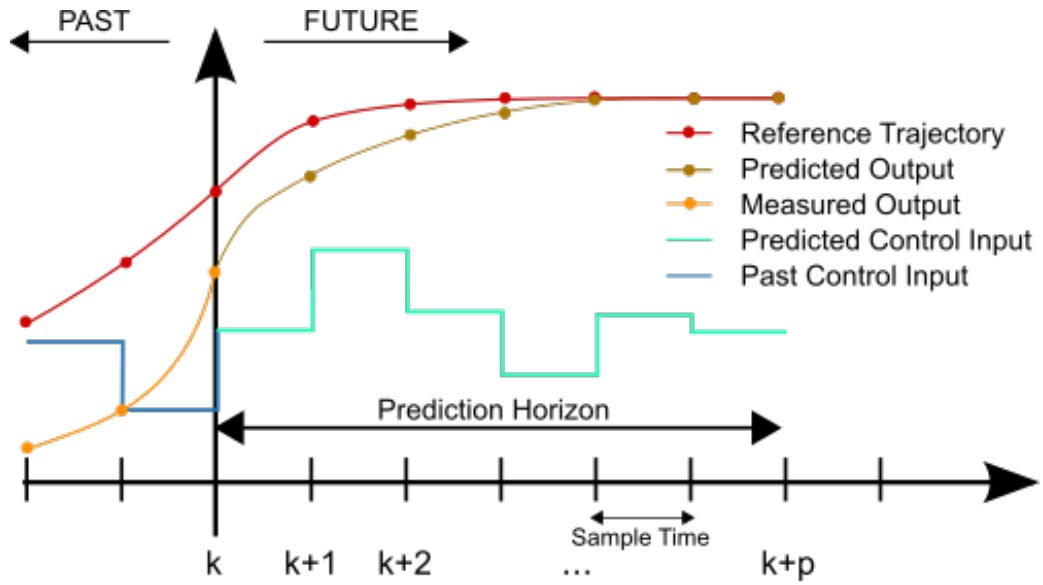


Figure 2.2: Basic principle of Model Predictive Control, adapted from Wikimedia Commons [9].

This strategy allows MPC to continuously update the control action according to the current state of the system. It is particularly useful when the reference changes over time or when constraints must be explicitly taken into account.

Cost Function Structure

At each control iteration, the NMPC controller computes the optimal command by minimizing a cost function over the prediction horizon. The cost function is composed of several objectives, each one associated with a specific behavior expected from the robot.

The global cost can be written as:

$$L(k) = \sum_{i=0}^{N-1} \left(J_{\text{pose}}(x_{k+i}, u_{k+i}) + J_{\text{vel}}(u_{k+i}) + J_{\text{align}}(x_{k+i}) + J_{\text{posture}}(x_{k+i}) \right) + J_f(x_{k+N}),$$

where:

- J_{pose} penalizes the error between the current end-effector pose and the desired reference pose;
- J_{vel} penalizes large control inputs in order to smooth the motion;
- J_{align} maintains a suitable end-effector direction during manipulation;
- J_{posture} guides the robot toward preferred intermediate postures;
- J_f represents the terminal cost at the end of the prediction horizon.

In the implemented formulation, no terminal cost is used. Therefore:

$$J_f(x_{k+N}) = 0.$$

Pose Tracking Cost

The pose tracking cost is responsible for driving the end-effector toward a desired pose. The current end-effector pose is obtained through the forward kinematics:

$$T_{ee}(x_k) = \begin{bmatrix} R_{ee}(x_k) & p_{ee}(x_k) \\ 0 & 1 \end{bmatrix}.$$

The desired pose is written as:

$$T_{\text{ref}}(s_k) = \begin{bmatrix} R_{\text{ref}}(s_k) & p_{\text{ref}}(s_k) \\ 0 & 1 \end{bmatrix},$$

where s_k denotes the current state of the state machine.

The position error is defined as:

$$e_p(k) = p_{ee}(x_k) - p_{\text{ref}}(s_k).$$

The orientation error is computed from the rotation error:

$$e_R(k) = \log\left(R_{ee}(x_k) R_{\text{ref}}(s_k)^T\right),$$

where $\log(\cdot)$ denotes the logarithmic map on $SO(3)$.

The complete end-effector tracking error is then written as:

$$e_{ee}(k) = \begin{bmatrix} W_p e_p(k) \\ W_R(s_k) e_R(k) \end{bmatrix}.$$

The pose tracking cost is expressed as:

$$J_{\text{pose}}(x_k, u_k) = \beta_1(s_k) \left\| e_{ee}(k) \right\|^2.$$

The coefficient $\beta_1(s_k)$ activates or reduces the influence of the pose tracking objective depending on the current phase of the task.

Velocity Regularization Cost

The velocity regularization cost penalizes large joint velocity commands:

$$J_{\text{vel}}(u_k) = \alpha_2(s_k) \|u_k\|^2 .$$

Since the control input corresponds to joint velocities, this term helps avoid aggressive commands and contributes to smoother robot motion, which is important in an assistive context where the manipulator operates close to the user.

Vertical Alignment Cost

During manipulation, the orientation of the end-effector must remain suitable for the task. In particular, after the object has been approached and grasped, the end-effector should maintain a stable orientation to avoid undesirable object motion.

The vertical alignment cost penalizes the deviation between the current end-effector z-direction vector e_{ez} and the desired vertical direction vector e_z:

$$J_{\text{align}}(x_k) = \alpha_3(s_k) \|e_{ez} - e_z\|^2 .$$

The activation coefficient $\alpha_3(s_k)$ determines whether this objective is active depending on the current state of the state machine. This term is especially important from the reaching phase until the object is put back, because the gripper must remain approximately horizontal to keep the object stable.

Posture Regularization Cost

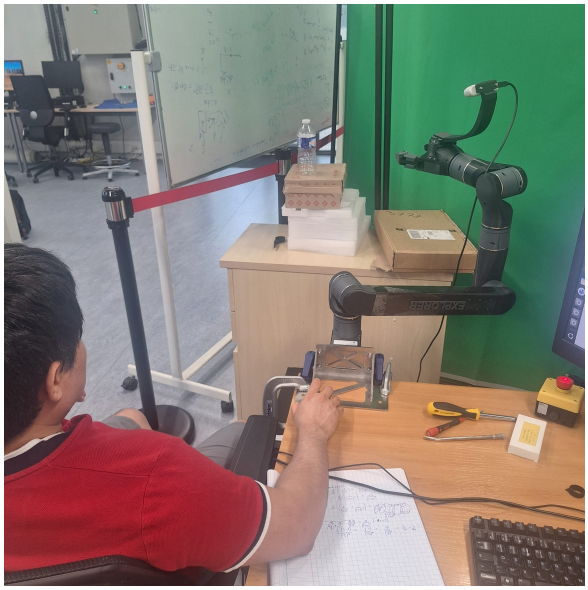
In some phases of the task, reaching a Cartesian position target is not sufficient to define a unique robot behavior. The joint configuration of the robot belongs to a six-dimensional configuration space. This means that the robot configuration is described by several angular joint variables, while the position tracking objective only constrains the three-dimensional position of the end-effector. Moreover, full pose tracking is not active in all states of the task. As a result, the NMPC can find several joint configurations that satisfy the same Cartesian position objective.

Therefore, without an additional regularization term, the optimizer may choose a configuration that reaches the desired position but is not necessarily the most suitable one for the assistive task. For example, two different joint configurations may place the gripper at the same target position, while one configuration keeps the arm farther from the user and the wheelchair, and another one brings the robot closer to them. In this case, both configurations may be valid from the point of view of position tracking, but

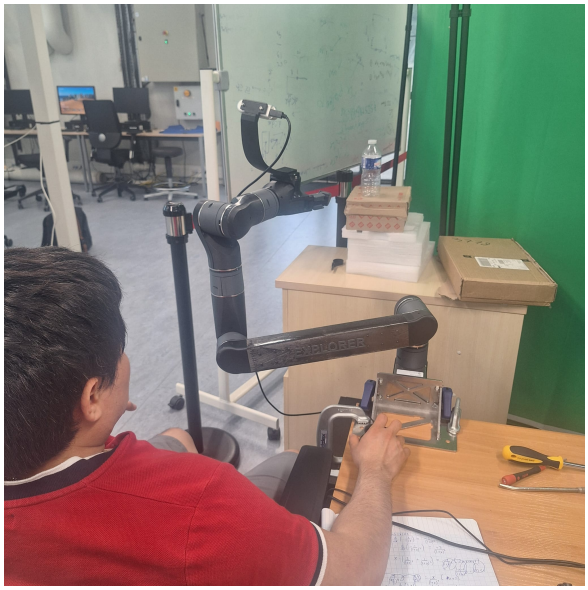
only one of them is preferable from a safety and usability point of view. As a result, the NMPC may converge toward different configurations depending on the initial joint state and on the cost terms used in the optimization.

For example, when the selected object is directly in front of the end-effector, the robot may reach it using configurations that either bring the arm closer to the wheelchair or keep it away from the user. Since collision avoidance is a major requirement, the second type of configuration is preferred.

Posture costs are therefore introduced to guide the optimizer toward safer and more physically acceptable joint configurations. These posture references are not strict targets that the robot must reach exactly. Instead, they act as soft guides inside the NMPC cost function. Their role is to make the optimizer prefer trajectories that pass close to the desired posture when several solutions are possible. It should be noted that configurations corresponding to a desired posture should be chosen so that the joints are far from their limits. This is crucial for the MPC so that it does not fall into a local minimum.



(a) Posture outside the chair safety envelope



(b) Posture entering the chair safety envelope

Figure 2.3: Example of two robot postures illustrating the importance of posture regularization

Let x_{nom}, x_{mouth}, x_{pouring}, and x_{goback} denote reference postures used during different phases of the task. The posture cost can be written as:

$$
\begin{aligned}
J_{\text{posture}}(x_k) = \alpha_4(s_k)\,(x_k - x_{\text{nom}})^T\, W_x\,(x_k - x_{\text{nom}}) \\
+\alpha_5(s_k)\,(x_k - x_{\text{mouth}})^T\, W_x\,(x_k - x_{\text{mouth}}) \\
+\beta_2(s_k)\alpha_6(s_k)\,(x_k - x_{\text{pouring}})^T\, W_x\,(x_k - x_{\text{pouring}}) \\
+\alpha_7(s_k)\,(x_k - x_{\text{goback}})^T\, W_x\,(x_k - x_{\text{goback}})\,.
\end{aligned}
$$

The activation coefficients α_4, α_5, α_6, and α_7 determine which posture objective is active for each state of the manipulation process. The factor $\beta_2(s_k)$ modulates the influence of the pouring posture term when required.

Reference Position and Orientation Generation

The state machine acts as a high-level supervisor for the NMPC controller. At each time step k, the current state s_k defines the reference position p_{ref} and the reference orientation R_{ref} used in the NMPC cost function.

The desired reference position is defined as follows:

$$p_{\text{ref}}(s_k) = \begin{cases} p_{\text{orbit},0}, & \text{if } s_k = 2, \\ p_{\text{orbit}}(\theta_{\text{user}}), & \text{if } s_k = 3, \\ p_{\text{obj}}, & \text{if } s_k = 4 \text{ or } s_k = 5, \\ p_{\text{obj}} + 0.10 e_z, & \text{if } s_k = 6, \\ \text{not considered}, & \text{if } s_k = 7 \text{ or } s_k = 8, \\ p_{\text{mouth}} + \varepsilon e_z, & \text{if } s_k = 9, \\ \text{not considered}, & \text{if } s_k = 10, \\ p_{\text{obj}} + 0.10\,(1 - \exp(-8 d_{\text{obj}}))\, e_z, & \text{if } s_k = 11, \\ \text{not considered}, & \text{if } s_k \in \{0, 1, 12\}. \end{cases}$$

In state $s_k = 11$, the term:

$$p_{\text{obj}} + 0.10\,(1 - \exp(-8 d_{\text{obj}}))\, e_z$$

is used to mimic the behavior of a human putting an object down. The robot must return the object to its initial position. Therefore, the desired position is based on the original object position p_{obj}, with an additional vertical offset. This offset decreases exponentially as the robot gets closer to the object position. As a result, the robot first keeps the object slightly above the target location, then progressively lowers it when approaching the final placement point. This generates a safer and more natural put-back trajectory.

During the object approach and grasping phases, the reference orientation is generated from a set of orthonormal vectors attached to the object.

In state $s_k = 4$, the first axis is chosen as the normalized vector pointing from the current orbit position toward the object:

$$e_x = \frac{p_{\text{obj}} - p_{\text{orbit}}}{\|p_{\text{obj}} - p_{\text{orbit}}\|}.$$

The second axis is chosen such that:

$$e_y = e_{z,\text{world}} \times e_x.$$

Finally, the third axis is obtained from the cross product:

$$e_z = e_x \times e_y.$$

At orbit, e_z is expected to be parallel to the world vertical axis. Since both e_x and e_y belong to the horizontal plane, the resulting vector e_z coincides with the world vertical axis:

$$e_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

The desired orientation matrix is therefore:

$$R_{\text{ref}} = \begin{bmatrix} e_x & e_y & e_z \end{bmatrix}.$$

This construction allows the end-effector orientation to remain consistent with the selected approach direction while maintaining a suitable orientation for grasping and manipulation.

Main cost weights. The first parameters to tune are the main scalar weights of the NMPC cost function. These weights define the global priority of each objective inside the optimization problem. They strongly influence the robot behavior in terms of convergence speed, smoothness, stability, and safety.

The four main weights are:

$$w_{\text{pose}}, \qquad w_{\text{vel}}, \qquad w_{\text{align}}, \qquad w_{\text{posture}}.$$

Pose tracking weight w_{pose}. The coefficient w_{pose} gives priority to the Cartesian pose tracking objective. This term is responsible for the servoing behavior of the end-effector toward the desired pose.

This parameter has a direct influence on the response time of the robot. If w_{pose} is too low, the robot gives little importance to the target pose. The motion becomes slow and the convergence toward the reference may be weak. Increasing this value makes the robot react faster and track the desired pose more strongly.

However, if w_{pose} is too high, the controller may generate aggressive commands. This can produce oscillations, vibrations, or even divergence. Therefore, this weight must be tuned first in order to obtain a sufficiently fast response without losing stability.

Velocity regularization weight w_{vel}. The coefficient w_{vel} penalizes the joint velocity command. Since the control input of the NMPC is the joint velocity vector, this term acts as a regularization on the robot speed:

$$J_{\text{vel}} = w_{\text{vel}} \|u_k\|^2 .$$

Experimentally, this coefficient reduces overshoot and makes the robot motion smoother. It limits abrupt changes in the velocity command and prevents jagged or discontinuous movements.

This parameter is tuned after w_{pose}. First, the pose tracking weight is adjusted to obtain a sufficiently reactive motion. Then, w_{vel} is increased to smooth the response and improve convergence. Ideally, the robot behavior should be close to a first-order convergence, with a progressive and non-oscillatory motion toward the target.

Alignment weight w_{align}. The coefficient w_{align} defines the priority given to the end-effector alignment objective. In this work, this objective is mainly used to keep the gripper approximately horizontal during manipulation.

This is important after the object has been reached and grasped, because an unsuitable gripper orientation could make the object unstable during lifting, transport, pouring, or return. A low value may allow the robot to reach the Cartesian target while losing the desired gripper orientation. A higher value forces the NMPC to preserve the horizontal orientation more strongly.

This parameter is tuned after the velocity regularization weight, once the basic position tracking behavior is stable and smooth.

Posture weight w_{posture}. The coefficient w_{posture} defines the importance of the posture regularization objective. This term guides the robot toward preferred joint configurations.

Some of these configurations may bring the robot links close to the wheelchair or the user, while others keep the arm in a safer region. The posture weight helps the optimizer choose configurations that are more compatible with safety, collision avoidance, and physical feasibility.

If w_{posture} is too low, the optimizer mainly follows the Cartesian objective and may choose undesirable joint configurations. If it is too high, the robot may prioritize the posture objective too much and reduce the quality of the Cartesian tracking.

Control Block Diagram

Figure 2.4 summarizes the complete control architecture used in the NMPC formulation.

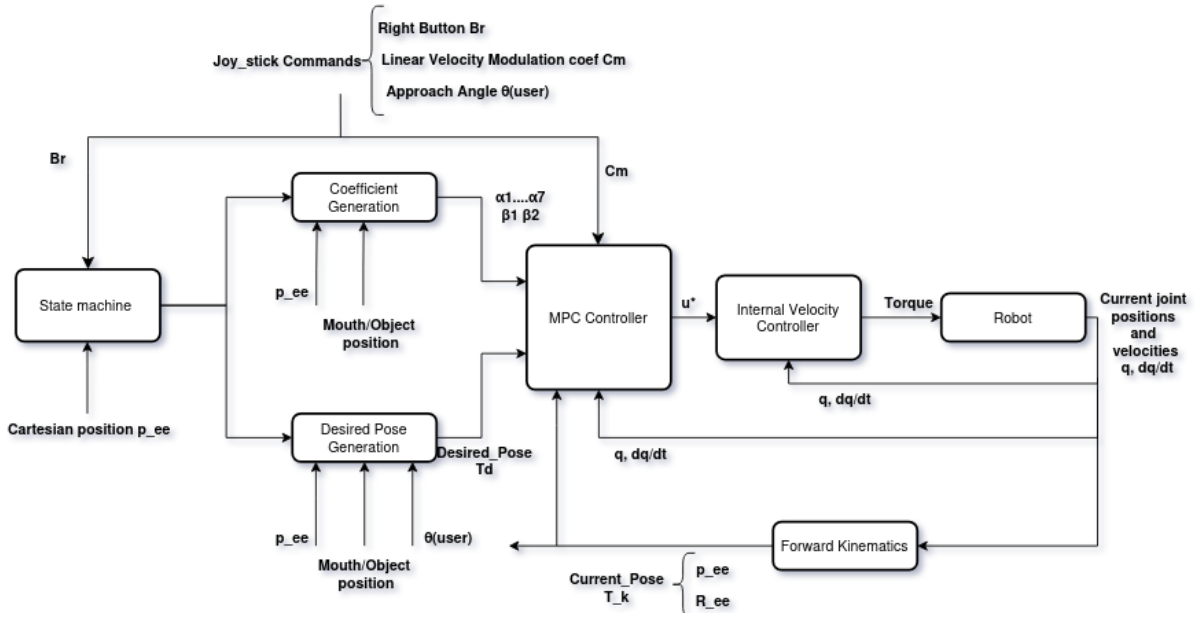


Figure 2.4: Control block diagram of the NMPC architecture

2.6 Vision System

The vision system is a central part of the developed approach. It provides the information required by the state machine and the NMPC controller to generate the robot motion. The system uses an eye-in-hand RGB-D camera mounted close to the robot end-effector. This configuration allows the robot to observe the workspace from the point of view of the gripper.

The vision pipeline has three main objectives:

- detecting the objects present in the scene;
- estimating the three-dimensional position of the selected object;
- detecting the user's mouth for the pouring phase.

The output of the vision system is used by the rest of the pipeline to update the selected object position, generate the desired pose, and decide when the robot can move toward the pouring configuration.

2.6.1 Object Detection and Segmentation

Object detection is performed using YOLO11-seg [10]. This model provides both object detection and instance segmentation. The detection part returns a bounding box and a class label for each detected object, while the segmentation part returns a pixel-level mask corresponding to the object.

In the developed system, the bounding box is used mainly for object identification and display in the user interface. The segmentation mask is then used to isolate the object more precisely before estimating its 3D position.

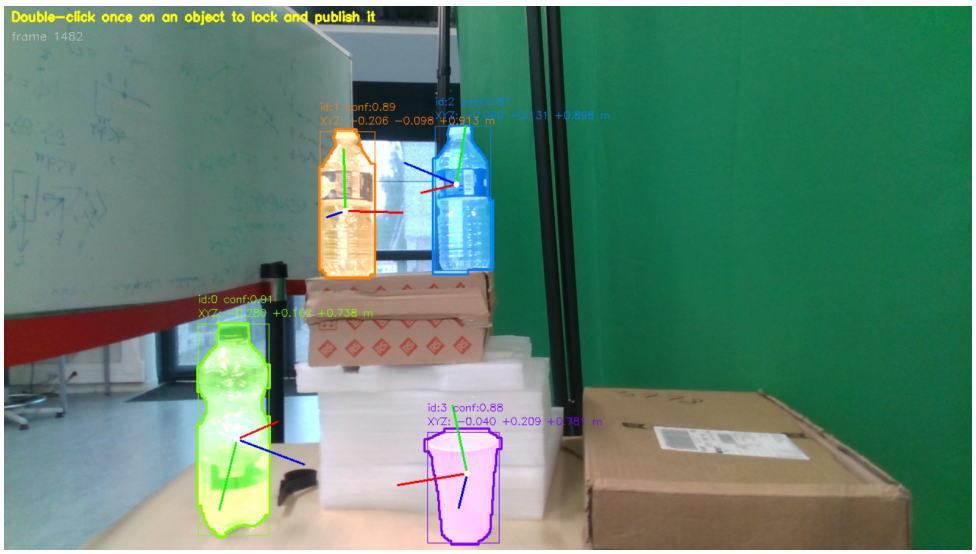


Figure 2.5: Example of object detection and segmentation using YOLO11-seg

The segmentation output is important because the bounding box alone does not accurately represent the real shape of the object. A bounding box may include background pixels, table pixels, or parts of other objects. Therefore, using the segmentation mask improves the quality of the 3D position estimation.

2.6.2 Object Position Estimation

Once the user selects an object, its three-dimensional position is estimated using the RGB-D information. The estimation is based on the segmentation mask provided by YOLO11-seg and the depth image provided by the RGB-D camera.

The estimation procedure is the following:

1. the segmentation mask is used to keep only the pixels belonging to the selected object;
2. the corresponding 3D point cloud is generated from the depth image;
3. the point cloud is sorted according to the distance from the camera;
4. the closest 10% and the farthest 10% of the points are removed;
5. the centroid is computed as the average of the remaining 80% of the points.

The estimated object position is therefore:

$$p_{\text{obj}} = \frac{1}{N_f} \sum_{i=1}^{N_f} p_i,$$

where p_i are the filtered 3D points and N_f is the number of points kept after filtering.

Using the segmentation mask is necessary because the center of the bounding box does not necessarily correspond to the real object centroid. Moreover, using all the points inside the bounding box can introduce large errors, since the bounding box may contain background points. Experimentally, this problem was significant: an object located at approximately 60 cm from the camera could be mixed with background points located several meters away.

The filtering step reduces the influence of outliers and remaining background points. It also helps when two objects of the same type are aligned in the camera direction, because the farther object should not influence the centroid estimation of the selected object.

2.6.3 Mouth Detection

During the drinking assistance phase, the robot must estimate the position of the user's mouth. For this purpose, MediaPipe Face Landmarker is used [11]. This tool detects facial landmarks in the camera image, including landmarks located around the mouth.

The detected landmarks are used to estimate the mouth position and orientation. This information is then sent to the state machine and to the NMPC controller in order to generate a suitable target pose for the pouring task.

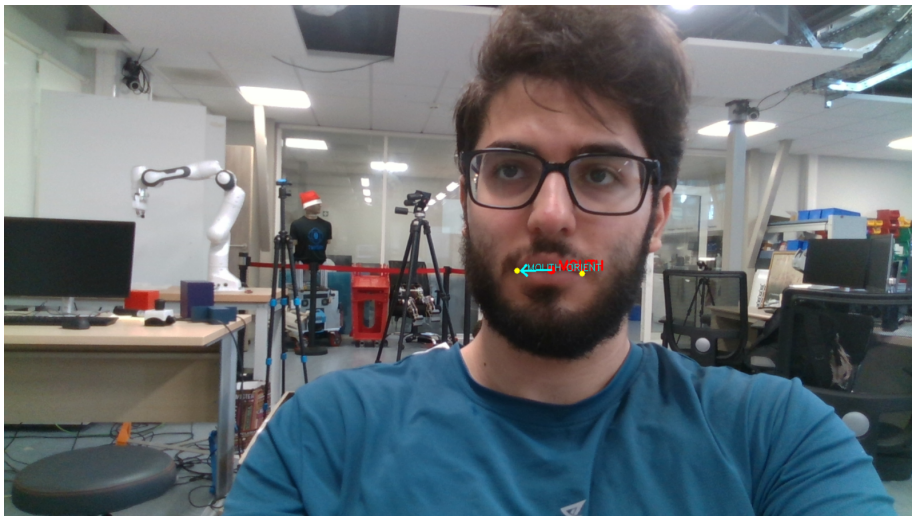


Figure 2.6: Example of mouth detection using MediaPipe

The mouth detection step is activated only when the robot reaches the appropriate phase of the task. Before this phase, the robot first moves to an intermediate posture that allows the user to appear in the camera field of view. However, a problem may occur if several people are detected in the scene during this intermediate posture. In this case, the system could detect several mouth positions, although only the user's mouth should be considered as a valid target.

To reduce this ambiguity, mouth detections that are not located inside a predefined 3D box are rejected. This box is defined with respect to the robot base frame. Since

the robot is intended to be attached close to the user, the user's mouth is expected to be located in a specific region of the robot workspace. In the implemented system, the accepted region is defined as:

$$x \in [-1, 0] \text{ m}, \qquad y \in [-1, 0] \text{ m}, \qquad z \in [0.1, 0.5] \text{ m}.$$

Only mouth positions located inside this volume are accepted and sent to the state machine. This filtering step helps reject false detections or detections corresponding to other people in the environment.

Chapter 3

Experimental Setup and Results

This chapter presents the experimental setup used to validate the developed approach and the results obtained during the manipulation task.

3.1 Experimental Setup

The experimental setup is composed of the Orthopus Explorer robotic manipulator, an Intel RealSense D435 RGB-D camera, a 3Dconnexion SpaceNavigator joystick, and a computer running the perception and control software.

Figure 3.1: An example of the experimental setup

3.1.1 Orthopus Explorer Robotic Arm

The Orthopus Explorer was selected because of its lightweight design, which makes it suitable for assistive robotics applications. Since the robot is intended to operate close to the user and potentially be mounted beside a wheelchair, using a lightweight manipulator is important to reduce the mechanical constraints on the platform and to improve safety during interaction.

The Orthopus Explorer can be controlled using different control modes, including position control, velocity control, and impedance control. In this work, the robot is operated in velocity control mode.

Although impedance control could be useful for improving physical interaction safety, it was not used in the implemented experiments. The experiments presented in this chapter are therefore based on velocity control, while position and impedance control remain available capabilities of the robotic platform.

Figure 3.2: Orthopus Explorer robotic manipulator used during the experiments

The perception system uses an Intel RealSense D435 RGB-D camera. The camera provides both RGB images and depth information. It is mounted close to the gripper, giving the robot an eye-in-hand configuration.

The RGB images are used for object and mouth detection, while the depth information is used to estimate the 3D position of the selected object.

Figure 3.3: Intel RealSense D435 RGB-D camera used for visual perception

In the assistive manipulation task, the joystick is located on either the right or left side of the user.

Figure 3.4: 3Dconnexion SpaceNavigator joystick used for user supervision

3.1.2 Joystick Interface

Figure 3.5: 3Dconnexion SpaceNavigator controller cap and buttons

The controller cap provides six degrees of input: three translational axes and three rotational axes. These axes are illustrated in Figure 3.6, where, from left to right, they correspond to `axes[1]`, `axes[2]`, `axes[0]`, `axes[4]`, `axes[5]`, and `axes[3]`, respectively (slightly different from the default configuration, where the axes are in sequential order from left to right). In ROS2, the joystick information is published as a message containing an array of axes and an array of buttons. The axes correspond to the continuous movements of the controller cap, while the buttons correspond to binary user inputs. It

should be noted that only `axes[0]`, `axes[2]`, and `axes[5]` are used alongside the right button for user validation.

Figure 3.6: Six-axis input generated by the SpaceNavigator controller cap

3.1.3 Object Selection Using the Joystick

Before the manipulation task starts, the user selects the target object from the visual interface. The perception system detects the objects present in the scene and displays them to the user. The joystick is then used to navigate through the available detected objects and validate the desired target.

Once the object is selected, its estimated three-dimensional pose is used as the target for the manipulation task. The selected object is then locked during the reaching phase in order to avoid unintentionally switching to another detected object.

This selection step ensures that the robot does not autonomously choose which object to manipulate. The task objective remains defined by the user.

3.1.4 Implementation of the Joystick Command Strategies

The joystick is used to provide several high-level commands to the system. In the implemented architecture, different joystick axes are associated with different roles: speed modulation, gripper opening and closing, and approach angle modification.

The signal associated with `axes[0]` is used to modulate the robot speed. This command allows the user to control the progression of the robot during the manipulation task.

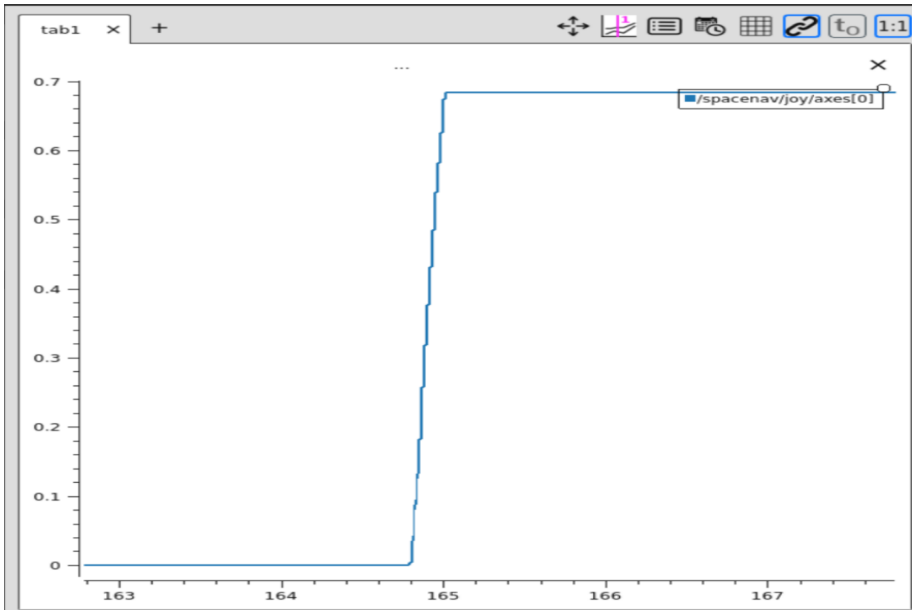


Figure 3.7: Example of a joystick signal corresponding to `axes[0]` visualized in PlotJuggler

The signal associated with `axes[2]` is used to open and close the gripper. Since the gripper command is a discrete action, the continuous joystick signal is transformed into a toggle command using an edge-triggered logic. This avoids repeatedly opening and closing the gripper while the user keeps the joystick axis activated.

Algorithm 1 Edge-triggered toggle command

Input: joystick signal s
Input: activation threshold τ
Input: toggle state b
Input: first-press flag f
pressed $\leftarrow (s < \tau)$
if pressed **and** f **then**
    $b \leftarrow \neg b$
    $f \leftarrow$ False
else if ¬pressed **then**
    $f \leftarrow$ True
end if

When the signal becomes lower than -0.6, it is interpreted as a grasping command. Each valid activation alternates between opening and closing the gripper.

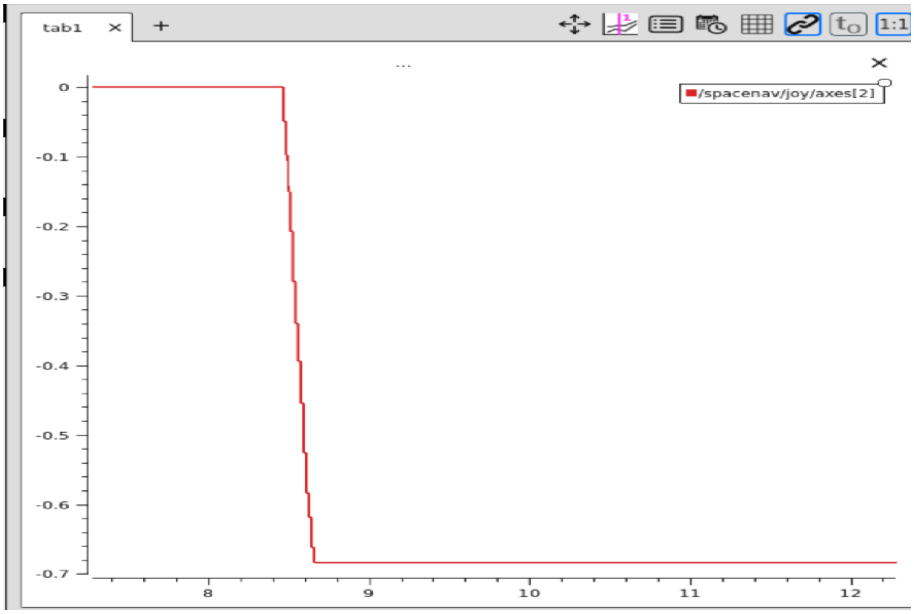


Figure 3.8: Joystick signal corresponding to `axes[2]` visualized in PlotJuggler

Finally, the rotational input around the vertical axis, corresponding to `axes[5]`, is used to modify the approach angle around the selected object. This allows the user to choose the direction from which the robot approaches the object during the orbiting phase.

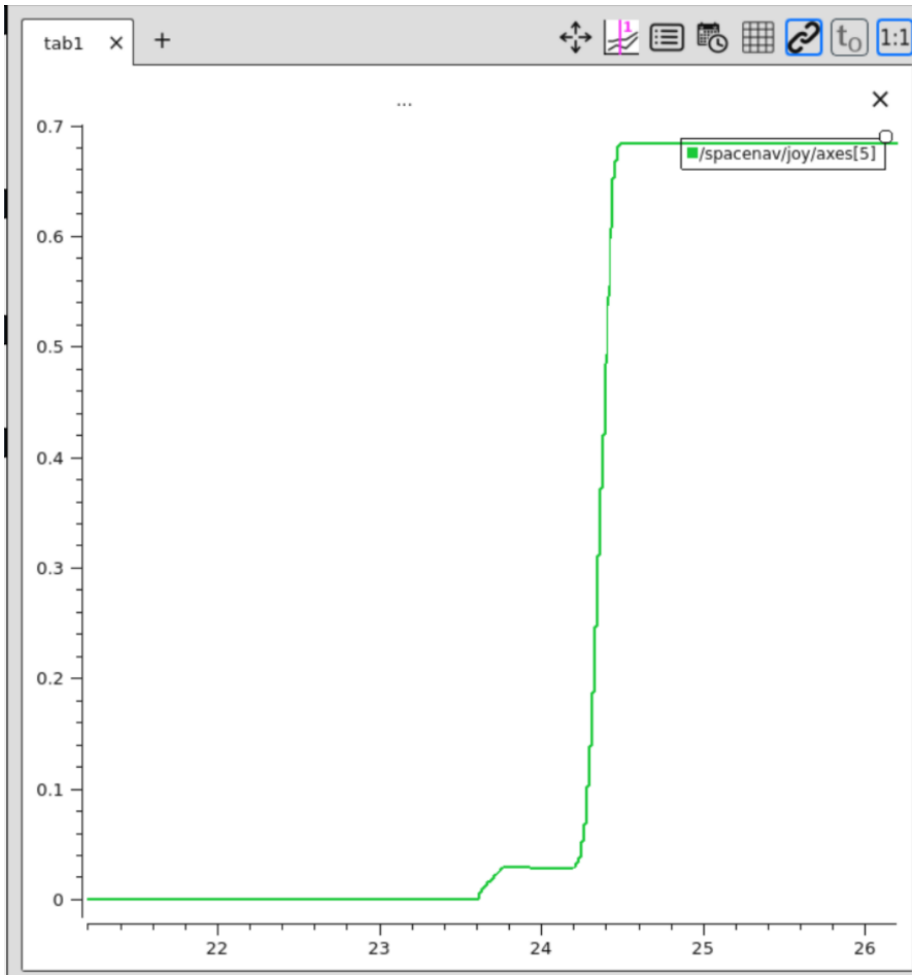


Figure 3.9: Joystick signal corresponding to `axes[5]` visualized in PlotJuggler

3.1.5 Input Smoothing, Normalization, and Saturation

Using the raw joystick signals directly is not optimal for robot control. The joystick inputs can change abruptly, and these sudden variations may generate discontinuous velocity commands. For example, if the user suddenly changes the speed input from a very small value to a much larger one, the resulting command can produce strong variations in the robot motion. This may lead to vibrations and can reduce the stability and smoothness of the manipulation task.

In the implemented system, two joystick axes are processed using a moving average filter: `axes[0]`, which is used for speed modulation and state-transition direction, and `axes[5]`, which is used to modify the approach angle around the selected object.

For a joystick signal s_k, the filtered signal $\bar{s}_k$ is computed over a window of N samples:

$$\bar{s}_k = \frac{1}{N} \sum_{i=0}^{N-1} s_{k-i}.$$

This filtering is applied to both `axes[0]` and `axes[5]`:

$$\bar{s}_k^{(0)} = \frac{1}{N} \sum_{i=0}^{N-1} s_{k-i}^{(0)},$$

$$\bar{s}_k^{(5)} = \frac{1}{N} \sum_{i=0}^{N-1} s_{k-i}^{(5)}.$$

The filtered signal $\bar{s}_k^{(5)}$ is used to modify the approach angle around the object. It allows the user to rotate the target point around the selected object during the orbiting phase.

For `axes[0]`, the absolute value of the filtered signal is used to compute the speed modulation coefficient. This coefficient must remain between 0 and 1:

$$\alpha_v = \min\left(1, \left|\bar{s}_k^{(0)}\right|\right).$$

The sign of $\bar{s}_k^{(0)}$ is kept separately:

$$\sigma_v = \operatorname{sign}\left(\bar{s}_k^{(0)}\right).$$

The coefficient α_v is used as the non-negative speed modulation command, corresponding to the output field `lin_x` inside the joystick node. The sign σ_v is used as an input variable for the state-transition logic. A positive sign corresponds to progressing forward in the task sequence, whereas a negative sign corresponds to a backward or rewind transition request.

With this processing, abrupt joystick variations are attenuated, the speed modulation coefficient remains bounded between 0 and 1, and the sign of the joystick command is preserved for the state-transition logic.

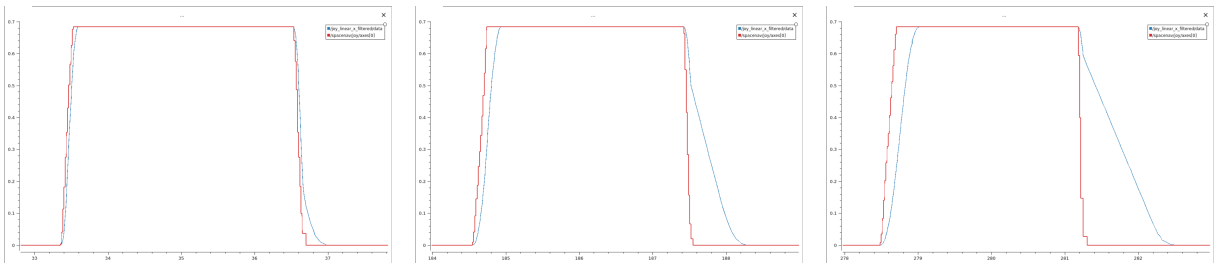
In order to choose an appropriate moving average window size, several experiments were carried out with values of N ranging from 10 to 40 samples. The objective was to find a compromise between filtering abrupt joystick variations and preserving sufficient responsiveness to the user's commands.

Table 3.1: Effect of the moving average window size on joystick input filtering

Window size N	**Observed behavior**
10	The signal remains responsive, but the filtering effect is limited. Sudden variations in the joystick input are still noticeable.
25	Good compromise between smoothing and responsiveness. Abrupt variations are reduced while the robot still reacts sufficiently quickly when the user changes or stops the input.
40	The signal becomes smoother, but the response delay starts to become noticeable. The robot motion becomes less reactive to rapid user input changes.

Based on these observations, a window size of $N = 25$ samples was selected as a good compromise. Lower values did not filter the joystick signal sufficiently, while higher values introduced too much delay in the robot response.

Figure 3.10 illustrates the effect of different moving average window sizes on the joystick signal. The blue signal corresponds to the filtered signal, while the red signal corresponds to the raw input.



(a) $N = 10$ (b) $N = 25$ (c) $N = 40$

Figure 3.10: Comparison of joystick input smoothing using different moving average window sizes

3.1.6 Software Framework and Libraries

The implementation relies on several software tools and libraries. ROS2 is used as the communication framework between the different modules of the system. Pinocchio is used for robot kinematics, CasADi is used for symbolic modeling of the optimal control problem, and acados is used to solve the NMPC problem in real time [6, 7, 8].

The perception pipeline uses YOLO11-seg for object detection and segmentation, and MediaPipe for mouth detection.

3.1.7 ROS2 Framework

The complete system is integrated into ROS2. The ROS2 framework allows the perception module, joystick interface, state machine, controller, and robot interface to exchange information through topics.

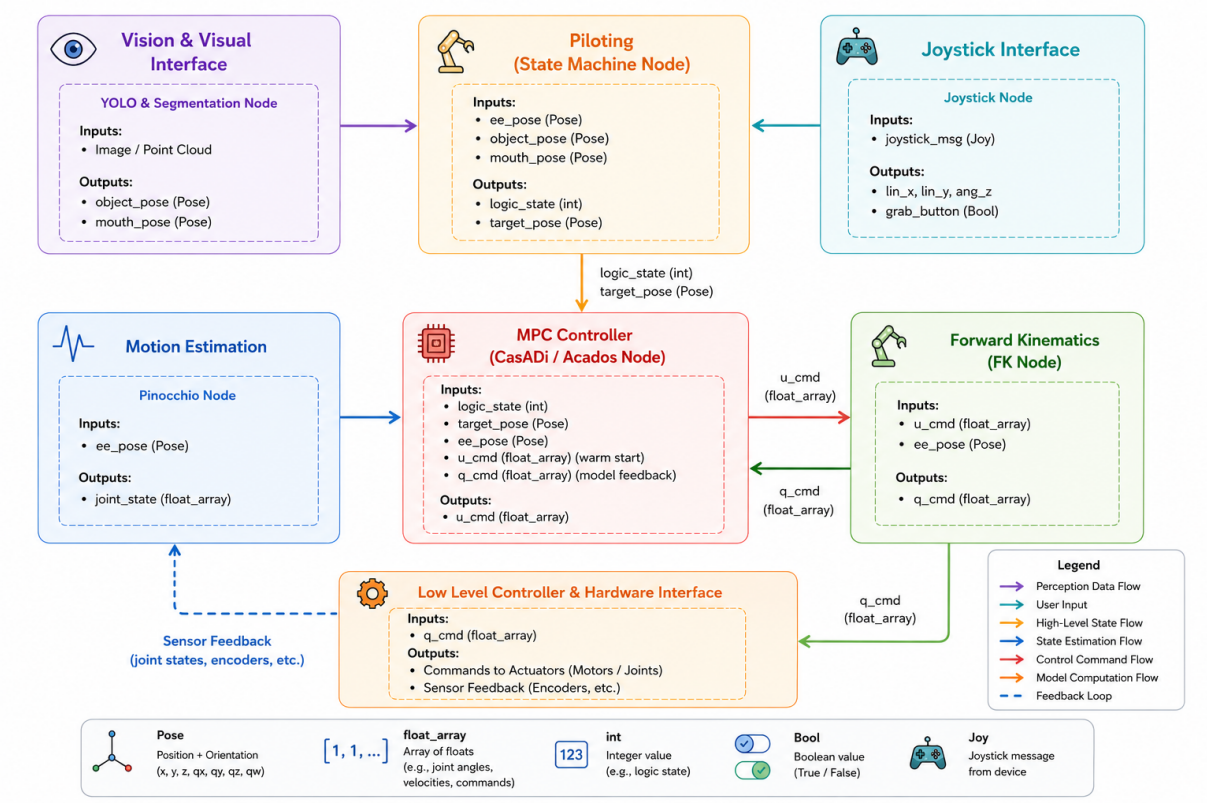


Figure 3.11: ROS2 framework used to integrate perception, user interaction, and control

3.2 MPC Parameter Tuning

The behavior of the NMPC controller depends strongly on the values selected for the cost weights, activation coefficients, arbitration coefficients, and internal weighting matrices. These parameters were tuned experimentally in order to obtain a compromise between convergence speed, smoothness, stability, and safe robot postures.

This section presents the final parameter values used during the experimental validation.

The main NMPC cost weights are:

$$w_{\text{pose}} = 0.02, \qquad w_{\text{vel}} = 3.5, \qquad w_{\text{align}} = 4.0, \qquad w_{\text{posture}} = 2.5.$$

The activation coefficients α_i are used to activate or deactivate specific cost terms according to the current logical state s_k of the state machine. The values used in the experiments are shown in Table 3.2.

Table 3.2: Activation coefficients used in the NMPC formulation

State s_k	$\boldsymbol{\alpha_1}$	$\boldsymbol{\alpha_2}$	$\boldsymbol{\alpha_3}$	$\boldsymbol{\alpha_4}$	$\boldsymbol{\alpha_5}$	$\boldsymbol{\alpha_6}$	$\boldsymbol{\alpha_7}$
$\{0, 1\}$	0	1	0	0	0	0	0
$\{2, 3\}$	1	1	0	0	0	0	0
$\{4, 5, 10\}$	1	1	1	0	0	0	0
6	1	1	1	1	0	0	0
7	0	1	1	0	1	0	0
8	0	1	1	0	0	1	0
$\{9, 11\}$	1	1	1	0	0	0	1
12	0	1	1	0	0	0	0

The coefficient α_1 activates the position tracking part of the pose cost, α_2 activates the velocity regularization term, and α_3 activates the vertical alignment objective. The coefficients α_4, α_5, α_6, and α_7 activate the different posture objectives associated with the nominal, mouth, pouring, and go-back postures.

3.2.1 Arbitration Coefficients

The arbitration coefficients β_1 and β_2 modulate the influence of some cost terms. Unlike the activation coefficients, they can introduce a progressive transition between objectives. The values used in the experiments are presented in Table 3.3.

Table 3.3: Arbitration coefficients used in the NMPC formulation

State s_k	$\boldsymbol{\beta_1(s_k)}$	$\boldsymbol{\beta_2(s_k)}$
$\{0, 1, 2, 3, 4, 5, 10, 12\}$	1	1
$\{6, 7, 9, 11\}$	0	1
8	$\exp(-0.05 d_{\text{mouth}})$	$1 - \exp(-0.05 d_{\text{mouth}})$

The coefficient β_1 modulates the influence of the pose tracking objective, while β_2 modulates the influence of the go-back posture objective. In state $s_k = 8$, both coefficients depend on the distance to the mouth d_{mouth}. This creates a smooth transition between pose tracking and posture regulation during the preparation for pouring.

3.2.2 Internal Weighting Matrices

The internal weighting matrices distribute the importance of each error inside a given cost term. They do not define the global priority of the objective, but they specify how

this objective is weighted between Cartesian directions or robot joints.

Position Weighting Matrix

The position weighting matrix is:

$$W_p = \begin{bmatrix} w & 0 & 0 \\ 0 & w & 0 \\ 0 & 0 & w \end{bmatrix},$$

with:

$$w = \begin{cases} 65.0, & \text{if } s_k = 3, \\ 20.0, & \text{otherwise.} \end{cases}$$

The value of w is increased during the orbiting phase $s_k = 3$. This reinforces the position tracking around the selected object while the user adjusts the approach direction.

Orientation Weighting Coefficient

The orientation error is weighted using:

$$W_R(s_k) = \begin{cases} 20.0, & \text{if } s_k = 2, \\ 40.0, & \text{if } s_k = 3, \\ 0.0, & \text{otherwise.} \end{cases}$$

This means that orientation tracking is mainly active during the motion toward the orbit and during the orbiting phase. During the other phases, this specific orientation term is deactivated, while the vertical alignment cost maintains the gripper orientation when required.

Joint Posture Weighting Matrix

The posture weighting matrix is fixed and defined as:

$$W_x = \operatorname{diag}(20.0,\ 5.0,\ 6.0,\ 7.0,\ 3.0,\ 3.0).$$

3.2.3 State and Input Bounds

The NMPC also uses bounds on the joint configuration and joint velocity commands.

The lower joint limits are:

$$x_{\min} = \begin{bmatrix} -2.97 & -2.14 & -2.97 & -2.97 & -1.75 & -2.97 \end{bmatrix}^T.$$

The upper joint limits are defined symmetrically:

$$x_{\max} = -x_{\min}.$$

The upper velocity command limits are:

$$u_{\max} = \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix}^T.$$

The lower velocity command limits are:

$$u_{\min} = -u_{\max}.$$

These bounds prevent the optimizer from generating joint configurations or velocity commands outside the admissible range. They also contribute to safer and smoother robot behavior during the experiments.

3.3 Experimental Results

The experiments were conducted with an MPC frequency of 20 Hz and an internal controller frequency of 250 Hz.

3.3.1 Experiment 1: Influence of the Vertical Alignment Cost

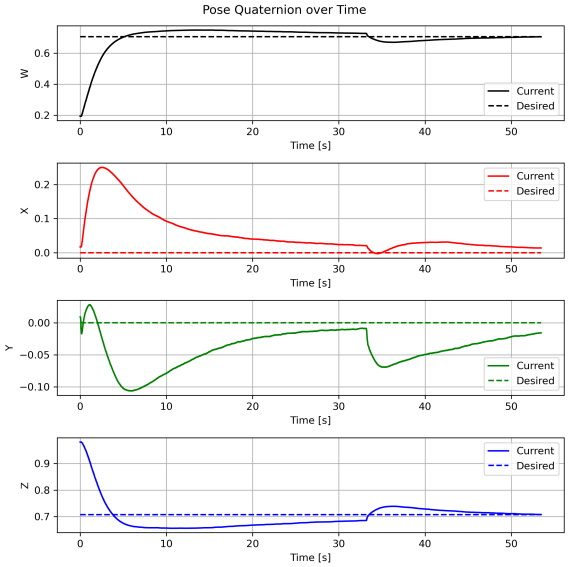
The first experiment evaluates the importance of the vertical alignment cost in the NMPC formulation. This experiment was performed in an autonomous configuration, without user interaction through the joystick. The objective was not to reproduce the complete semi-autonomous assistive task, but rather to tune the NMPC weights and to visualize the influence of the vertical alignment term on the robot behavior.

The experiment consists of moving the robot between two predefined desired poses. These two setpoints define the Cartesian references that the end-effector must reach successively. Since the task is autonomous, the reference trajectory is not affected by the user's joystick command. This makes it possible to compare the behavior of the controller with and without the vertical alignment cost under the same conditions.

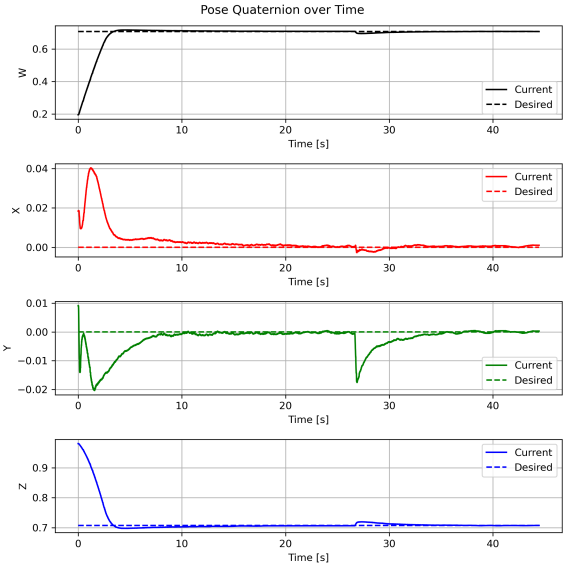
Both experiments were performed from the same initial configuration. At the beginning of the motion, the end-effector was already approximately horizontal. This represents a favorable initial condition, because the controller starts from a configuration where the gripper orientation is already suitable for manipulation. Therefore, any deviation from the horizontal orientation during the trajectory is mainly caused by the absence or presence of the vertical alignment cost.

Two configurations were compared: one with $w_{\text{align}} = 0.0$ and the other with $w_{\text{align}} = 4.0$.

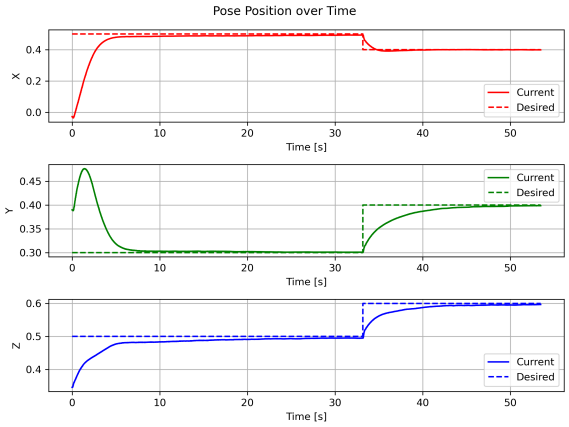
Figure 3.12 shows the comparison between the two configurations. The quaternion plots show the evolution of the current and desired end-effector orientation, while the position plots show the corresponding Cartesian tracking behavior.



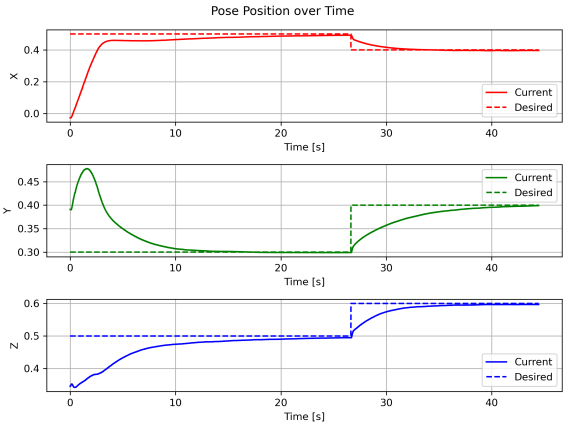
(a) Quaternion tracking without vertical alignment



(b) Quaternion tracking with vertical alignment



(c) Position tracking without vertical alignment



(d) Position tracking with vertical alignment

Figure 3.12: Comparison of the robot behavior with and without the vertical alignment cost

Without the vertical alignment cost, the position tracking remains acceptable, as shown in Figure 3.12c. The end-effector reaches the desired Cartesian references along the x, y, and z directions. However, the orientation tracking in Figure 3.12a shows that the end-effector orientation is not well preserved during the motion.

In the experiment without vertical alignment, the roll-related deviation reaches approximately 0.24 rad around $t = 2.5$ s. It then decreases slowly and remains higher than approximately 0.05 rad until about $t = 15$ s. This means that the end-effector remains

significantly tilted for more than ten seconds.

The pitch-related deviation also becomes significant. It reaches approximately 0.1 rad around $t = 5.5\,\text{s}$, and remains higher than approximately 0.05 rad between $t = 5\,\text{s}$ and $t = 15\,\text{s}$. A second deviation appears after the change of reference around $t = 33\,\text{s}$, where the pitch-related error reaches approximately 0.07 rad around $t = 35\,\text{s}$. This shows that the end-effector orientation is not correctly regulated when the vertical alignment cost is removed.

This behavior is problematic for assistive manipulation. During object transport, and especially during a drinking assistance task, the gripper must remain approximately horizontal in order to keep the object stable. Specifically, in this situation, where the initial orientation is almost perfectly horizontal, a roll or pitch deviation of more than 0.087 rad is unreliable, since the trajectory is small and the deviation from the horizontal orientation is noticeable.

With the vertical alignment cost, the behavior is significantly improved. As shown in Figure 3.12b, the orientation deviation remains much smaller. The roll-related deviation reaches only approximately 0.04 rad during the initial transient, around $t = 1.5\,\text{s}$, and then quickly converges close to zero. After approximately $t = 5\,\text{s}$, it remains almost negligible.

The pitch-related deviation is also strongly reduced. Its maximum value is approximately 0.02 rad around $t = 1.5\,\text{s}$. After this short transient, it converges close to zero and remains stable for most of the trajectory. A second short deviation appears around $t = 27\,\text{s}$, when the reference changes, but the error remains small and quickly decreases again.

The position tracking with vertical alignment, shown in Figure 3.12d, remains satisfactory. The addition of the vertical alignment cost does not prevent the robot from reaching the desired Cartesian position. Instead, it improves the orientation behavior while preserving the ability of the controller to track the position reference.

This experiment shows that the vertical alignment cost is necessary for the considered task. Even when the robot starts from a favorable horizontal end-effector orientation, removing this cost leads to significant orientation deviations during the motion. Without vertical alignment, the roll and pitch deviations can reach approximately 0.24 rad and 0.1 rad, respectively. With $w_{\text{align}} = 4.0$, these deviations are strongly reduced and limited to short transient phases. Therefore, the vertical alignment term is essential to maintain a stable gripper orientation during object transport and pouring.

3.3.2 Experiment 2: Complete Semi-Autonomous Manipulation Task

The second experiment illustrates an example of the complete semi-autonomous manipulation task. Unlike the first experiment, which was performed autonomously to tune and visualize the influence of a specific NMPC weight, this experiment involves the complete

interaction loop between the user, the joystick, the state machine, and the NMPC controller. It should be noted that, in this experiment, the state referred to as the pouring state does not correspond to a real pouring motion. Instead, it was temporarily replaced by a drinking assistance state. In this state, the robot brings the object close to the user's mouth and allows the user to drink using a straw. This choice was made because implementing a complete pouring motion is more complex and requires additional constraints on the cup orientation, liquid flow, mouth position, and safety during human interaction.

The objective was to evaluate the behavior of the system during a representative manipulation sequence including object approach, orbiting, reaching, grasping, lifting, intermediate posture regulation, drinking assistance using a straw, and object return.

Figure 3.13 shows the evolution of the end-effector orientation during the complete task. The orientation is represented using roll, pitch, and yaw angles in degrees. The vertical black lines indicate the transitions between the different states of the state machine.

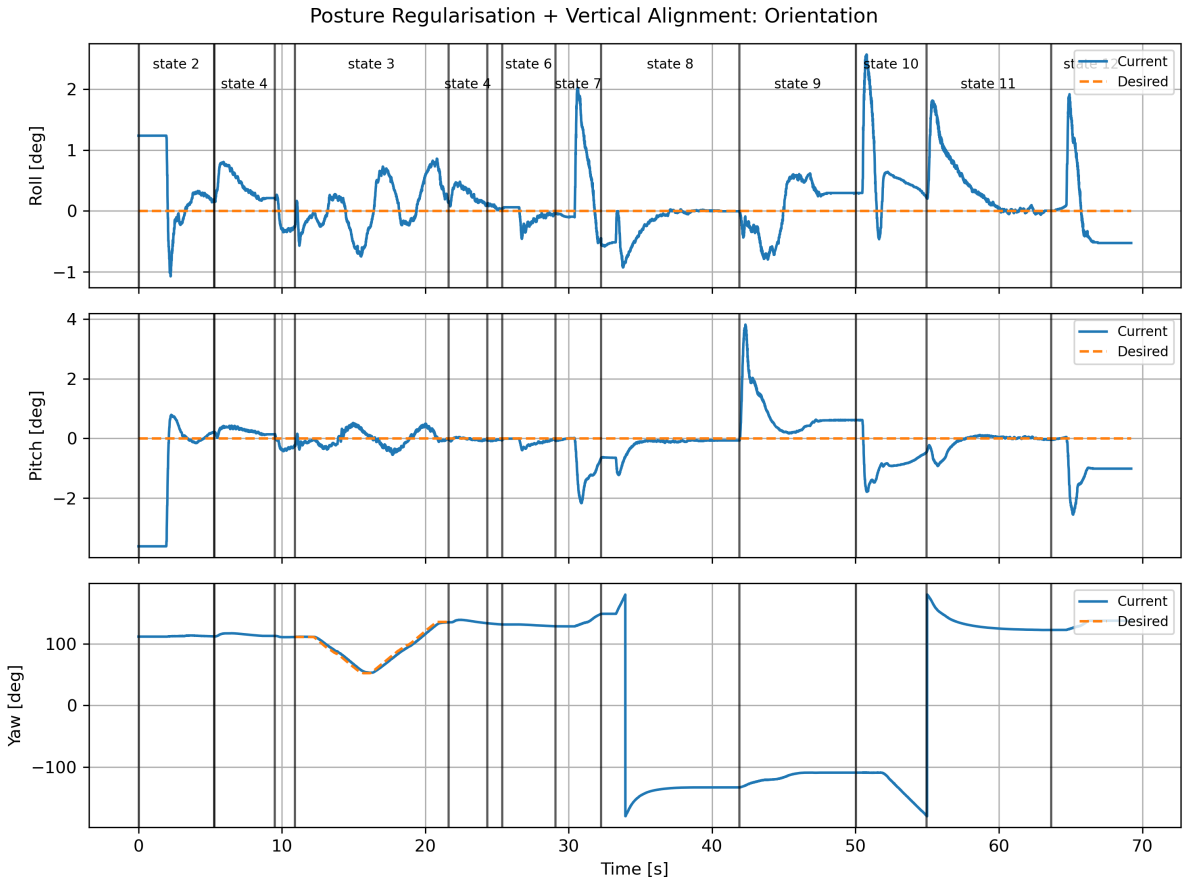


Figure 3.13: Orientation evolution during the complete semi-autonomous manipulation task

The desired roll and pitch are kept equal to 0° during the complete manipulation task. This means that the controller tries to maintain the end-effector approximately horizontal while the object is manipulated. The results show that the current roll and pitch remain close to zero during most of the experiment. The maximum roll error is approximately

2.5°, while the maximum pitch error is approximately 3.9°. These maximum deviations only appear during short transient phases, mainly around state transitions.

This behavior confirms the effect of the vertical alignment and posture regularization costs. The end-effector orientation remains stable during the manipulation task, which is important for object transport and for bringing the object close to the user's mouth in a safe and controlled way. Compared with the experiment without vertical alignment, where the roll and pitch deviations were much larger and lasted for a long time, the complete task shows that the gripper orientation is maintained around the desired horizontal configuration.

For the yaw angle, the desired trajectory is only important during state 3, which corresponds to the orbiting phase. In this state, the user modifies the approach direction around the object using the joystick, and the yaw reference changes accordingly. The plot shows that the current yaw follows the desired yaw accurately during this phase. In the other states, yaw tracking is not a priority, because the main objective is to maintain the gripper horizontal and execute the manipulation task safely.

Figure 3.14 shows the evolution of the end-effector Cartesian position during the same experiment. The plot compares the desired and current positions along the x, y, and z axes.

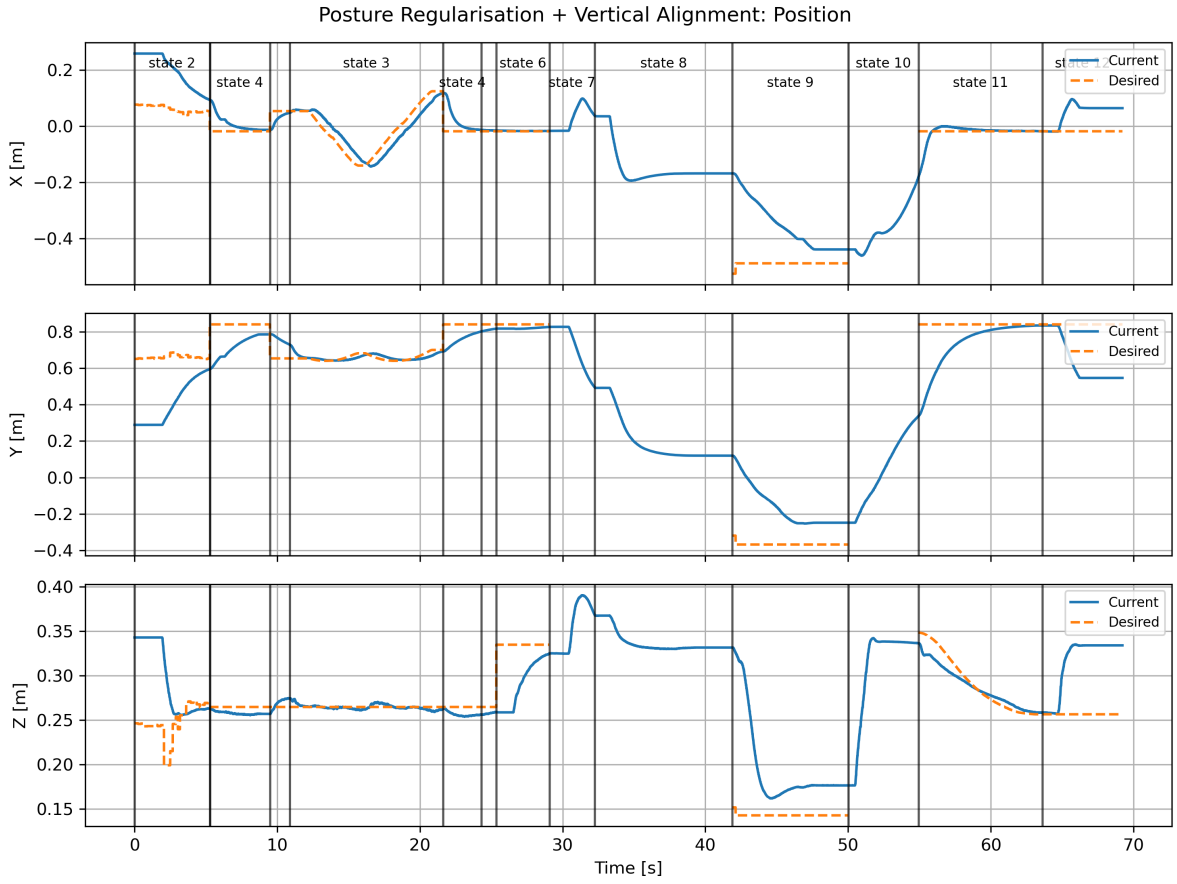


Figure 3.14: Cartesian position evolution during the complete semi-autonomous manipulation task

The position plot shows that the end-effector follows the desired Cartesian references during the states where a target position is defined. As mentioned previously, the desired position is not plotted in states 7, 8, and 10, because these states correspond to intermediate posture configurations. In these phases, the objective is not to track a Cartesian position target, but rather to guide the robot toward a suitable joint posture.

In state 11, the tracking of the z-coordinate is particularly important because this state corresponds to putting the object back. The plot shows that the current z-position follows the desired evolution well. This motion emulates the gesture of returning the object to its original place. In the implementation, this behavior is generated using a vertical offset that decreases exponentially as the end-effector approaches the initial object position. Therefore, the robot first keeps the object slightly above the placement area and then progressively lowers it when it gets closer to the target.

Some slight jaggedness can be observed in the Cartesian trajectories. This is mainly due to imperfections in the user's joystick command during the semi-autonomous experiment. Since the user controls the progression of the task, small variations in the joystick input can affect the robot motion. This effect was not observed in the same way during the autonomous experiment of Section 3.3.1, where the motion was generated without user input and the trajectories were smoother.

Another limitation observed during the experiments concerns the object position estimation. In some cases, the detected object position was not perfectly accurate. This produced a slight mismatch between the estimated object center and the gripper center frame when reaching the object. As a result, the gripper could arrive with a small alignment error with respect to the real object position.

A second limitation was observed during the return to the finalise state. After putting the object back, the robot moves back toward its initial configuration. In some cases, the gripper fingers could pass too close to the object and potentially hit it during this return motion. This shows that the transition after the put-back phase should include an additional safety movement before returning to the initial posture.

Overall, the performance of the system is satisfactory. The robot is able to execute the complete manipulation sequence while maintaining a stable end-effector orientation and following the desired Cartesian references. The experiment was repeated several times using a cup and different object positions, and similar results were obtained. This confirms the consistency of the proposed semi-autonomous manipulation architecture, while also highlighting some points that should be improved in future work.

3.3.3 Demonstration

A demonstration of the system was carried out at ISIR, Sorbonne Université, in Paris. The demonstration showed that the proposed architecture can be used in a real experimental context and that the user can supervise the robot through the joystick while the controller generates the motion automatically. In this demonstration, the robot was attached to a

moving cart. Some changes were made to the desired postures because the robot was positioned on a different side than in the LAAS-CNRS laboratory. Despite these changes, the pipeline remained robust, and the manipulation process was successful.

3.4 Discussion

The experimental results show that the proposed approach is functional for the considered assistive manipulation scenario. The perception system provides the required object and mouth positions, the state machine correctly organizes the task, and the NMPC controller generates smooth and predictable robot motions.

The joystick interface allows the user to supervise the robot without directly controlling all joint motions. This reduces the control effort required from the user while preserving user involvement in the task. The goal is also to reduce the user's mental load: instead of manually controlling each joint or each Cartesian motion, the user only provides high-level commands such as speed modulation, approach direction, and action validation. This semi-autonomous strategy aims to complete the manipulation task in the shortest possible time while keeping the motion safe, smooth, and easy for the user to supervise.

The results also show the importance of filtering the joystick input. Without filtering, abrupt variations in the joystick signal may lead to discontinuous velocity commands. The moving average filter reduces this effect and improves the smoothness of the robot motion.

Overall, the experimental validation confirms that the combination of visual perception, joystick-based supervision, state-machine coordination, and NMPC control is suitable for semi-autonomous assistive manipulation.

Chapter 4

Conclusion and Perspectives

4.1 Conclusion

This internship focused on the development of a semi-autonomous control architecture for an assistive robotic manipulator. The objective was to allow a user to supervise a robotic arm during manipulation tasks, while the robot automatically generated smooth and constrained motions.

The developed system combines an eye-in-hand RGB-D perception system, object detection and segmentation, mouth detection, joystick-based user supervision, a state machine, and NMPC control. The user can select the target object, adjust the approach direction, control the progression speed, and validate the grasping action.

Experimentally, the proposed approach proved to be functional and effective for the considered assistive manipulation scenario. Despite the presence of some jaggedness, the generated robot motions can be considered visually smooth, and no sudden movements were observed during the tests. A demonstration was also carried out at ISIR, Sorbonne Université, confirming the relevance of the developed system in a real experimental context.

4.2 Perspectives

Several improvements could be considered in future work.

First, the current system estimates the position of the object, but not its full pose. However, for the manipulation task, the most important objective is not only to know the object pose itself, but to determine a suitable desired grasping pose for the gripper. Estimating the full object pose would provide more information about the object orientation in space, which could then be used to compute a better grasping pose.

This would reduce the effort required from the user, since the robot could automatically infer a more appropriate approach direction and gripper orientation. It would also improve the generalization ability of the system to different objects and object configurations.

Approaches for 6D object pose estimation, such as MegaPose [12], could be considered for this purpose. Other grasping libraries or learning-based grasp detection methods could also be investigated, especially methods that directly estimate desired grasping poses instead of only estimating the object pose.

Second, impedance control could be introduced to improve safety during physical interaction. Compared with pure joint position control, impedance control adds mechanical compliance to the robot behavior. This could reduce the risk of injury to the user or damage to the robot in case of unexpected contact [13].

Third, due to the limited duration of the internship, the manipulation task was not experimentally tested in a scene containing several objects at the same time. However, this case was already taken into account in the software architecture. The perception module can detect multiple objects, and the user interface is designed to allow the user to select the desired object before the manipulation task starts. This functionality should therefore be tested experimentally in future work in order to validate the complete multi-object selection and manipulation pipeline.

Fourth, more time could be dedicated to parameter tuning. The NMPC behavior depends strongly on the selected weights, activation coefficients, and posture references. A more systematic tuning procedure would improve the compromise between convergence speed, smoothness, stability, and safety. Another improvement concerns the accuracy of the object position estimation. During the experiments, slight mismatches were sometimes observed between the detected object position and the real object position. This can lead to a small alignment error between the gripper center frame and the object during the reaching phase. A better calibration of the camera with respect to the end-effector could reduce this error. More specifically, a more accurate estimation of the camera-to-end-effector transformation would improve the projection of the detected object position into the robot reference frame, and therefore increase the accuracy of the reaching motion.

The return motion after the put-back phase could also be improved. In the current implementation, when the robot goes back to the finalise state, the gripper fingers may pass close to the object and could hit it after the object has been released. A possible solution would be to add an intermediate retreat movement before returning to the initial configuration. For example, after putting the object back, the robot could first move away from the object along the approach direction, or return backward from the put-back angle, before moving toward the finalise state. This would increase the safety of the return trajectory and reduce the risk of contact with the object.

Another important perspective is the implementation of a real pouring state. In the current experiments, the pouring state was temporarily replaced by a drinking assistance state, where the robot brings the object close to the user's mouth and allows the user to drink using a straw. A complete pouring motion was not implemented due to its higher complexity. Future work could focus on designing a dedicated pouring strategy, including cup tilting, control of the pouring angle, estimation of the mouth position, user

safety constraints, and validation of the liquid flow during the task. This would make the drinking assistance scenario more complete and closer to a fully functional assistive manipulation task.

Finally, collision constraints could be directly included in the optimization problem. One possible direction would be to use conformal geometric algebra, which provides a unified framework to represent geometric primitives such as points, lines, planes, spheres, and other shapes. This could help formulate collision constraints in a more general way for complex robot and environment geometries [14].

Overall, the developed system represents a first functional step toward semi-autonomous assistive manipulation. The results show that combining user supervision, visual perception, and NMPC control is a promising approach for helping users perform manipulation tasks while preserving safety and predictability.

Bibliography

[1] J. Luo, L. Zhu, L. Li, and P. Hong, "Robot Visual Servoing Grasping Based on Top-Down Keypoint Detection Network," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, 2024.

[2] K. Muelling, A. Venkatraman, J.-S. Valois, et al., "Autonomy Infused Teleoperation with Application to Brain Computer Interface Controlled Manipulation," *Autonomous Robots*, vol. 41, no. 6, pp. 1401–1422, 2017.

[3] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. Sucan, "Towards Reliable Grasping and Manipulation in Household Environments," *Experimental Robotics*, 2011.

[4] F. Gebrayel, M. Mujica, and P. Danes, "Visual Servoing for Vine Pruning based on Point Cloud Alignment," *ICINCO*, 2024.

[5] F. D. P. da Luz, "Model Predictive Control for Assistive Robotic Manipulation," Master's Thesis, University of Lisbon, 2021.

[6] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ Library: A Fast and Flexible Implementation of Rigid Body Dynamics Algorithms and their Analytical Derivatives," in *IEEE International Symposium on System Integration (SII)*, 2019.

[7] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A Software Framework for Nonlinear Optimization and Optimal Control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019. doi: `https://doi.org/10.1007/s12532-018-0139-4`.

[8] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados: A Modular Open-Source Framework for Fast Embedded Optimal Control," *Mathematical Programming Computation*, 2021. doi: `https://doi.org/10.1007/s12532-021-00208-8`.

[9] M. Behrendt, "MPC scheme basic.svg," Wikimedia Commons, 2009. Available: `https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg`. Accessed: July 22, 2026.

[10] G. Jocher and J. Qiu, "Ultralytics YOLO11," Ultralytics, 2024. Available: https://github.com/ultralytics/ultralytics. Accessed: August 3, 2026.

[11] Google AI Edge, "MediaPipe Face Landmarker," Google for Developers. Available: https://developers.google.com/edge/mediapipe/solutions/vision/face_landmarker. Accessed: August 3, 2026.

[12] Y. Labbé, L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpentier, M. Aubry, D. Fox, and J. Sivic, "MegaPose: 6D Pose Estimation of Novel Objects via Render & Compare," in *Proceedings of the 6th Conference on Robot Learning*, PMLR, vol. 205, pp. 715–725, 2023. Available: https://proceedings.mlr.press/v205/labbe23a.html.

[13] N. Hogan, "Impedance Control: An Approach to Manipulation: Part I–Theory," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–7, 1985.

[14] Y. Wu, G. Wang, S. Chen, Z. Shi, Y. Guan, and X. Li, "Formalization of Robot Collision Detection Method based on Conformal Geometric Algebra," arXiv preprint arXiv:2312.04598, 2023. Available: https://arxiv.org/abs/2312.04598.