POLITÉCNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES
UNIVERSIDAD POLITÉCNICA DE MADRID

José Gutiérrez Abascal, 2. 28006 Madrid
Tel.: 91 336 3060
info.industriales@upm.es

www.industriales.upm.es

Daniel Sotelo Aguirre

05 TRABAJO FIN DE MASTER

INDUSTRIALES

TRABAJO FIN DE MASTER

DEVELOPMENT AND INTEGRATION OF A NMPC-CONTROLLED LEGGED-MANIPULATOR PLATFORM FOR SEARCH AND RESCUE OPERATIONS

TRABAJO FIN DE MASTER
PARA LA OBTENCIÓN DEL
TÍTULO DE **MASTER EN AUTOMÁTICA Y ROBÓTICA**

SEPTIEMBRE **2024**

Daniel Sotelo Aguirre

DIRECTORES DEL TRABAJO FIN DE MASTER:
Antonio Barrientos Cruz
Christyan Cruz Ulloa

INDUSTRIALES
ETSII | UPM

POLITÉCNICA

UNIVERSIDAD
POLITÉCNICA
DE MADRID

Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Industriales

Máster Universitario en Automática y Robótica

Trabajo Fin de Máster

Development and Integration of a NMPC-Controlled Legged-Manipulator Platform for Search and Rescue Operations

Author: **Daniel Sotelo Aguirre**

Director: **Antonio Barrientos Cruz**
Full Professor

External Director: **Christyan Cruz Ulloa**
Postdoctoral Researcher

Madrid. September, 2024

"It's supposed to be automatic,
but actually you have to push this button."

John Brunner

Acknowledgements

This thesis marks the culmination of an incredible journey, one that I could not have completed without the unwavering support of many individuals. I am deeply grateful for the guidance and encouragement I have received throughout this process. First and foremost, I would like to express my heartfelt gratitude to Antonio Barrientos for giving me the opportunity to work on such an exciting project. Your efforts in finding a topic that aligned with my interests, along with your insightful advice and support throughout the project, have been invaluable.

I would also like to extend my heartfelt thanks to my colleagues and peers at CAR, which has been like a second home to me throughout this year. A special recognition goes to Christyan for your invaluable assistance in the lab and for always being available to help. I would also like to thank David for his support in overcoming the challenges I faced as a novice to Linux and ROS. Your help made a significant difference in my learning process.

My heartfelt thanks also go to my family, especially my parents, for their unconditional love and support. Thank you for always being there to listen, encourage, and keep me grounded. To my brother, thank you for the demo video editing and all the laughs. Your sense of humor and ability to bring joy to every situation have been a much-needed source of relief during this adventure.

I am also deeply grateful to my friends in Madrid over the past two years, who have made me feel at home. Your friendship and support have meant the world to me. I would also like to thank my friends from Pamplona, who provided me with a much-needed escape and sense of disconnection when I needed it most.

A sincere thanks to the Wheely Wonkas robotics team for your understanding and support throughout this project. Thank you for giving me the space to focus on this thesis when I needed it most and for your comprehension during the busy times.

I also want to extend my gratitude to the incredible people I met during the double master's program. Thank you for all the funny moments, for introducing me to great and interesting people, and for making this experience not only intellectually enriching but also filled with memorable experiences.

Lastly, I want to express my gratitude to everyone who, in one way or another, contributed to this thesis. Whether through a kind word, a shared laugh, or a small act of kindness, each gesture has made a difference. Thank you all for being part of this journey with me.

Executive Summary

The rapid advancement of **robotics** has revolutionized various industries, particularly in the realm of **Search and Rescue (SAR)**, where robots are becoming a major focus of research for deployment in hazardous and unpredictable environments. Among these, **legged-manipulator** platforms are attracting significant research interest due to their potential for performing tasks that require both mobility and manipulation in complex, unstructured terrains. These platforms combine the agility and stability of legged robots with the dexterity of manipulators, making them uniquely suited to SAR operations where the ability to traverse rough terrain and interact with objects is essential.

At the Technical University of Madrid, the RobCib research group, part of the Center for Automation and Robotics (CAR), has been at the forefront of developing advanced robotic systems for SAR operations. Over the years, the group has made significant strides in this area, particularly with the development of ARTU-R, a quadruped robot designed for victim detection in SAR scenarios. ARTU-R was a pioneering project that demonstrated the potential of legged robots in SAR tasks, particularly in environments where wheeled or tracked robots would struggle. However, despite its successes, the project also highlighted several limitations, particularly in the robot's control system, which relied on **predefined gait patterns** that were **not well-suited** to the dynamic and **unpredictable conditions** typically encountered in SAR missions. Additionally, it lacked the capability to **manipulate objects** with a payload higher than 250 g, a critical function for tasks such as opening doors or clearing debris.

Motivated by these challenges, the next phase of research within the RobCib group focused on developing a more sophisticated and versatile robotic platform. This led to the integration of a more capable manipulator with a robust quadruped platform. This Master's Thesis centers on the **integration and control** of this legged-manipulator platform, specifically designed to enhance the operational capabilities of SAR robots. The project involved the adaptation, implementation, and testing of a **Nonlinear Model Predictive**

Control (NMPC) system for the Unitree AlienGo quadruped robot, equipped with a Z1 robotic arm as shown in Figure R1. The primary objective was to create a robust control system that could significantly improve the robot's locomotion and manipulation capabilities in challenging SAR environments, such as traversing unstable terrains and performing tasks like door opening.

Figure R1. Unitree AlienGo quadruped equipped with Z1 arm [46][86].

NMPC is an advanced control strategy used to handle complex, dynamic systems with nonlinear behavior, like the legged-manipulator platform described in this thesis. NMPC works by **predicting the future behavior** of the system over a finite time horizon and optimizing control actions based on this prediction. By continuously solving an **optimization proble**m at each time step, NMPC allows for real-time adjustments to the robot's movements and interactions, ensuring smooth and precise control in response to dynamic environments.

To achieve these objectives, the research undertook a comprehensive review of existing control strategies used in legged robots. It was clear that traditional control systems, which were often designed for structured environments, were insufficient for the unpredictable and dynamic conditions of SAR operations. The NMPC system used in this thesis was tailored to overcome these challenges, providing a more flexible and adaptive control strategy that could dynamically adjust the robot's behavior in response to the changing conditions of its environment.

To develop the NMPC system, an external code repository based on **ROS Noetic** was used as a foundation. However, this repository had several limitations that needed to be addressed to adapt it to the specific requirements of the Unitree Aliengo quadruped and the Z1 robotic arm. The enhancements enabled the controller to be tailored to the specific kinematics and dynamics of the system, allowing it to be implemented in a more advanced system based on ROS nodes that utilizes computer vision and planning modules, eliminating the need for direct user input.

In this Master's Thesis, significant effort was dedicated to developing an **advanced computer vision system** to enhance the robot's ability to interact autonomously with its environment, a critical requirement for SAR operations where constant user control may not be available. The vision system was specifically designed to **detect and manipulate door handles**, enabling the robot to perform complex tasks such as door opening without human intervention. Utilizing information retrieved from two depth cameras, this system allows the robot to accurately position itself, detect and manipulate door handles autonomously in controlled simulation environments.

To validate the NMPC system and the integration of the Z1 manipulator, extensive **simulations** were conducted using the **Gazebo** platform. These simulations assessed the robot's performance in various SAR-relevant scenarios, such as navigating uneven terrains, climbing stairs, and opening doors. The results demonstrated significant improvements over previous control strategies, showcasing the enhanced capabilities of the system.

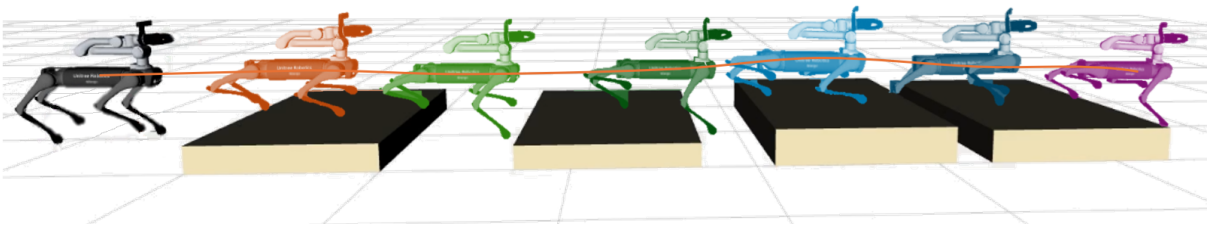


Figure R2. Pallet path simulation.

In addition to simulations, **real-world tests** were conducted in controlled environments to further evaluate the system's performance. Although limited by time constraints and resource sharing, these tests provided valuable insights into the robot's practical capabilities and highlighted areas for future improvement.

Overall, this thesis represents a substantial improvement over the control methods previously employed by the research group for their legged robotic platforms. The successful implementation of the NMPC system, combined with the vision-integrated Z1 manipulator, marks a substantial step forward. This advancement lays a strong foundation for future research within the lab for enhancing the robot's ability to autonomously navigate and interact with complex environments. It also opens up new possibilities for integrating more sophisticated manipulation tasks, refining the control strategies for better adaptability in dynamic and unpredictable SAR scenarios, and expanding the robot's functionality to tackle increasingly challenging missions.

Keywords: Legged robots, quadruped robotics, Search and Rescue, Nonlinear Model Predictive Control, Whole-Body Control, mobile manipulators, vision-based control, door opening.

UNESCO Codes:

- 120304 Artificial Intelligence
- 120311 Computer software
- 330412 Control devices
- 330417 Real-time systems
- 330419 Robotics
- 330420 Computer vision

Resumen Ejecutivo

El rápido avance de la **robótica** ha revolucionado diversas industrias, especialmente en el ámbito de la **búsqueda y rescate** (SAR, por sus siglas en inglés *Search and Rescue*), donde los robots se están convirtiendo en un tema principal de investigación para su despliegue en entornos peligrosos e impredecibles. Entre estos, las robots **manipuladores con patas** están atrayendo un interés significativo debido a su potencial para realizar tareas que requieren tanto movilidad como manipulación en terrenos complejos y no estructurados. Estas plataformas combinan la agilidad y estabilidad de los robots con patas con la destreza de los manipuladores, lo que las hace especialmente adecuadas para operaciones de SAR donde es esencial la capacidad de atravesar terrenos difíciles e interactuar con objetos.

En la Universidad Politécnica de Madrid, el grupo de investigación RobCib, ha estado a la vanguardia en el desarrollo de sistemas robóticos avanzados para operaciones de SAR. A lo largo de los años, el grupo ha logrado avances significativos en esta área, particularmente con el desarrollo de ARTU-R, un robot cuadrúpedo diseñado para la detección de víctimas en escenarios de SAR. ARTU-R es un proyecto pionero que ha demostrado el potencial de los robots con patas en tareas de búsqueda y rescate, especialmente en entornos donde los robots con ruedas u orugas tendrían dificultades. Sin embargo, a pesar de los éxitos, el proyecto también ha puesto de manifiesto varias limitaciones, en particular en el sistema de control del robot, que depende de **patrones de marcha predefinidos** que no son adecuados para las **condiciones dinámicas** e impredecibles típicas de las misiones SAR. Además, ARTU-R carece de la capacidad de **manipular objetos** con una carga útil superior a 250 g por la limitación del brazo robótico que soporta, lo cual es una función crítica para tareas como abrir puertas o despejar escombros.

Motivado por estos desafíos, la siguiente fase de investigación dentro del grupo RobCib se centró en desarrollar una plataforma robótica más sofisticada y versátil. Esto llevó a la integración de un manipulador más capaz con una plataforma cuadrúpeda robusta.

Este Trabajo de Fin de Máster (TFM) se centra en la **integración y control** de este robot cuadrúpedo manipulador, específicamente diseñada para mejorar las capacidades operativas de los robots SAR. El proyecto involucró la adaptación, implementación y prueba de un sistema de **Control Predictivo Basado en Modelo No Lineal** (**NMPC**, por sus siglas en inglés *Nonlinear Model Predictive Control*) para el robot cuadrúpedo Unitree AlienGo, equipado con un brazo robótico Unitree Z1. El objetivo principal fue crear un sistema de control robusto que pudiera mejorar significativamente las capacidades de locomoción y manipulación del robot en entornos SAR desafiantes, como atravesar terrenos inestables y realizar tareas como abrir puertas.

Figure R1. Cuadrúpedo Unitree AlienGo equipado con brazo manipulador Z1 [46][86].

El control predictivo no lineal es una estrategia de control avanzada utilizada para gestionar sistemas complejos y dinámicos con comportamiento no lineal, como el cuadrúpedo manipulador presentado en este TFM. El NMPC funciona **prediciendo el comportamiento futuro** del sistema en un horizonte temporal finito y optimizando las acciones de control en función de esta predicción. Al resolver de manera continua un **problema de optimización** en cada paso de tiempo, el NMPC permite realizar ajustes en tiempo real en los movimientos e interacciones del robot, garantizando un control suave y preciso en respuesta a entornos dinámicos.

Para alcanzar estos objetivos, se realizó una revisión exhaustiva de las estrategias de control existentes utilizadas en robots con patas. Fue evidente que los sistemas de control basados en generación de patrones de marcha, a menudo diseñados para entornos estructurados, eran insuficientes para las condiciones impredecibles y dinámicas de las operaciones SAR. El sistema NMPC propuesto en este trabajo se diseñó para superar estos desafíos, proporcionando una estrategia de control más flexible y adaptable que pudiera ajustar dinámicamente el comportamiento del robot en respuesta a las cambiantes condiciones de su entorno.

Para desarrollar el sistema NMPC, se utilizó como base un repositorio de código

externo basado en **ROS Noetic**. Sin embargo, este repositorio presentaba varias limitaciones que debían ser abordadas para adaptarlo a los requisitos específicos del cuadrúpedo Unitree AlienGo y el brazo robótico Z1. Las mejoras permitieron adaptar el controlador a la cinemática y dinámica específicas del sistema, lo que permitió su implementación en un sistema más avanzado basado en nodos de ROS que utiliza módulos de visión por computador y planificación, eliminando la necesidad de entrada directa de comandos por parte del usuario.

En este TFM, se dedicó un esfuerzo significativo al desarrollo de un **sistema avanzado de visión por computador** para mejorar la capacidad del robot de interactuar autónomamente con su entorno, un requisito crítico para las operaciones SAR donde el control constante por parte del usuario puede no estar disponible. El sistema de visión se diseñó específicamente para **detectar y manipular manillas de puertas**, permitiendo que el robot realizara tareas complejas como abrir puertas sin intervención humana. Utilizando información obtenida de dos cámaras de profundidad, este sistema permite que el robot se posicione con precisión, detecte y manipule manillas de puertas de manera autónoma en entornos de simulación controlados.

Para validar el sistema NMPC y la integración del manipulador Z1, se realizaron **simulaciones** extensivas utilizando la plataforma **Gazebo**. Estas simulaciones evaluaron el desempeño del robot en diversos escenarios relevantes para SAR, como la navegación en terrenos irregulares, subir escaleras y abrir puertas. Los resultados demostraron mejoras significativas sobre las estrategias de control anteriores, mostrando las capacidades mejoradas del sistema.

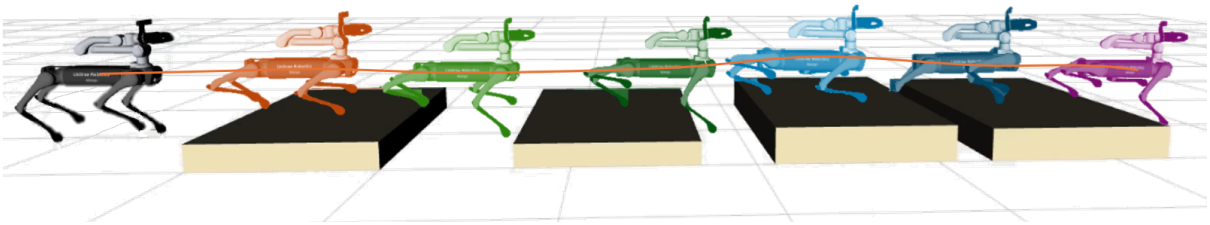


Figure R2. Simulación de camino de bloques.

Además de las simulaciones, se realizaron **pruebas en el mundo real** en entornos controlados para evaluar el desempeño real del sistema. Aunque estuvieron limitadas por restricciones de tiempo y la compartición de recursos, estas pruebas proporcionaron interesantes resultados sobre las capacidades prácticas del robot y señalaron áreas para futuras mejoras.

En general, este TFM representa una mejora sustancial respecto a los métodos de control utilizados previamente por el grupo de investigación para sus robots cuadrúpedos.

La implementación exitosa del sistema NMPC, combinada con el manipulador Z1 integrado con visión, marca un avance significativo. Este avance sienta una base sólida para investigación futura dentro del grupo para mejorar la capacidad del robot de navegar e interactuar autónomamente con entornos complejos. También abre nuevas posibilidades para integrar tareas de manipulación más sofisticadas, refinando las estrategias de control para una mejor adaptabilidad en escenarios SAR dinámicos e impredecibles, y ampliando la funcionalidad del robot para abordar misiones cada vez más desafiantes.

Palabras clave: Robots con patas, robótica cuadrúpeda, búsqueda y rescate, control predictivo basado en modelo, control de cuerpo completo, manipuladores móviles, control basado en visión, apertura de puertas.

Códigos UNESCO:

- 120304 Inteligencia Artificial
- 120311 Software
- 330412 Dispositivos de control
- 330417 Sistemas en tiempo real
- 330419 Robótica
- 330420 Visión por computador

Contents

List of Figures

List of Tables

List of Acronyms

CAR Center for Automation and Robotics. 1

CNNs Convolutional Neural Networks. xx, 23, 24, 27, 32, 39

COCO Common Objects in Context. 41

CoM Center of Mass. 17, 29, 74, 87, 89–92, 96, 99

CPG Central Pattern Generators. 21, 55, 101

CS Combined Systems. 14

DOF Degrees of Freedom. 2, 15, 27, 29, 34, 36, 56, 57, 79

ERL European Robotics League. 12

FCD Full Centroidal Dynamics. 36, 59

GPU Graphics Processing Unit. 23, 42

HDMI High Definition Mutimedia Interface. 28

HPIPM High-Performance Interior-Point Method. 37

iLQR Iterative Linear Quadratic Regulator. 39

IMU Inertial Measurement Unit. 29, 35, 38, 51, 85

IPMs Interior Point Methods. 20

IR Infrared. 30

LF Left Front. 15

LH Left Hind. 15

LiDAR Light Detection and Ranging. 29

mAP Mean Average Precision. 41, 43, 64, 65

ML Machine Learning. 1

MPC Model Predictive Control. xix, 7, 17–20, 24, 25, 27, 32, 101

MR Mixed Reality. 2

NIST National Institute of Standards and Technology. xix, 8, 55, 99

NMPC Nonlinear Model Predictive Control. xvi, xvii, xx, xxi, 3–5, 19, 20, 34–38, 49, 50, 52, 55, 57–60, 73, 77–79, 83, 84, 86–89, 91–97, 99–107, 120

OCS2 Optimal Control for Switched Systems. 39

OOP Object-Oriented Programming. 50

PCA Principal Component Analysis. xx, 44, 46, 47, 69

PCL Point Cloud Library. 44, 67

PD Proportional Derivative. 35, 60

PID Proportional-Integral-Derivative. 58, 86

PLA Polylactic Acid. 81

PPO Proximal Policy Optimization. 25

QP Quadratic Program. 20, 21, 37, 38, 51, 120

RANSAC Random Sample Consensus. xx, 23, 24, 44–46, 67, 69, 95, 98

ReLU Rectified Linear Unit. 40

RF Right Front. 15

RH Right Hind. 15

RL Reinforcement Learning. xix, 3, 22, 23, 25, 55, 106

RMSE Root Mean Squared Error. 85, 100, 103

ROI Region of Interest. 23

ROS Robot Operating System. xvi, xx, xxi, 5, 27, 28, 30, 32–34, 39, 50, 51, 53, 59, 62, 65–70, 73–76, 78–80, 119, 120

SAR Search and Rescue. xv, xix, 1–5, 7–12, 14, 50, 54, 55, 58, 62, 73, 83, 86, 87, 94, 104–106, 119, 125–127

SDG Sustainable Development Goal. xxi, 5, 126, 127

SDK Software Development Kit. 29, 53, 82, 102

SLIP Spring-Loaded Inverted Pendulum. xix, 17, 18

SLQ Sequential Linear Quadratic. 39

SQP Sequential Quadratic Programming. 20, 37

SRBD Single Rigid Body Dynamics. 36, 59

SS Separate Systems. 14

TPU Thermoplastic Polyurethane. 81

UAV Unmanned Aerial Vehicle. xix, 9

UDP User Datagram Protocol. 102, 104

UGV Unmanned Ground Vehicle. xix, 10, 11

URDF Unified Robot Description Format. xx, 39, 51–53, 56, 58, 120

USAR Urban Search and Rescue. xix, 7, 8

USB Universal Serial Bus. 28

USV Unmanned Surface Vehicle. xix, 9

UUV Unmanned Underwater Vehicle. xix, 9

VMC Virtual Model Control. xix, 18

WBC Whole-Body Control. xxiii, 20, 21, 25, 34, 35, 37, 38, 50, 60, 61, 84, 93, 120

WBS Work Breakdown Structure. xvii, xxii, 129, 130

YOLO You Only Look Once. xxiii, 40–43, 62, 63, 67, 106

ZMP Zero Moment Point. xix, 17, 18

List of Symbols

NMPC Formulation Symbols

$\boldsymbol{\tau}_a$	Actuation joint torques vector
$\boldsymbol{\tau}_j^*$	Optimized joint torques vector
$\boldsymbol{d}$	Signed distances between link pairs
$\boldsymbol{f}(\cdot)$	System continuous dynamics function
$\boldsymbol{f}_e$	Contact force vector
$\boldsymbol{f}_e e$	End-effector force
$\boldsymbol{g}_1$	State-input equality constraint function
$\boldsymbol{g}_2$	State-only equality constraint function
$\boldsymbol{h}$	Inequality constraint function
$\boldsymbol{h}_{com}$	Normalized centroidal momentum vector
$\boldsymbol{j}(\cdot)$	System switching dynamics function
$\boldsymbol{K}_d$	WBC differential gain matrix
$\boldsymbol{K}_p$	WBC proportional gain matrix
$\boldsymbol{p}_e$	End-effector position
$\boldsymbol{Q}$	NMPC standard weight matrix
$\boldsymbol{q}$	Generalized coordinates vector
$\boldsymbol{q}_b$	Base pose vector

$\boldsymbol{q}_j$ Joint positions vector

$\boldsymbol{q}_j^*$ Optimized joint positions vector

$\boldsymbol{R}$ NMPC control weight matrix

$\boldsymbol{u}$ Control input vector

$\boldsymbol{u}^*$ Optimized control input vector

$\boldsymbol{u}^{ref}$ Reference control input vector

$\boldsymbol{u}_{\text{input}}$ End-effector force control input

$\boldsymbol{v}_j$ Joint velocities vector

$\boldsymbol{v}_e$ Contact point linear velocity vector

$\boldsymbol{w}_e$ Contact wrench vector

$\boldsymbol{x}$ State vector

$\boldsymbol{x}^*$ Optimized state vector

$\boldsymbol{x}^{ref}$ Reference state vector

$\boldsymbol{x}_0$ Initial state vector

$\boldsymbol{x}_{wbc}$ WBC decision variables vector

δ Inequality constraint smoothing coefficient

ϵ Minimum allowable distance between collision pairs

$\mathcal{S}_{arm}^e$ Set of arm end-effectors

$\mathcal{S}_{foot}^e$ Set of foot end-effectors

μ_s Static friction coefficient

$\Phi(\cdot)$ Mayer cost term

e Contact point

J Cost function

$L(\cdot)$ Lagrangian cost term

n_p Number of possible pairs of robot links

T Final time

t Time

t_0 Initial time

t_i Switching time

v_{swing} Reference swing leg trajectory

Vision Model Performance Metrics Symbols

FN False negatives

FP False positives

n Confidence level

p Precision

r Recall

TP True positives

Camera Intrinsic Parameters

$\boldsymbol{K}$ Camera intrinsic matrix

c_x Principal point x coordinate

c_y Principal point y coordinate

f_x Pixel x-axis focal length

f_y Pixel y-axis focal length

s Skew coefficient

Door Parameters

α Angle between door center line and normal direction

$\hat{\boldsymbol{c}}$ Camera direction vector

$\hat{\boldsymbol{n}}$ Door wall normal vector

ψ Angle between camera direction and normal direction

$d_{\perp_0}$ Desired normal distance to door

$d_{\perp}$ Normal distance to door

d_{door} Distance to the door center

p Pixel offset

R_m Door swing radius in meters

R_p Door swing radius in pixels

Chapter 1

Introduction

The present chapter introduces the developed work, providing a brief contextualization of the project and its motivation. It outlines the work objectives, highlights the main contributions, and describes the structure of this document.

1.1. Background

The field of robotics has seen significant advancements, particularly in the development of robots capable of navigating complex terrains. In particular, **legged robots** have become invaluable in **SAR operations** due to their ability to quickly and safely traverse unstructured environments and access areas that are challenging for traditional wheeled or tracked robots. Over the past decade, these robots have evolved from simple prototypes to sophisticated machines capable of autonomous navigation, object detection, and, to some extent, manipulation.

The RobCib research group, part of the Center for Automation and Robotics (CAR), has conducted significant research in developing robots for SAR applications. Their work has primarily focused on enhancing their autonomy for **exploring and navigating complex terrains**. One of their notable projects is ARTU-R (A1 Rescue Tasks UPM Robot), designed to assist in victim detection using a combination of Machine Learning (ML) techniques and advanced sensor systems [1]. However, the control system of ARTU-R, which relied on predefined gait patterns, posed **challenges in dynamic and unpredictable environments** typical of SAR operations. Additionally, the robot lacked the capability to **manipulate objects**, such as opening doors, which is essential for comprehensive missions.

Another significant research effort within RobCib involved the integration of a 6 Degrees of Freedom (DOF) manipulator with ARTU-R, as shown in Figure 1.1. This project focused on the modeling, simulation, and control of a robotics system combining the quadruped's mobility with the manipulator's dexterity, aiming to enhance the robot's capability to perform tasks such as object manipulation in challenging environments [2, 3]. This project leveraged Mixed Reality (MR) through Hololens glasses to provide an immersive control interface, significantly improving operator efficiency and decision-making in complex environments. The integration allowed for more complex operations, such as **delivering supplies to victims** or **clearing debris**, highlighting the potential of combining legged robots with manipulators for SAR tasks.

Figure 1.1. Unitree A1 and WidowX-250 manipulator assembly [3].

A key challenge identified in both projects was the **need for a more robust control system** that could handle the complexities of real-world SAR environments. The existing control approaches, while effective for structured environments or specific tasks, struggled in scenarios requiring rapid adaptation and decision-making, such as those involving obstacle negotiation or manipulation under uncertain conditions.

1.2. Motivation

Building upon the research previously conducted by the RobCib group, this Master's Thesis aims to address the limitations observed in earlier control approaches for quadruped robots in SAR tasks within the lab. Previous efforts predominantly utilized the smaller A1 quadruped robot, which, despite its effectiveness, relied heavily on predefined **gait patterns**. These patterns, while sufficient for basic mobility, **lacked the adaptability** required for the complex and dynamic environments typical of SAR operations.

To overcome these limitations, this thesis will leverage the larger and more capable **Unitree AlienGo** robot, named KLARA, equipped with the **Z1 robotic arm**. The enhanced capabilities of the AlienGo, combined with the flexibility of the Z1, provide a robust platform for implementing more advanced control strategies. The focus will be on applying **NMPC** to improve the robot's navigation and manipulation skills, particularly in executing tasks like autonomous door opening and movement in unstructured environments. NMPC was chosen for its ability to provide precise control, maintain stability, and handle dynamic constraints in real-time [4].

Reinforcement Learning (RL) control is, together with NMPC, one of the most exploited quadrupedal control methods [5, 6]. It offers promising adaptability and learning capabilities, but also presents challenges such as the "sim-to-real gap" and high computational demands [7] as it will be seen in Chapter 2. These factors make NMPC a more suitable choice for the immediate goals of this project. However, **RL** remains a valuable alternative and will be **explored in a parallel Master's Thesis**.

1.3. Objectives

1.3.1. Main Objective

The primary objective of this Master's Thesis is to **develop and implement a robust NMPC system** for the integrated control of a legged robot, the Unitree AlienGo, with a Z1 manipulator, aimed at enhancing the robot's autonomous navigation and manipulation capabilities in SAR environments. This projects seeks to overcome the limitations of previous control strategies by leveraging the larger and more capable AlienGo platform to achieve reliable and adaptive performance in complex unstructured environments.

1.3.2. Specific Objectives

In order to accomplish this main objective the project has been divided into the following secondary objectives, which can be used as control milestones:

- Analyze the existing control systems used in legged robots and identify the limitations in their application to SAR tasks.
- Design and implement a NMPC strategy for the Unitree AlienGo robot, focused on improving its navigation and obstacle negotiation capabilities.
- Integrate the Z1 manipulator with the AlienGo robot, ensuring seamless coordination between locomotion and manipulation tasks.

- Develop and simulate test scenarios to validate the performance of the integrated system in SAR-relevant tasks, such as autonomous door opening.

- Optimize the NMPC parameters and the control architecture to enhance the robot's adaptability in dynamic and unpredictable environments.

- Implement and evaluate the performance of the system in real-world conditions focusing on its ability to autonomously navigate in unstructured environments.

1.4. Work Contributions

The completion of these objectives has involved significant technical and research achievements. The main contributions of this work are:

- **Development and implementation of a NMPC system:** Building upon an external GitHub repository as the foundational base, a robust NMPC strategy was designed and implemented for the Unitree AlienGo robot. This significantly improved the robot's navigation and obstacle negotiation capabilities, making it more effective in complex, unstructured environments typical of SAR operations.

- **Integration of the Z1 manipulator with the Unitree AlienGo:** The project successfully integrated the Z1 robotic arm with the AlienGo quadruped in simulation, achieving coordinated control between locomotion and manipulation tasks. This integration extended the robot's functionality, enabling it to perform complex operations such as autonomous door opening in SAR environments. Progress towards physical hardware integration was made too, though it was not finished due to time constraints.

- **Simulation and validation in SAR scenarios:** Extensive simulations were conducted using Gazebo to validate the performance of the integrated legged-manipulator platform. These simulations focused on SAR-relevant tasks, offering valuable insights into the system's capabilities and limitations in dynamic and unpredictable environments.

- **Real-world testing in a controlled environment:** Real-world testing was carried out in a constructed path with various obstacles. Although the tests offered valuable data on the controller capabilities, they were limited in scope due to time constraints and the need to share the robot with another project.

1.5. Structure of the Document

The document is structured into six chapters, each addressing a different aspect of the project, and four appendixes including additional documentation.

- **Chapter 1 – Introduction:** Introduces the developed work, contextualizing the project and describing its motivation. It presents the work objectives, the main contributions and the document structure.
- **Chapter 2 – Literature Review:** Examines the existing research and technological advancements related to legged robots and their application in SAR operations. This chapter highlights key control strategies, and the integration of manipulators with quadruped robots.
- **Chapter 3 – Methodology:** Details the tools, techniques, and hardware used throughout the project. This includes the physical components like the Unitree AlienGo and Z1 manipulator, the software tools such as Robot Operating System (ROS) and Gazebo, and the NMPC controller foundational base GitHub repository.
- **Chapter 4 – Development:** Describes the technical implementation of the project, including the system architecture, simulation modeling, NMPC system, vision algorithms for door opening, and planner algorithm. The chapter concludes with system integration, combining all components into a functional whole.
- **Chapter 5 – Results and Discussion:** Presents the outcomes from simulation and real-world testing. It evaluates the robot's performance in non-structured environments and tests the NMPC system's door-opening capabilities. The chapter also analyzes the results, highlighting successes, limitations, and potential improvements.
- **Chapter 6 – Conclusions and Future Work:** Summarizes the outcomes of the project, discussing the achievements, limitations, and potential improvements. This chapter also suggests directions for future research and development.

The appendixes include the developed GitHub repository used in the project; the definition of the controller parameters; the study of economic, social, legal, and environmental impacts, and contribution to Sustainable Development Goals (SDGs); and the temporal planning and budget of the project.

Chapter 2

Literature Review

This chapter reviews the literature on **legged robots**, focusing on their application in **SAR operations**. It covers the evolution and classification of legged robots, with an emphasis on quadrupeds. The **integration of manipulators** with these robots is explored, highlighting their potential to perform complex tasks in unstructured environments. Additionally, the chapter examines model-based and model-free **control strategies**, focusing on Model Predictive Control (MPC). Finally, it reviews how the **door-opening problem** has been addressed in the literature.

2.1. Search and Rescue Robotics

2.1.1. Introduction to SAR Operations

SAR operations are crucial in emergencies, focusing on **locating and assisting people** in danger while **minimizing exposure** to life-threatening conditions [8]. These time-sensitive missions often occur in hazardous environments, such as disaster-stricken areas, where human responders face significant risks. The urgency of SAR operations is underscored by the **critical time window** after a disaster, with survival rates highest within the first 17 hours [9]. However, exceptions exist, such as the 2023 Turkey earthquake, where victims were rescued 248 hours after the event [10].

Disasters are typically classified as **natural** (e.g., earthquakes, floods) or **human-made** (e.g., structural collapses, attacks), each presenting unique challenges [11]. Natural disasters affect large areas, while human-made incidents often occur in confined urban spaces, necessitating Urban Search and Rescue (USAR) efforts [12].

Robotics have become vital in SAR operations, particularly in dangerous or inaccessible situations. Notable examples include the use of robots during the 2001 World Trade Center collapse [13], the 2011 Fukushima nuclear disaster [14], and the 2017 Mexico City earthquake [15]. These robots enhance mission effectiveness and safety by aiding in tasks like victim identification, environmental mapping, and debris removal.

These missions face significant **challenges** due to the hazardous and unpredictable environments in which they occur. These challenges include navigating **unstable terrain**, identifying victims in **low-visibility conditions**, and overcoming obstacles like **debris and structural hazards** [16]. Robotics plays a crucial role in addressing these challenges by combining **autonomy with teleoperation**, enabling precise and adaptive responses in complex scenarios. Advanced sensors and locomotion systems allow robots to perform critical tasks such as environmental mapping, victim detection, and debris clearance, ultimately enhancing the effectiveness and safety of SAR missions.

2.1.2. Classification of Post-Disaster Environments

One of the key developments in SAR operations is the **standardization of post-disaster environments** to guide the appropriate selection and deployment of robotic systems. The National Institute of Standards and Technology (NIST) has defined **three levels** of difficulty based on the structural damage and complexity of the environment [17, 18] as shown in Figure 2.1 [19].

(a) **(b)** **(c)**

Figure 2.1. NIST Test Arenas at the Robocup USAR 2004 Competition : (a) Yellow arena, (b) Orange arena, (c) Red arena [19].

- **Yellow Zone:** This is the least challenging environment, where the terrain consists primarily of doors, shutters, and simple obstacles. The ground is **uniform**, similar to an office setting. In this scenario, robots with basic mobility can fully explore the area, making it suitable for robots with limited agility.
- **Orange Zone:** This medium-difficulty environment features varied ground materials and includes **ramps or stairs**, requiring more advanced maneuverability. Robots operating in this zone must be capable of mapping the environment in real-time, as the terrain can be reconfigured dynamically.

- **Red Zone:** The most complex and unstructured environment, it can undergo real-time changes, such as a secondary collapse. Victims in this zone are often buried under rubble. The ground is not only composed of multiple materials but is also **unstable**, with obstacles like wood, plastic, or rods that can impede movement. Robots operating in this environment must be highly adaptable and capable of navigating these challenging conditions.

2.1.3. Classification of SAR Robots

SAR robots can be classified into several categories based on their operational environment, size, weight, payload capacity, and locomotion system. These classifications are crucial for determining the appropriate robot for specific SAR missions, especially given the varied and challenging conditions found in post-disaster environments.

2.1.3.1. Operational Environment

SAR robots are often categorized based on the environment they are designed to operate in [20]. Some examples can be seen in Figure 2.2.

(a) **(b)** **(c)**

Figure 2.2. SAR robots for different environments: (a) Foldable UAV from UZH [21], (b) EMILY USV [22], (c) Mamba UUV from NTNU [23].

- **Unmanned Aerial Vehicles (UAVs):** These robots provide an aerial view of disaster areas, making them ideal for covering large regions quickly. UAVs are particularly useful in natural disasters like floods or avalanches, where they can rapidly locate victims over expansive areas. However, their flight time and payload are often limited by battery life and weight restrictions.
- **Unmanned Surface Vehicles (USVs) and Unmanned Underwater Vehicles (UUVs):** These robots are designed for operations in aquatic environments. USVs are used for surface-level inspections, such as coastline surveys after tsunamis, while UUVs operate underwater to reach depths unsafe for human divers. Communication and navigation challenges are significant for these robots due to the harsh and often unpredictable underwater conditions.

- **Unmanned Ground Vehicles (UGVs):** Operating on land, UGVs are crucial for navigating debris-strewn environments, such as collapsed buildings in urban areas. These robots vary widely in size and capability, from small, agile units that can squeeze through tight spaces to larger models designed for heavy-duty tasks like debris removal.

2.1.3.2. Size and Weight

Another important classification of SAR robots focuses on their size and weight [24], which directly impacts their mobility and the types of environments they can navigate:

- **Man-packable robots:** These are small, lightweight robots that can be carried and deployed by a single person. Their compact size makes them ideal for navigating narrow spaces within collapsed structures or other confined areas.
- **Man-portable robots:** Slightly larger than man-packable robots, these can be transported by two team members or in small vehicles. They balance portability with increased functionality, such as carrying more sensors or more robust communication equipment.
- **Maxi robots:** These large, robust robots require special transportation. They are typically used in scenarios requiring significant power, such as moving large debris or performing extensive mapping of a disaster area.

2.1.3.3. Payload Capacity

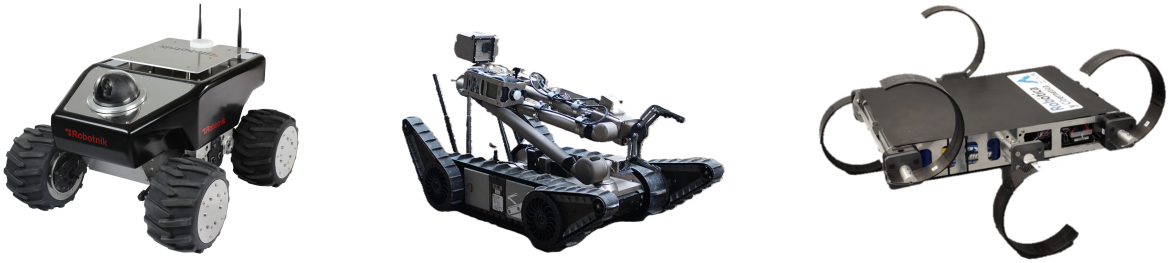
The payload capacity of SAR robots is a critical factor, determining the types and amounts of equipment they can carry, such as cameras, sensors, communication devices, or robotic arms for manipulation. **Robots with higher payload** capacities can carry more advanced equipment, making them suitable for complex tasks like environment mapping [25] or victim detection [26]. **Smaller robots**, with limited payloads, are often used for rapid initial assessments or tasks that require high agility.

2.1.3.4. Locomotion System

The locomotion system of SAR robots is another critical factor, particularly for UGVs, which must navigate various terrains [27]. Figure 2.3 shows some examples.

- **Wheeled robots:** Known for their stability, speed, and energy efficiency, wheeled robots perform well on flat surfaces but struggle with uneven terrain. Some designs incorporate adaptive wheels to improve terrain versatility.
- **Tracked robots:** They excel in rough and uneven environments, such as debris-filled areas, due to their superior traction and stability. This makes them ideal for heavily damaged urban areas.

- **Legged robots:** These robots, inspired by animals, offer unmatched adaptability to complex terrains. Bipedal, quadrupedal, hexapod, and even arachnid-inspired designs allow these robots to traverse uneven, unstable, and cluttered environments. Legged robots are gaining popularity in SAR missions due to their versatility and ability to navigate obstacles that would hinder other robot types.



(a) **(b)** **(c)**

Figure 2.3. SAR UGVs with different locomotion systems: (a) Wheeled SUMMIT-XL from Robotnik [28], (b) Tracked Packbot from iRobot [29], (c) C-legged CLHERO from RobCib [30].

2.2. Quadruped Robots

Quadruped robots have emerged as a promising solution for SAR operations due to their ability to navigate complex, unstructured environments. Inspired by biological counterparts like dogs, these robots excel where wheeled or tracked robots struggle. Compared to bipedal robots, quadrupeds offer greater **stability** and reliability on uneven terrain. While hexapods provide stability, quadrupeds are **faster** and more **energy-efficient**.

2.2.1. Historical Background and Evolution

The development of quadruped robots has a rich history, beginning with early mechanical prototypes in the late 19th century and evolving into today's highly sophisticated machines. The first notable attempt was the "Mechanical Horse" created by Rygg in 1893 [31], which laid the groundwork for later innovations. The mid-20th century saw advancements such as the "Walking Truck" [32], which introduced the concept of **individual leg control**. Significant progress was made in the 1980s and 1990s with the work of Marc Raibert and the MIT Leg Laboratory, leading to the development of dynamic quadruped robots capable of **running and jumping**. The introduction of robots like BigDog [33] by Boston Dynamics in the early 2000s marked a turning point, showcasing the potential of quadruped robots in real-world applications. Recent developments, such as Spot by Boston Dynamics [34] and ANYmal by ETH Zurich [35], represent the current state-of-the-art, featuring advanced sensors, autonomy, and robustness suitable for various challenging environments.

(a) (b) (c)

Figure 2.4. Examples of quadruped robots: (a) BigDog by Boston Dynamics [33], (b) Spot by Boston Dynamics [34], (c) ANYmal by ANYbotics and ETH Zurich [35].

2.2.2. Applications

Quadruped robots have shown significant promise in SAR applications, particularly through their participation in drills and competitions. While **no documented cases** of quadruped robots being deployed in actual SAR missions exist, their potential has been explored extensively in **controlled environments**. For example, the ANYmal robot has been tested in industrial inspection and SAR missions during the ARGOS challenge and the European Robotics League (ERL) Emergency Robots competition [36].

The **DARPA Subterranean Challenge** has been a significant platform for advancing quadruped robots in SAR. In this competition, robots like Boston Dynamic's Spot have been used to explore unstructured, poorly lit environments autonomously [37]. These robots have excelled in navigating difficult terrains, overcoming communication challenges, and identifying objects of interest. The combination of legged robots with aerial capabilities, as demonstrated by **multimodal systems**, further enhances versatility and effectiveness [38].

In addition to SAR missions, quadruped robots have found applications across a variety of other fields due to their versatility. In **industrial settings**, they inspect hazardous environments like pipelines and remote infrastructure [39]. In **scientific exploration**, these robots navigate challenging terrains such as caves and volcanic craters, collecting data where humans cannot [40]. **Agriculture** benefits from their ability to monitor crops and assist in precision farming [41], while in **security and defense** they are used for reconnaissance and surveillance in difficult terrains [42].

2.3. Quadruped Manipulators

The integration of manipulation capabilities into quadruped robots significantly **enhances their functionality**, allowing them to interact with their environment beyond

simple locomotion. By combining the mobility of legged robots with the precision of robotic arms, these machines can navigate challenging terrains and perform complex tasks, making them ideal for environments like disaster zones and rugged landscapes. As research advances, quadruped manipulators are increasingly capable of diverse tasks, from basic pushing to intricate operations using dedicated arms, opening new possibilities in search and rescue, inspection, and agriculture applications.

2.3.1. Manipulation Capabilities

Quadruped robots exhibit a range of manipulation capabilities, depending on their design and intended applications. These capabilities can be broadly categorized into non-grasping manipulation, and manipulation using dedicated arms [43].

2.3.1.1. Non-Grasping Manipulation

Non-grasping manipulation refers to the ability of quadruped robots to interact with their environment without using traditional grippers. Instead, these robots use their **bodies or legs to push, kick, or nudge objects**, which is particularly useful for tasks where precision is less critical, such as clearing debris or repositioning obstacles.

2.3.1.2. Dedicated Manipulation Arm

For more complex and precise tasks, quadruped robots can be equipped with dedicated manipulation arms. These arms, designed with multiple degrees of freedom, enable the robot to perform **delicate operations** like opening doors, picking up objects, or using tools. Integrating these arms requires advanced control systems to maintain the robot's balance and stability, especially when combining locomotion with manipulation. The addition of dedicated arms significantly broadens the functionality of quadruped robots, allowing them to tackle a wider range of applications across various fields. Figure 2.5 shows some examples and Table 2.1 [44] presents some study cases found in the literature.

(a) **(b)** **(c)**

Figure 2.5. Examples of quadruped manipulators: (a) Jueying X20 by Deep Robotics [45], (b) AlienGo with Z1 by Unitree [46], (c) Spot by Boston Dynamics [47].

Quadruped Robot	Quadruped's Weight (kg)	Maximum Payload (kg)	Robotic Arm	DoF	Arm's Weight (kg)	Continuous Payload (kg)	Planning Approach	Applications
A1 [3]	12	5	WidowX 250	6	2.63	0.25	SS	SAR
AlienGo [48]	21.5	13	Unitree Z1	6	4.3	2	SS	Fire rescue
AlienGo [49]	21.5	13	ViperX 300	5	3.63	0.75	CS	Opening doors
ANYmal [50, 51]	30	15	Kinova Jaco 2	6	4.4	1.6	SS/CS	Object carrying
ANYmal C [4]	30	15	DynaArm	4	-	7	CS	Object transport
Self-designed [52]	205	-	Self-designed	5	35	24	SS	Grasping
Spot [53]	32.7	14	Kinova Gen3	7	8	4	SS	Dynamic grasping
Spot [47]	32.7	14	Spot Arm	5	8	5	CS	Object transport

Table 2.1. Quadruped manipulator study cases found in the literature [44].

2.3.2. Integration Approaches

The integration of manipulation capabilities into quadruped robots can be approached in **two main ways**: Separate Systems (SS) and Combined Systems (CS) [54]. The planning approach followed for each study case presented in Table 2.1 is also shown in the table. Each approach offers distinct advantages and challenges depending on the complexity of the tasks and the operational environment.

2.3.2.1. Separate Systems

In the separate systems approach, the quadruped robot's **locomotion and manipulation** functions are treated as **independent subsystems**. The robot first moves into position using its legs, and once stationary, the dedicated manipulation arm or legs perform the required task. This approach **simplifies control** since the movement and manipulation tasks are handled sequentially, reducing the complexity of maintaining balance and stability. However, the robot's ability to perform simultaneous locomotion and manipulation is **limited**, which can be a drawback **in dynamic environments** where both movement and interaction are required concurrently.

Despite its simplicity, this method has notable **limitations**. For instance, poor placement of one subsystem can make the final destination unreachable, and while each subsystem may be optimized individually, this doesn't always lead to a globally optimal solution. Additionally, the approach doesn't fully utilize the system's capabilities, as mutual interferences between the quadruped and the manipulator can be difficult to predict and manage. However, the simplicity of control in this method does offer a stable and robust solution, even if it doesn't leverage the full potential of the system.

2.3.2.2. Combined Systems

The combined systems approach treats the quadruped robot and its manipulation capabilities as a single, **integrated system**. In this approach, locomotion and manipulation are coordinated simultaneously, allowing the robot to move and interact with objects in a more fluid and dynamic manner. This method requires more **sophisticated**

control algorithms to manage the interactions between the robot's body and its manipulator, ensuring stability and balance are maintained throughout the operation. While this approach is more complex, it enables the robot to perform **advanced tasks**, such as navigating rough terrain while carrying or manipulating objects.

This approach, though powerful, comes with its own set of **challenges**. It requires significantly more computing power and energy, and the paths generated are not always optimized, often necessitating additional optimization algorithms. If not properly implemented, solutions can become unstable and less robust to interferences. However, by using the system's full capabilities, it allows for complex movements and multi-tasking, although achieving a robust solution remains difficult, especially without causing instability in the robot.

2.4. Control Strategies

Effective control strategies are crucial for the successful operation of quadruped robots, directly influencing their ability to navigate complex environments and perform diverse tasks. This section explores the key control methodologies employed in quadruped robotics.

2.4.1. Gait and Motion Control

Gait and motion control are fundamental aspects of quadruped robots, directly influencing their stability, efficiency, and adaptability across various terrains. Understanding the principles of gait is essential for optimizing the robot's performance in both static and dynamic environments.

2.4.1.1. Gait

Gait refers to the **coordinated movement pattern** of a quadruped robot's limbs during locomotion [5]. Quadruped robots typically have **12 DOF**, with each leg having three joints. This configuration allows for precise control and complex movements, mimicking the locomotion of animals like dogs or horses. The legs are identified using the nomenclature LH (left hind), RH (right hind), LF (left front), and RF (right front), which standardizes discussions and control strategies.

Selecting the right gait is crucial for effective navigation and maintaining stability across various terrains. Key principles include stride, duty factor, and relative phase. The **stride** is the distance of the torso movement during one gait cycle. Its length and timing are vital for determining the robot's speed and smoothness. The **duty factor** is the percentage of the gait cycle during which a foot stays on the ground, with higher duty

factors enhancing stability but potentially reducing speed. **Relative phase** refers to the timing differences between leg movements, which defines the type of gait, such as walking or trotting.

Gaits are categorized as static or dynamic. **Static gaits**, like walking, keep at least three legs on the ground, providing excellent stability but slower movement, ideal for rough terrains. **Dynamic gaits**, such as trotting or galloping, involve phases where fewer legs are grounded or all are airborne, allowing faster movement but requiring more advanced control to maintain balance.

The choice of gait directly impacts a robot's stability, speed, energy efficiency, and obstacle-handling capability. In dynamic environments, the robot must adapt its gait in real-time based on the terrain and tasks [55]. Effective gait strategies are therefore essential for optimizing the robot's overall performance and ensuring smooth, stable movement across various conditions.

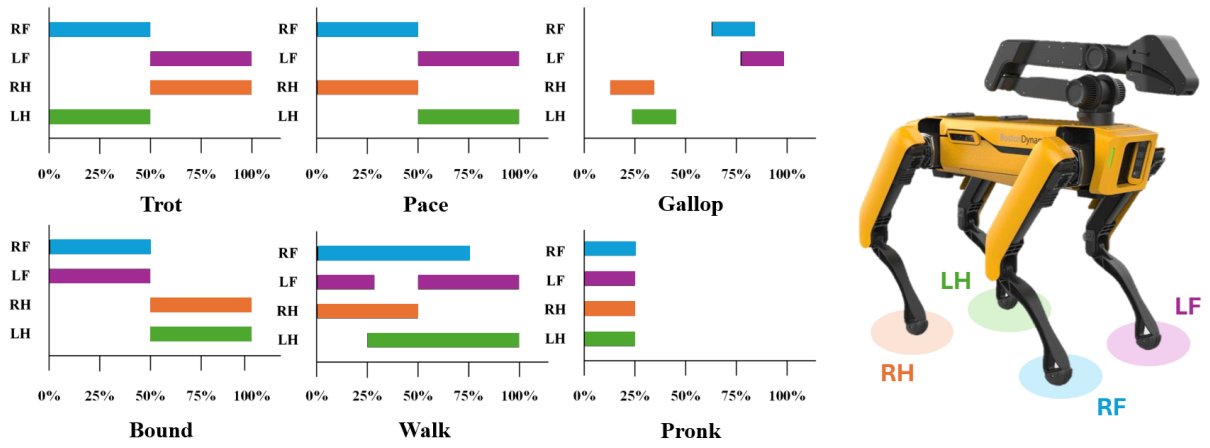


Figure 2.6. Gait graphs for quadrupedal robots, adapted from [5].

2.4.1.2. Motion Control

Motion control in quadruped robots is pivotal for maintaining stability, executing precise movements, and adapting to various terrains and tasks. It encompasses the **algorithms and strategies** that manage **how the robot's legs move** in coordination with each other, responding dynamically to internal and external forces. It is considered to be a tough challenge because the quadruped robot is a **high-dimensional nonlinear time-varying system** with floating base [56]. Motion control methods can be broadly classified into **model-based** and **model-free** approaches, each with distinct advantages and trade-offs. Figure 2.7 represents a generic simplified control architecture usually used to control quadruped manipulators.

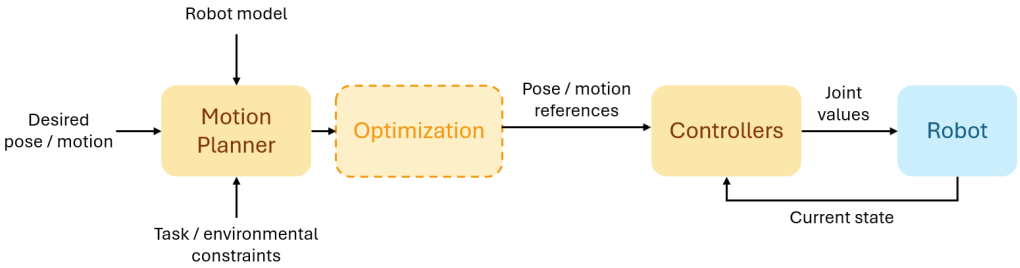


Figure 2.7. Simplified framework block diagram [44].

The system starts with the input of the desired pose or motion, which defines the robot's trajectory. This input feeds into the motion planner, which generates a feasible path considering the robot's model and environmental constraints. The optional optimization block, often integrated within the motion planning process such as in MPC, further refines the trajectory. The controller then converts the optimized trajectory into specific joint commands. A feedback loop provides ongoing updates to the controllers, enabling adaptive control and dynamic responses.

2.4.2. Model-Based Control

Model-based control methods use mathematical models of the robot's dynamics to plan and execute motions. These methods provide high precision and are essential for tasks requiring stability and predictability.

2.4.2.1. Zero Moment Point (ZMP) Control

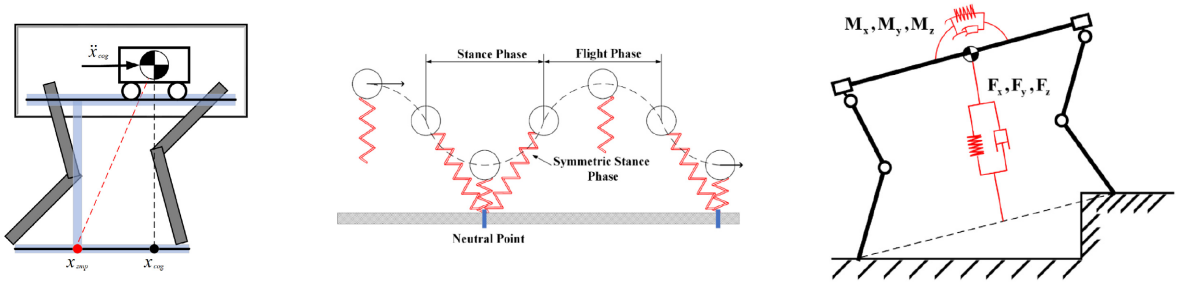
ZMP control is a fundamental method for maintaining the stability of quadruped robots during **static gaits** [57]. The ZMP criterion ensures that the projection of the robot's Center of Mass (CoM) falls within the support polygon formed by its feet. This method is crucial for tasks requiring stable walking, particularly on uneven terrain. By modeling the robot as a cart-table system, the ZMP can be calculated and used to plan stable torso trajectories. Figure 2.8a shows the car-table ZMP model.

2.4.2.2. Spring-Loaded Inverted Pendulum (SLIP) Model

The SLIP model is widely used to control **dynamic gaits**, such as trotting and galloping [58]. It captures the energy dynamics during movement, using a spring-like mechanism to model leg compliance and energy recovery. Raibert's three control method [59], which divides motion into forward movement, jumping, and posture adjustment, is a notable application of the SLIP model. Figure 2.8b shows the SLIP model.

2.4.2.3. Virtual Model Control (VMC)

VMC simplifies the control of complex movements by using **virtual forces and components** like springs and dampers [60], as shown in Figure 2.8c. This intuitive control method allows for effective interaction with the environment, and **reduces the computational demands** while offering flexibility in adjusting control parameters.



(a) (b) (c)

Figure 2.8. Model-based control strategies quadruped models [5]: (a) ZMP model, (b) SLIP model, (c) VMC model.

2.4.2.4. Inverse Dynamics Control

Inverse dynamics control calculates the required joint torques based on desired joint motions and the **robot's dynamic model** [61]. This method is a direct way to achieve **active compliance control**, which can reduce the stiffness of the robot's motion and enhance stability during interaction with external forces. The approach involves solving the dynamic equations of the robot, considering the interaction forces, and adjusting the joint torques accordingly. It is effective but requires **highly accurate modeling** and real-time application might be hard due to its **computational complexity**.

2.4.2.5. Model Predictive Control (MPC)

MPC is an advanced control strategy that has **gained significant traction in the field of robotics** [62, 63], especially for complex systems like quadruped robots, thanks to advancements in computational power. MPC operates by **predicting future states** of the robot based on a dynamic model and **optimizing control inputs** over a **finite time horizon** [64]. This predictive capability allows MPC to handle constraints and make real-time decisions that optimize the robot's performance across various tasks, such as dynamic locomotion and obstacle avoidance. Some of the core principles of MPC are:

- **Prediction horizon:** MPC uses a model of the robot's dynamics to predict future states over a defined horizon, T. At each time step, the control inputs are optimized to minimize a cost function J, which typically includes terms for tracking errors:

$$J = \min_{\boldsymbol{u}(\cdot)} \left[\Phi(\boldsymbol{x}(T)) + \int_0^T L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, dt \right] \tag{2.1}$$

where $\boldsymbol{x}(t)$ represents the state vector; $\boldsymbol{u}(t)$, the control input vector; $L(\cdot)$ is a time-varying running cost or Lagrangian term (ensures optimal performance throughout the process); and $\Phi(\cdot)$ the cost at the terminal state $\boldsymbol{x}(T)$ or Mayer term (ensures the system reaches the desired final state). The optimization process is subject to the system initial state, $\boldsymbol{x}_0$, and the system dynamics defined by function $\boldsymbol{f}$:

$$\begin{cases} \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \\ \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \end{cases} \tag{2.2}$$

- **Handling constraints:** MPC excels in environments where constraints are crucial, such as maintaining stability, avoiding collisions, and ensuring that the robot's actuators operate within safe limits. These constraints can be explicitly included in the optimization problem, ensuring operation within physical limits.

- **Optimization-based approach:** Since the quadruped robot's dynamics and constraints are **inherently nonlinear**, it is a **NMPC optimization problem**. In addition to this, it is a **switched system**, that is, it consists of a finite number of dynamical subsystems subjected to discrete events which cause transition between these subsystems [65], something which happens when walking (different foot contact states). This problem can be formulated according to [4] as:

$$\begin{cases} \min\limits_{\boldsymbol{u}(\cdot)} & \left[\sum\limits_i \phi_i(\boldsymbol{x}(t_{i+1})) + \int_{t_i}^{t_{i+1}} l_i(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, dt \right] & \\ \text{s.t.} & \boldsymbol{x}(t_0) = \boldsymbol{x}_0 & \text{Initial state} \\ & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}_i(\boldsymbol{x}(t), \boldsymbol{u}(t), t) & \text{System flow map} \\ & \boldsymbol{x}(t_{i+1}^+) = \boldsymbol{j}(\boldsymbol{x}(t_{i+1})) & \text{System jump map} \\ & \boldsymbol{g}_{1i}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = 0 & \text{State-input equality constraints} \\ & \boldsymbol{g}_{2i}(\boldsymbol{x}(t), t) = 0 & \text{State-only equality constraints} \\ & \boldsymbol{h}_i(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \geq 0 & \text{Inequality constraints} \\ & \text{for } t_i < t < t_{i+1} \text{ and } i \in \{0, 1, \cdots, I-1\} & \end{cases} \tag{2.3}$$

In the problem, t_i is the switching time, and t_I is the final time. For each mode, the nonlinear cost function consists of a pre-jump cost, $\phi_i(\cdot)$, that indicates the desirability of ending states before a switch occurs; and a cumulative cost over each interval, $l_i(\cdot)$, that accounts for example for the error from a reference trajectory.

The system flow map is driven by the system dynamics $\boldsymbol{f}_i(\cdot)$, and the system jump map, $\boldsymbol{j}(\cdot)$, defines how the state $\boldsymbol{x}$ changes discontinuously at the time of switching t_{i+1}, reflecting changes due to new foot contact state.

NMPC operates on a **receding horizon principle**, where the optimization is performed over a finite future window. **Only the first control input is applied**, and the process is repeated at each time step with updated state information. Figure 2.9 illustrates the receding horizon principle for MPC controllers [66].

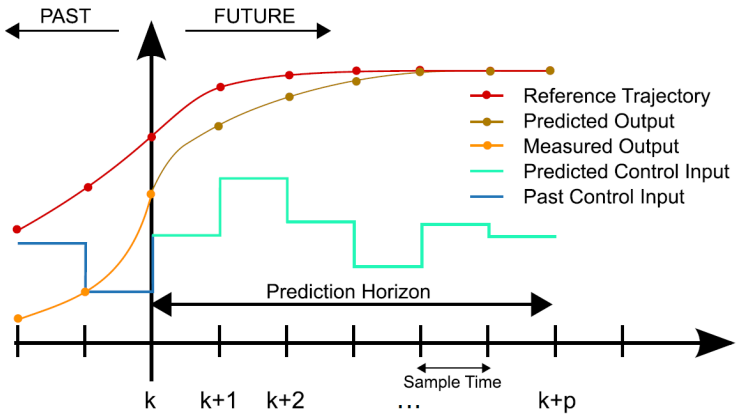


Figure 2.9. Receding horizon principle applied in MPC controllers [66].

To solve this efficiently, the problem is often tackled using **Sequential Quadratic Programming (SQP)** [67]. In SQP, the nonlinear problem is approximated by a series of Quadratic Program (QP) problems [68]. Each QP problem is solved iteratively, with the solution providing an update to the trajectory or control inputs. The SQP process involves the following steps:

1. **Linearization:** Each iteration starts by linearizing the nonlinear dynamics and constraints around the current estimate or trajectory. This involves calculating the derivatives of functions relative to the state and control variables.
2. **Quadratic Programming (QP):** The linearized problem is then solved as a QP. The QP problem typically minimizes a quadratic objective function subject to linear constraints. Different numerical solvers exist but the most commonly used are Interior Point Methods (IPMs).
3. **Iterative Refinement:** The process repeats, with each iteration refining the trajectory and control inputs based on the updated solutions. Convergence is typically determined by thresholds on the norm of the Lagrangian gradient and the feasibility of the constraints.

In advanced quadruped robots, MPC is often combined with **Whole-Body Control (WBC)** [69, 70]. MPC predicts the robot's future states and provides optimized tra-

jectories, while WBC ensures that these trajectories are executed effectively, taking into account the robot's full-body dynamics and task priorities. WBC is a framework designed to manage the simultaneous control of multiple tasks in a robot with a complex structure, such as a quadruped. It integrates the control of locomotion, manipulation, and posture stabilization, allowing the robot to perform coordinated actions across its entire body.

WBC organizes the robot's **tasks into a strict hierarchy** based on their importance. High-priority tasks, such as maintaining balance, are solved first. Lower-priority tasks, such as specific limb movements, are handled afterward in the null space of higher-priority tasks. This ensures that **critical functions are never compromised** [56]. The framework solves these tasks using a cascade of QP problems, where each QP corresponds to a different priority level. The solution to a higher-priority QP shapes the feasible space for the lower-priority tasks.

2.4.3. Model-Free Control

Model-free control strategies are gaining popularity in the field of robotics due to their ability to **adapt to complex and dynamic environments** without relying on detailed mathematical models of the robot's dynamics. Unlike model-based approaches, which require precise system modeling and are often computationally intensive, model-free control methods are generally **more flexible** and can learn directly from data. These approaches are particularly valuable in scenarios where the robot must operate in unpredictable or unstructured environments. The two primary categories within model-free control are **Central Pattern Generators (CPG)** and **intelligent control** methods.

2.4.3.1. Central Pattern Generator (CPG) Control

Central Pattern Generators (CPGs) are inspired by the neural circuits found in animals that **generate rhythmic outputs**. CPG-based control is a bionic approach that constructs oscillators to produce rhythmic motions and coordinate the movement of the robot's legs [71]. This method is highly effective for dynamic motion control, offering simplicity and flexibility in generating various gait patterns. CPG controllers not only synchronize leg movements but also enable smooth gait transitions with minimal control input. It is particularly suitable for rough terrain locomotion, where the ability to adapt to varying conditions is essential. The quadrupeds from RobCib, namely KLARA (walKing Legged sAr Robot) and ARTU-R (A1 Rescue Task UPM Robot), use this locomotion method by default as provided by Unitree. This control method proved to be **insufficient in ensuring stability during testing in harsh conditions**, leading to frequent falls.

2.4.3.2. Intelligent Control Methods

Intelligent control methods leverage computational algorithms to handle the complexity of robotic control without relying on predefined models. This category includes techniques such as fuzzy control, neural networks, genetic algorithms, and RL.

- **Fuzzy control:** Translates sensor inputs and internal states into fuzzy sets, applies a set of fuzzy rules to determine the appropriate actions, and then converts these fuzzy conclusions back into precise control commands [72].
- **Neural networks:** These networks adjust the robot's actions based on learned models from extensive training data, enabling adaptive and efficient responses to dynamic environments [73].
- **Genetic algorithms:** Simulate natural selection processes iteratively adjusting parameters for selecting the most effective strategies over generations based on fitness criteria [74].
- **Reinforcement Learning (RL):** It has become a **leading approach** in developing control policies for quadruped robots, particularly in scenarios requiring **adaptive and robust locomotion** across diverse terrains [75]. By allowing robots to learn through interaction with their environment, RL provides a powerful method for optimizing behaviors without the need for explicit robot dynamic models (though model-based RL also exists).

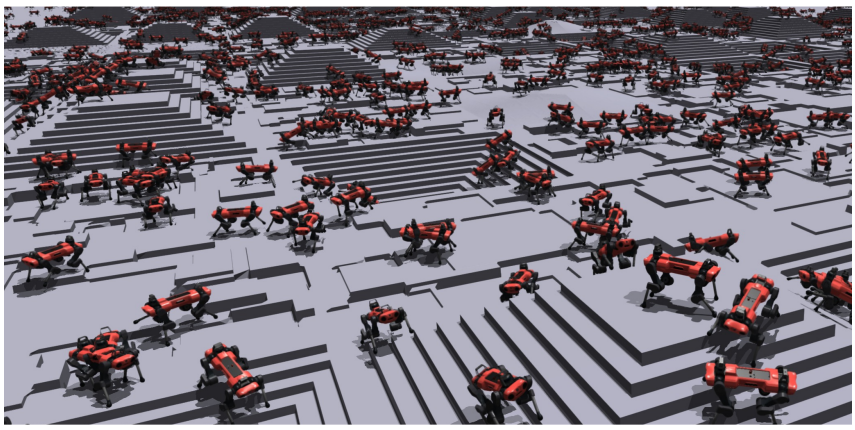


Figure 2.10. ANYmal learning to walk through RL in Isaac Gym [76].

In RL, the robot, or agent, learns by interacting with its environment through a cycle of states, actions, and rewards. The agent observes its current **state**, selects an **action** based on a **policy**[1], performs the action, and then receives feedback in the form of a **reward**. The goal is to learn a policy that maximizes the total accumulated reward over time, guiding the robot to perform complex behaviors.

[1] Strategy that dictates the agent's actions based on its current state.

Deep Reinforcement Learning (Deep RL) has transformed legged locomotion by enabling quadruped robots to **learn complex behaviors** such as walking, running, and jumping. A key advancement in this field is **NVIDIA's Isaac Gym** [76], a GPU-accelerated simulator that drastically reduces the time required for training RL policies, allowing robots to learn basic tasks in minutes [77]. Part of the broader Isaac Lab platform, Isaac Gym provides essential tools for simulating and training robotics systems. Figure 2.10 shows ANYmal learning to walk in Isaac Gym.

While these advancements have accelerated research, they rely heavily on **computational power** rather than breakthrough in RL algorithms. This dependence highlights challenges like the "sim-to-real" gap[2] [7], data inefficiency, and high computational demand. To address these, techniques like "learning-by-cheating"[3] are used to simplify training in stages [78]. Recent innovations have expanded RL applications, enabling quadruped robots to navigate complex terrains, recover from disturbances, and perform tasks like object manipulation.

2.5. Door Opening with Robotic Systems

Autonomous door opening **remains a complex challenge** in robotics, requiring the integration of **advanced perception, control, and manipulation** strategies.

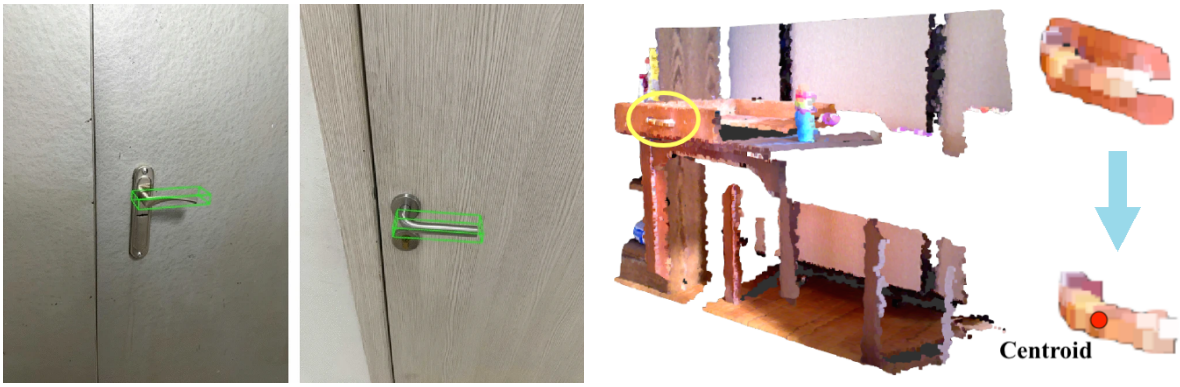
2.5.1. Handle Detection and Grasping Pose Estimation

Robotic door opening often begins with a reliable **detection** and **localization** of the **door handle**. Though traditional methods relied heavily on geometric features and edge detection techniques to identify handles, they often struggled with variability in handle shapes, sizes, and lighting conditions. More recent approaches leverage deep learning, particularly **Convolutional Neural Networks (CNNs)**, to improve the robustness and accuracy of the handle detection [79, 80]. These models are trained on large datasets of images, enabling them to generalize across different environments and handle types. Figure 2.11a shows the 3D detection and localization of a door handle using CNNs.

In addition to detecting the handle, estimating the **correct grasping pose** is crucial for successful manipulation. Point cloud data from RGB-D cameras[4] is commonly used in this process. A popular approach involves filtering the Region of Interest (ROI) around the detected handle using **Random Sample Consensus (RANSAC) algorithm** to

[2] Difference between the robot's behavior in simulation and its performance in the real world.
[3] Imitation learning approach where a RL teacher is first trained with privileged information, followed by deriving a student policy based on visual inputs.
[4] Cameras that capture both color (RGB) images and depth data simultaneously.

differentiate between inliers (door plane) and outliers (handle) as shown in Figure 2.11b [81]. The handle's position is then determined as the centroid of the outliers. This method effectively isolates the handle from the door, allowing the robot to compute the handle's pose accurately relative to the door's plane. The grasping pose is calculated through a transformation matrix derived from the handle's centroid and the door plane's normal vector, enabling precise end-effector placement for grasping.



(a) (b)

Figure 2.11. Handle detection and localization: (a) Handle detection through CNNs [82], (b) RANSAC algorithm application [81].

2.5.2. Model-Based Inference of Door Kinematics

Understanding the **kinematic model** of a door is essential for planning and executing the appropriate manipulation strategy for door opening. Doors generally fall into two primary kinematic categories: **prismatic** (sliding) and **revolute** (hinged), each requiring different parameters for accurate identification and subsequent manipulation [81]. For revolute doors, the model parameters typically include the **center of rotation**, the **radius** of the door's arc, and the **plane of rotation**. For prismatic doors, the focus is on identifying the direction and extent of translational motion.

Identifying which model a door adheres to is crucial for the robot to execute the correct motion. The inference of these kinematic models presents significant challenges due to the presence of noise and outliers in sensor data, which can result from environmental factors or sensor inaccuracies. Once the kinematic model is identified, it serves as the basis for planning the robot's motion.

2.5.3. Control Strategies

Door opening is a complex task for robotic systems, requiring precise coordination, adaptability to different door types, and the ability to manage dynamic environments. Traditional control methods, such as **MPC**, have been widely used for such tasks, but

they face limitations in certain unstructured environments [4, 67, 83]. Recently, **RL** methods, particularly those using **Proximal Policy Optimization (PPO)**, have been explored to address these challenges [79, 84].

Traditional methods like MPC have proved effective in certain scenarios, being **highly reliable in predictable environments**. In door-opening tasks, MPC can generate control inputs that ensure smooth and stable interaction by predicting the door's behavior over a time horizon. When used with a hierarchical WBC, the robot is able to maintain balance while applying the necessary force to open a door. An MPC controller can also be used to track pre-computed offline trajectories instead of acting as a planner [83], which enables more optimized interactions but at the cost of losing real-time processing.

RL PPO algorithm is well-suited for continuous and high-dimensional action spaces typical of robotic manipulation tasks. PPO provides **stable and efficient learning**, making it ideal for tasks requiring precise coordination and adaptability, such as door opening. The state space includes joint positions and velocities, the relative position of the end effector to the door handle, and sensor readings like force and torque. The reward functions are designed to guide the robot in learning effective door-opening strategies such as proximity rewards, force control rewards to avoid damage or task completion rewards.

2.5.4. Advanced Manipulation Techniques

Door manipulation involves challenges related to the high degrees of freedom involved. Researchers have developed the concept of **"documented objects"**, where robots are provided with explicit instructions on how to interact with objects like doors [85]. This concept simplifies the planning process by reducing the search space for possible configurations, enabling more efficient task execution.

Since the kinematic door model parameters might not be completely accurate, **force control and compliance are critical** for managing environmental uncertainties during door manipulation [79]. Advanced techniques such as compliant motion allow robots to adjust their actions based on real-time feedback, ensuring safe and effective interaction with doors. This is particularly important in tasks involving dynamic forces, such as opening a spring-loaded door while maintaining stability during movement.

Chapter 3

Methodology

This chapter presents the hardware and software components used to develop and implement the project. The hardware setup includes the Unitree AlienGo quadruped robot, the Unitree Z1 manipulator, and two depth cameras. On the software side, the ROS framework serves as the backbone, supporting simulation and visualization through tools such as Gazebo and RViz. Additionally, a base MPC GitHub repository is utilized as foundation for the controller development, while CNNs are employed for robot's perception algorithms.

3.1. Hardware Equipment

The main hardware equipment utilized in this project is the Unitree AlienGo robot KLARA. This robot was chosen due to its superior payload capacity and enhanced stability with respect to the Unitree A1 robot ARTU-R. In addition to the quadruped, the Unitree Z1 manipulator will be employed to perform the manipulation tasks.

3.1.1. Unitree AlienGo Quadruped Robot

The AlienGo robot by Unitree Robotics, shown in Figure 3.1, is a quadruped with **12 DOF**, equipped with 12 high-torque motors capable of advanced locomotion tasks. Weighing approximately 20 kg, with a width of 31 cm and a length of 65 cm, the AlienGo **can carry loads up to 13 kg** and achieve stable movement even in challenging environments with a **maximum speed** of around **1.8 m/s** [46]. Among its most notable features are:

Figure 3.1. AlienGo robot [46].

- **Robust structure:** The AlienGo is designed to withstand impacts and maintain stability, even when subjected to external disturbances. This makes it particularly suitable for operations in unstructured environments.
- **Developer's version:** The robot is equipped with two onboard computers that support connections via two High Definition Mutimedia Interface (HDMI) ports, two Ethernet ports, and two Universal Serial Bus (USB) 3.0 ports, enabling a wide range of customizations and remote operations. This version allows developers to program directly in **C/C++** and integrate with **ROS**. Additionally the robot accounts with several power outputs of 5, 12 and 19 V.
- **Advanced Control Architecture:** The AlienGo features a tri-layer control system. It includes a soft real-time x86 control board for sensor setup and execution, a non-real-time ARM board (NVIDIA Jetson TX2) for auxiliary modules, and a hard real-time control board for critical operations. The latter ensures the robot's stability and balance, accessible only via the remote controller. Figure 3.2 shows the robot's architecture scheme.

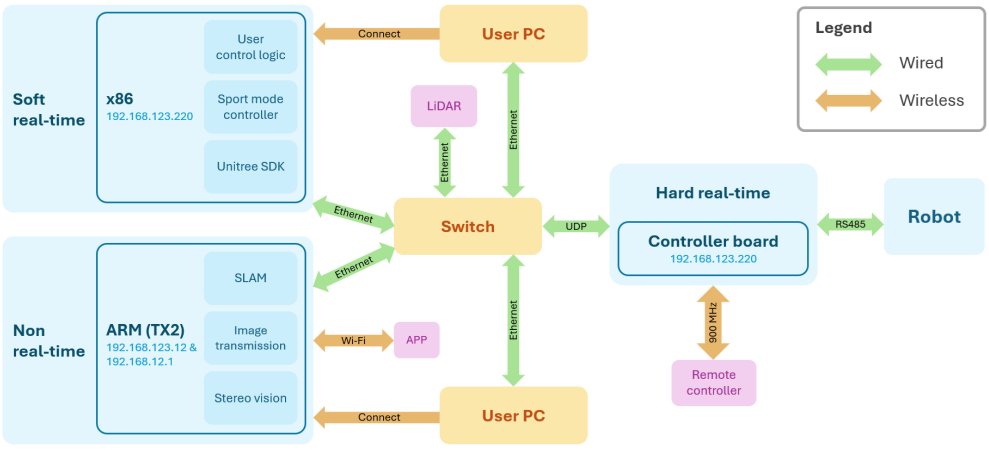


Figure 3.2. AlienGo robot architecture [44].

- **Sport mode:** The robot includes two operational modes, with the Sport Mode 3.0 offering smoother and more stable movement, particularly useful for navigating uneven terrain and overcoming obstacles. This mode enhances the robot's capability to perform dynamic tasks while maintaining balance.

- **Ease of connectivity:** Users can connect to the robot's system either through a Wi-Fi network generated by the robot or via an Ethernet cable, offering flexibility in how the robot is controlled and monitored.

- **Sensors Integration:** The robot accounts with **encoders and gyroscopes on each joint** and **four foot force sensors** to detect whether or not each foot is in contact with the ground. It also has an **Inertial Measurement Unit (IMU)** for measuring the CoM accelerations and angular velocities. Additionally, the robot supports a wide variety of sensors such as **Light Detection and Ranging (LiDAR)**, RGB-D, thermal or multispectral **cameras**. KLARA has two depth and one visual odometer camera. However, for this work, only the RGB-D camera mounted on the Z1 manipulator, the visual odometry camera and one depth D-435 camera connected to the AlienGo will be used as described in the next sections.

- **Extended battery life:** The robot incorporates a 12.600 mAh battery that enables it to operate between 2.5 and 4 hours, depending on its locomotion demands.

3.1.2. Unitree Z1 Manipulator Robot

The Unitree Z1 manipulator was chosen for its integration with the AlienGo quadruped for several reasons. First, because it is specifically designed for seamless integration with Unitree's quadruped robots, ensuring optimal communication, power management, and synchronization. Additionally, it has **more capabilities** than the WidowX 250 robotic arm tested in previous works with ARTU-R [2, 3].

The Unitree Z1 manipulator is a **6-DOF robotic arm**, in contrast with the 5 DOF of the WidowX 250, made of aluminum alloy, ensuring a balance between durability and weight [86]. With a total weight of 4.5 kg, it offers a maximum load capacity ranging from 2 kg in air to 3-5 kg under more stable conditions, making it suitable for a variety of tasks in dynamic environments. The arm has a maximum reach of 730 mm, with a joint range of motion that allows for flexible and precise operations. It is equipped with advanced safety features, including torque and overload protection, making it reliable for use in unpredictable environments. The arm's operation is managed through an Ethernet connection, with power supplied at 24 V and a current requirement of over 20 A. The arm's Software Development Kit (SDK) supports real-time control frequencies of up to 300 Hz. Figure 3.3 shows both the Z1 arm and its integration with the AlienGo.

(a) (b)

Figure 3.3. Integration of Unitree AlienGo quadruped and Z1 manipulator [86]: (a) Unitree Z1 manipulator, (b) Z1 manipulator mounted on AlienGo quadruped.

3.1.3. Depth Cameras

As commented in the previous section, two depth cameras have been used in this work, the SR305 Camera mounted on the Z1 arm and the D435 located on the front of the robot.

3.1.3.1. Intel RealSense SR305 Camera

As it was seen in the literature, many works employ a **RGB-D camera attached to the robotic arm** to perform the manipulation tasks [79, 81]. The Intel RealSense SR305 camera [87] was selected for this project to be attached to the Z1 manipulator for its practicality and **cost-effectiveness**. Although its range is not as extensive as the Intel RealSense D435, the SR305 provides sufficient functionality for successful door-opening tasks. The decision to use the SR305 was also influenced by its availability in the lab.

This depth camera is designed for short-range depth perception, making it suitable for applications where precise and detailed 3D imaging is essential within a confined space, such as detecting and interacting with door handles. The SR305's performance, though optimized for ranges between **0.2 and 1.5 meters**, is adequate for the controlled environments in which the manipulator will operate. Additionally, its compatibility with the existing system and support for ROS made it an easy choice for the project's needs.

It is equipped with two main types of sensors: a color sensor and an infrared IR sensor. These sensors work together to provide both depth and color information. The **color sensor** captures **1920×1080** high-resolution RGB images, providing detailed color information about the environment. The **IR sensor**, in conjunction with an IR projector, is responsible for capturing depth information by projecting an infrared light pattern onto the environment. The reflected IR light is then captured by the IR sensor, and the

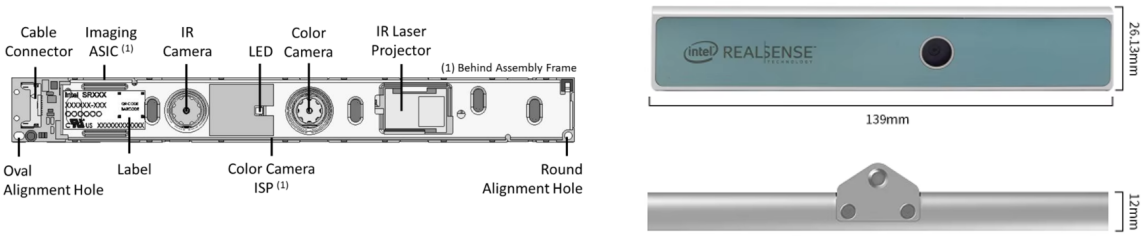


Figure 3.4. Intel RealSense SR305 Camera [87].

distortion in the light pattern is used to calculate the distance between the camera and the objects in its field of view, resulting in a **640×480 depth image**. This camera will be used for close look at the door handle height and its ideal for tracking the object to be manipulated.

3.1.3.2. Intel RealSense D435 Camera

The Intel RealSense D435 camera [87] was selected to be mounted on the robot's base for its excellent depth perception capabilities over a wider field of view and longer range compared to the SR305 model. This model is essential for the project's requirements of accurately assessing the base's position during locomotion tasks. The D435 employs stereo vision technology with two global shutter sensors, capturing depth and RGB information simultaneously, making it highly efficient for real-time applications.

This depth camera provides a significantly larger depth field of view, with a diagonal FOV exceeding 90 and supports depth streaming at resolutions up to **1280×720** with a frame rate of 90 FPS. It is capable of capturing RGB images at **1920×1080** resolution. The D435 can operate effectively over a depth range of **0.2 meters to more than 10 meters**, depending on lighting conditions, which makes it suitable for environments where varying distances and object sizes are involved [121]. Figure 3.5 shows the employed camera and its location within the quadruped.

Figure 3.5. Intel RealSense D435 Camera [121].

The Intel RealSense D435's broad depth perception capabilities are particularly useful for loco-manipulation tasks, such as door opening, where it is crucial to accurately perceive

the door's dimensions and the manipulator's relation to it. Its compatibility with the ROS system and Intel RealSense SDK, as well as its support for USB 3.1, makes it easily integrable with the existing robot architecture. This camera will be mounted at the base of the robot, complementing the SR305 mounted on the arm, and ensuring that visual feedback is available for both manipulation and locomotion tasks.

3.2. Software Tools

The software tools used in this project are essential for integrating and executing the robotic system's control and vision tasks. The tools are divided into three key areas: ROS for system integration and simulation, the base MPC repository used as foundation for robot control, and CNNs for vision tasks.

3.2.1. Robot Operating System (ROS)

ROS is an open-source framework widely used for developing robotic applications [88]. It was first introduced in 2007 by Eric Berger and Keenan Wyrobel at Stanford University and has since become the de facto standard for robotic system implementation, particularly in research and increasingly in industry due to its flexibility and wide array of tools [89]. At its core, ROS operates on a system of nodes, each of which handles a specific function within the robot. These nodes communicate through a publish-subscribe model using **topics** for message passing, allowing data to flow asynchronously between different parts of the system. Additionally, **services** enable synchronous communication, where a node can send a request and wait for a response, which is crucial for tasks requiring immediate feedback or control.

ROS is **language-agnostic**, meaning nodes can be implemented in various programming languages, such as C++ or Python, allowing flexibility in development. In this project, the nodes handling **vision-related tasks** have been implemented in **Python**, while the rest of nodes, particularly those related to **control and motion**, are written in **C++**. This division is justified by the strengths of each language: Python offers ease of use and rapid development, making it ideal for processing image data and integrating machine learning models, while C++ provides performance and fine-grained control necessary for real-time robotic control tasks.

The choice of **ROS Noetic**, even with ROS 2 available since 2017, is strategic. It is the final release of ROS 1, providing extensive documentation, stable support, and a broad user base. This version is also compatible with the existing MPC repository used in the project and aligns with the tools and previous work done within the lab, ensuring a smooth development process. Several key ROS tools are employed in this project.

Gazebo is a powerful simulation tool within the ROS ecosystem that provides a highly realistic environment for testing and validating robotic systems [90]. It enables developers to simulate complex scenarios and supports **physics-based simulations**, allowing for accurate modeling of sensor data, actuator dynamics, and environmental interactions. Gazebo additionally offers the possibility to add plugins and even to create new ones.

RViz is a crucial **visualization tool** within the ROS framework that allows developers to interactively monitor and analyze the robot's state and sensor data in real-time [91]. It provides a flexible and powerful environment for visualizing various types of data, including sensor outputs, robot model states, and trajectory plans, making it an indispensable tool for debugging and development. Moreover, RViz provides **tools to interact** with the robot's control systems, such as setting target poses, sending commands, and visualizing sensor outputs like point clouds and images. This interactive capability is crucial for fine-tuning control algorithms and ensuring that the robot behaves as expected under different scenarios, whether in simulation or real-world operation. Figure 3.6 shows a screenshot of the RViz display during development.

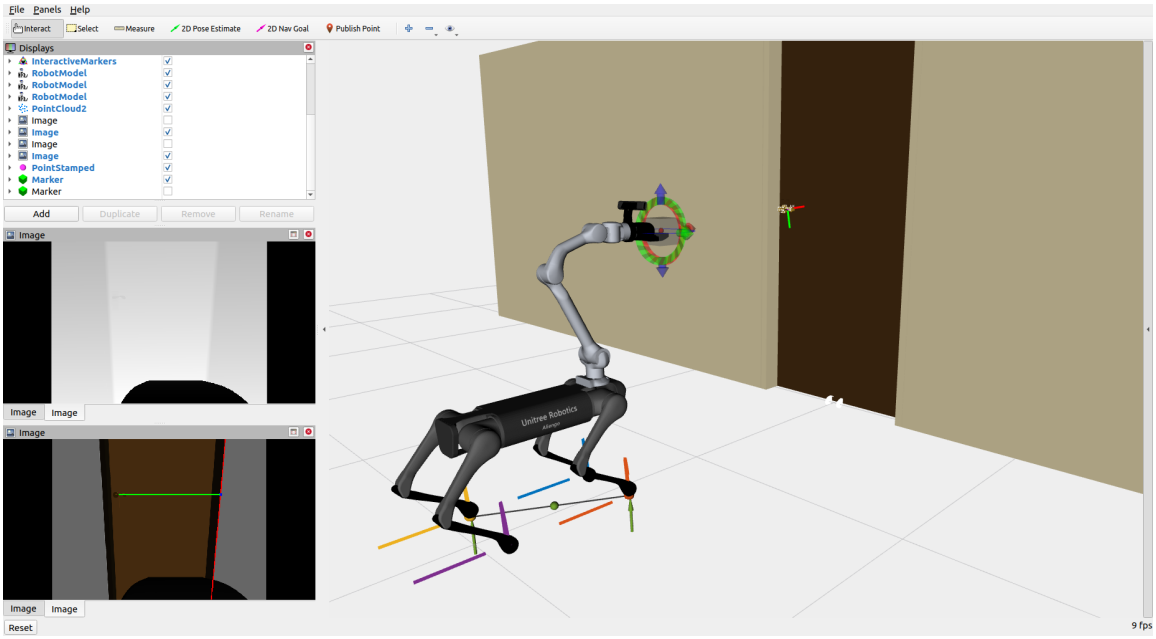


Figure 3.6. RViz screenshot showing simulated camera color and depth images, handle point cloud and centroid, and some additional markers.

Finally, **rosbag** is a powerful tool within the ROS ecosystem used for **recording and playing back** ROS message data [92]. It allows developers to capture all messages published over ROS topics during a robot's operation and save them to a file, known as a "bag" file. This capability is especially useful for **debugging, testing, and analysis**, as it enables the recreation of specific scenarios of the analysis of sensor data without needing to rerun the entire robotic system.

3.2.2. Base NMPC Controller Repository

The decision to utilize a combined approach of NMPC with WBC for the integrated control of both the arm and the quadruped systems—specifically the AlienGo Unitree quadruped coupled with the Unitree Z1 manipulator—stems from the need for precise and dynamic interaction within complex environments. NMPC provides optimal control planning over a finite horizon of the whole assembly, considering future states to make informed decisions, while WBC facilitates effective management of multiple tasks.

Given the complexity inherent in implementing this advanced controller, it was deemed necessary to **build upon existing code foundations**. This approach allows for focusing on the development of more advanced applications by leveraging proven frameworks that facilitate stability and reliability. The project initially utilized a repository featuring a **NMPC-WBC controller** specifically designed for the A1 Unitree quadruped without any arm [93]. This repository used **ROS Noetic** and provided a solid foundation for testing basic functioning and feasibility and is based on the work done at ETH Zurich [4, 56, 67, 94].

As the project evolved, there was a transition to a more advanced repository [95]. This new repository built upon the initial one but included extension for the **AlienGo** platform coupled with the **Kinova Jaco v2 6-DOF arm** equipped with three fingers. This upgrade was critical for exploring manipulative tasks. The system framework architecture can be seen in Figure 3.7. The repository is still under development but some of the tasks it has achieved are whole-body planning, **end-effector motion tracking**, stability with force disturbance, and whole-body compliance control [96]. In addition to this, one of the repository branches was focused on **real hardware implementation**, setting a robust base for the integration of the Z1 arm and the addition of vision and planner modules. These enhancements are pivotal for executing more complex tasks such as door opening.

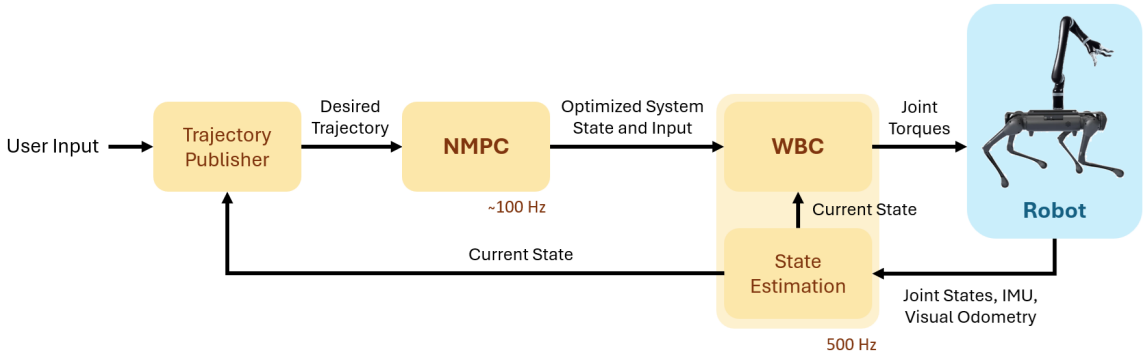


Figure 3.7. Base GitHub repository NMPC controller [93, 95, 96].

As shown in Figure 3.7, once the user sets the desired end-effector position, it is converted to a state trajectory (base and end effector target poses) and then sent to the

NMPC. The NMPC evaluates the optimized system state and input to the WBC. The WBC then figures out the joint torques according to the optimized states and inputs from the NMPC according to a hierarchy of tasks. The torque is set as a feed-forward term and is sent to the robot's motor controller. Low-gain joint-space position and velocity Proportional Derivative (PD) commands are sent to the robot's motors to reduce the shock during foot contact and for better tracking performance. Both the NMPC and WBC need to know the current robot state, the base orientation and joint state, all obtained directly from the IMU and the motor measurements. Running in the same loop with WBC, a linear Kalman filter estimates the base position and velocity from base orientation, base acceleration, and joint foot position measurement [94].

3.2.2.1. Trajectory Publisher Implementation

The trajectories publisher acts as a state and input **reference generator** ($\boldsymbol{x}^{ref}$ and $\boldsymbol{u}^{ref}$) which employs the end-effector position set by the user, the current state of the robot and the gait schedule timing. The position of the end-effector can be directly modified from an interactive marker in RViz. A total of **11 gait schedules** are configured with their respective switching times, including stance, trot, standing trot, flying trot, dynamic walk, etc.; and it is possible to switch from one to another from the terminal console.

3.2.2.2. NMPC Implementation

The model used in this NMPC problem for the robot is a **centroidal model**, which is particularly advantageous for legged robots. This model integrates both kinematic and dynamic aspects but simplifies the dynamics to enhance computational efficiency. Specifically, it treats the robot as a poly-articulated floating-base system with an unactuated 3D rigid body core and fully actuated limbs, as it can be seen in Figure 3.8.

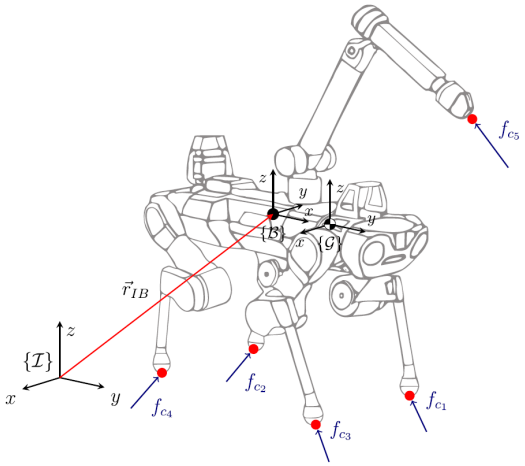


Figure 3.8. Centroidal multi-limbed floating-base model with $\mathcal{I}$, $\mathcal{B}$, and $\mathcal{G}$ being the inertial, base and centroidal frames respectively [4].

The system dynamics within this framework can be modeled in two distinct ways, selectable via a configuration file: **Single Rigid Body Dynamics (SRBD)**, that simplifies the dynamics assuming constant inertia regardless of the joint positions and massless limbs, and **Full Centroidal Dynamics (FCD)**, which fully integrates the motion of the robot's limbs into the centroidal dynamics calculations [65, 97]. The SRBD model works well if the limbs have small mass compared to the total robot mass.

The robot has a total of 18 DOF, 12 corresponding to the quadruped and 6 of the Kinova arm (which coincides with the DOF of the Z1 arm). This way, the optimal control problem is formulated according to 2.3 with the system state $\boldsymbol{x}$ and input $\boldsymbol{u}$ defined as:

$$\boldsymbol{x} = [\boldsymbol{h}_{com}^T, \boldsymbol{q}_b^T, \boldsymbol{q}_j^T]^T, \;\; \boldsymbol{u} = [\boldsymbol{w}_e^T, \boldsymbol{v}_j^T]^T, \tag{3.1}$$

where $\boldsymbol{h}_{com} \in \mathbb{R}^6$ is the collection of the normalized centroidal momentum, $\boldsymbol{q}_b \in \mathbb{R}^6$ is the base pose and $\boldsymbol{q}_j \in \mathbb{R}^{18}$ are the joint positions. $\boldsymbol{w}_e \in \mathbb{R}^{18}$ are the contact wrenches acting on the end-effectors (note that it is assumed that the foot contacts are only subjected to 3-DOF forces and the manipulator end-effector is subjected to a 6-DOF wrench since it can also suffer a torque actuation) and $\boldsymbol{v}_j \in \mathbb{R}^{18}$ are the joint velocities.

Cost Function

The NMPC cost formulation is simple and just defined by the **quadratic cost of tracking the error** of all states and the input:

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{u}, t) &= \|\boldsymbol{x} - \boldsymbol{x}^{ref}\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u} - \boldsymbol{u}^{ref}\|_{\boldsymbol{R}}^2 \\ \Phi(\boldsymbol{x}) &= \|\boldsymbol{x}_I - \boldsymbol{x}_I^{ref}\|_{\boldsymbol{Q}}^2, \end{aligned} \tag{3.2}$$

where $\boldsymbol{Q} \in \mathbb{R}^{30\times30}$ is the standard weight matrix, and $\boldsymbol{R} \in \mathbb{R}^{36\times36}$ is the control weight matrix; both positive definite matrices that act as **important tuning parameters** of the NMPC planner.

Constraints

With respect to the constraints, four types of state-input equality constraints and four types of inequality constraints are implemented. The state-input equality constraints are the following:

$$\begin{cases} \boldsymbol{v}_{e_i} = \boldsymbol{0} \;\mid\; e_i \in \mathcal{S}_{foot}^e & \text{Zero closed foot contact velocities} \quad (3.3a) \\ \boldsymbol{w}_{e_i} = \boldsymbol{0} & \text{Zero open contact wrench} \quad (3.3b) \\ v_{e_i}^z - v_{swing}(t) = 0 \;\mid\; e_i \in \mathcal{S}_{foot}^e & \text{Swinging foot contact trajectory} \quad (3.3c) \\ \boldsymbol{p}_{e_i} - \boldsymbol{p}_{\text{desired}} = \boldsymbol{0} \;\mid\; e_i \in \mathcal{S}_{arm}^e & \text{End-effector position constraint} \quad (3.3d) \end{cases}$$

where $\boldsymbol{v}_{e_i}$ is the linear velocity of contact point e_i expressed in the inertial frame $\mathcal{I}$, and $\mathcal{S}^e_{foot}$ is the set of foot end-effectors. $\mathcal{S}^e_{arm}$ is the set of arm end-effectors, and $\boldsymbol{p}_{e_i}$ and $\boldsymbol{p}_{\text{desired}}$ are the current and desired end-effector positions, respectively. The equality constraints have the following implications: the foot of a stance leg should not separate or slip with respect to the ground (3.3a), wrenches $\boldsymbol{w}_{e_i}$ at open contacts vanish (3.3b), all swing legs should track a reference trajectory $v_{swing}(t)$ along the surface normal (3.3c), and the end-effector position tracks the desired one [4]. The inequality constraints are the following:

$$\begin{cases} \boldsymbol{q}_j^{min} \leq \boldsymbol{q}_j \leq \boldsymbol{q}_j^{max} & \text{Joint positions operational limits} \quad (3.4a) \\ \boldsymbol{v}_j^{min} \leq \boldsymbol{v}_j \leq \boldsymbol{v}_j^{max} & \text{Joint velocities operational limits} \quad (3.4b) \\ \boldsymbol{d}(\boldsymbol{x}) - \epsilon \cdot \mathbf{1}_{n_p \times 1} \geq \mathbf{0}_{n_p \times 1} & \text{Robot self-collisions avoidance} \quad (3.4c) \\ \mu_s f_{e_i}^z - \sqrt{{f_{e_i}^x}^2 + {f_{e_i}^y}^2 + \delta^2} \geq 0 & \text{End-effector force inside friction cone} \quad (3.4d) \end{cases}$$

where $\boldsymbol{d}(\boldsymbol{x}) \in \mathbb{R}^{n_p}$ are the signed distances between n_p pairs of links that are represented with primitive collision bodies, and $\epsilon \geq 0$ is the minimum allowable distance between collision pairs [83]. μ_s is the static friction coefficient and $\delta \neq 0$ is needed to smoothen the friction constraints. The inequality constraints have the following implications: the joint positions must not overpass their physical limits (3.4a), the same applies to the joint velocities (3.4b), robot's self-collisions are avoided with a certain margin ϵ (3.4c), and finally, the forces acting on the feet lie in their respective friction cones, that is, the maximum static friction is not overpassed (3.4d).

Optimal Control Problem Solving

To solve this optimal control problem, a direct multiple shooting method is formulated to transcribe the optimal control problem to a non linear program problem, which is solved using SQP. The QP subproblem is solved using the High-Performance Interior-Point Method (HPIPM) [98]. The outputs of the NMPC planner are the optimized state and control input, $\boldsymbol{x}^*$ and $\boldsymbol{u}^*$, which are passed to the WBC controller. The **time horizon** of the NMPC planner is $T = 1$ s, and the loop computes feedforward trajectories at an average update rate of **100 Hz**.

3.2.2.3. WBC and State Estimator Implementation

The optimal reference plans for the base and limbs are tracked by a WBC controller that tries to fulfill a set of prioritized tasks. These tasks are formulated as a **hierarchical QP** that optimizes for the generalized accelerations and contact forces. The purpose of the WBC is not to directly track the optimal ground reaction forces but to capture their influence by tracking the reference motion they induce on the base to keep the robot

balanced [4]. This is because the WBC adjusts the optimized forces from the NMPC if they violate any high priority objectives. The only exception are the arm contact forces since the WBC has no knowledge of the manipulated object's dynamics. WBC only considers the current moment, and each task corresponds to equality or inequality constraints dependent on decision variables. The decision variables for the WBC are:

$$\boldsymbol{x}_{wbc} = \left[\ddot{\boldsymbol{q}}^T, \boldsymbol{w}_e^T\right]^T, \tag{3.5}$$

where $\ddot{\boldsymbol{q}} = [\ddot{\boldsymbol{q}}_b, \ddot{\boldsymbol{q}}_j]^T \in \mathbb{R}^{24}$ are the accelerations of the generalized coordinates. The computation of these decision variables is made through a series of calculations from $\boldsymbol{x}^*$ and $\boldsymbol{u}^*$ [56]. Table 3.1 shows the WBC prioritized tasks list.

Priority	Type	Task
0	=	Floating base equations of motion
	≥	Torque limits
	≥	Friction cone constraints
	=	No motion at the contact points
1	=	Base motion tracking
	=	Swing feet trajectory tracking
2	=	Arm joint motion tracking
	=	Contact force tracking

Table 3.1. WBC prioritized tasks list [93, 96].

The WBC solves the QP problem in the null space of the higher priority tasks' linear constraints and tries to minimize the slacking variables of inequality constraints. This approach can consider the full nonlinear rigid body dynamics and ensure strict hierarchy results with good real-time performance [56]. The outputs from the WBC are the optimized joint position, velocity and torque references for the low-level control module: $\boldsymbol{q}_j^*$, $\dot{\boldsymbol{q}}_j^*$, and $\boldsymbol{\tau}_j^*$, respectively. The actuator torque commands, $\boldsymbol{\tau}_a$, are generated by the low-level control module running at 2 kHz, according to Equation 3.6:

$$\boldsymbol{\tau}_a = \boldsymbol{\tau}_j^* + \boldsymbol{K}_p(\boldsymbol{q}_j^* - \boldsymbol{q}_j) + \boldsymbol{K}_d(\dot{\boldsymbol{q}}_j^* - \dot{\boldsymbol{q}}_j), \tag{3.6}$$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are matrices with control tuning parameters corresponding to the proportional and differential gains of each of the 18 joints of the quadruped manipulator.

The **state estimator** consists of a **Kalman filter** which fuses the motor encoder readings, the visual odometry from the AlienGo front camera, and the IMU measurements to estimate the base pose and angular velocities, and the joint positions and velocities. This state estimation is fed into both the whole body planner and the controller. The WBC, along with the state estimator constitute the main control loop running at 500 Hz.

3.2.2.4. External Libraries and Dependencies

The repository leverages powerful external libraries to solve optimal control problems, handle complex dynamics and ensure efficient collision management and detection.

- **Optimal Control for Switched Systems (OCS2):** It is a robust C++ toolbox that facilitates the **solution of optimal control problems** for systems with switchable dynamics [65]. It incorporates advanced algorithms such as Sequential Linear Quadratic (SLQ) for continuous-time domain and Iterative Linear Quadratic Regulator (iLQR) for discrete-time domain constraints. OCS2 excels in managing path constraints using Augmented Lagrangian or relaxed barrier methods, making it highly suitable for robotic tasks that demand the integration of complex kinematic and dynamic models from Unified Robot Description Format (URDF) files. Its capability to interface seamlessly with ROS enhances its utility in real-time robotic applications, especially on platforms with limited computational power.

- **Pinocchio:** It is an open-source library [99] focused on providing fast and flexible implementations of **rigid body dynamics algorithms** and their derivatives [100]. It is built on the Eigen library [101] for linear algebra and seamlessly integrates with the Flexible Collision Library for efficient collision detection [102]. Pinocchio is crucial for simulating articulated robotic systems, offering rapid kinematics and dynamics computations.

- **HPP-FCL:** It is an extension for the Flexible Collision Library [102], offering **enhanced collision detection capabilities** tailored for robotics applications [103]. It features an efficient implementation of the GJK (Gilbert-Johnson-Keerthi) algorithm [104], supports safety margins for collision detection, and incorporates accelerated collision techniques to improve performance.

3.2.3. Convolutional Neural Networks

CNNs are a specialized type of artificial neural network designed for processing and analyzing visual data, making them particularly effective for **image-related tasks**. Unlike traditional neural networks, CNNs leverage the spatial structure of images, taking into account the relationships between pixels by applying **convolutional operations**, as shown in Figure 3.9 [105]. These operations involve passing the image through **filters** (or kernels) that detect patterns such as edges, textures, or even more complex features as the network deepens. The input to a CNN is typically a tensor representing the image, with dimensions corresponding to height, width, and the number of channels (e.g., three channels for RGB images).

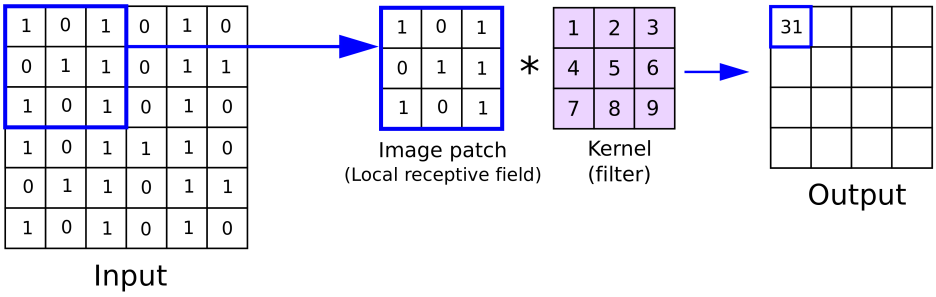


Figure 3.9. Convolution operation explanation [105].

The CNN architecture consists of several key components. Initially, convolutional layers apply filters to the input image, producing feature maps that highlight specific patterns. These are followed by **activation functions**, commonly Rectified Linear Unit (ReLU), which introduces non-linearity into the model [106]. To reduce the dimensionality and computational load, **pooling layers** are employed. These layers typically perform max-pooling, which retains only the most prominent features by selecting the maximum value from a subset of the feature map. This process is repeated in multiple layers, progressively extracting more abstract features as the image passes through the network. Eventually, **fully connected layers** at the end of the network combine these features to make a final prediction, such as classifying the image or detecting objects within it. A diagram with the whole generic architecture is shown in Figure 3.10 [107].

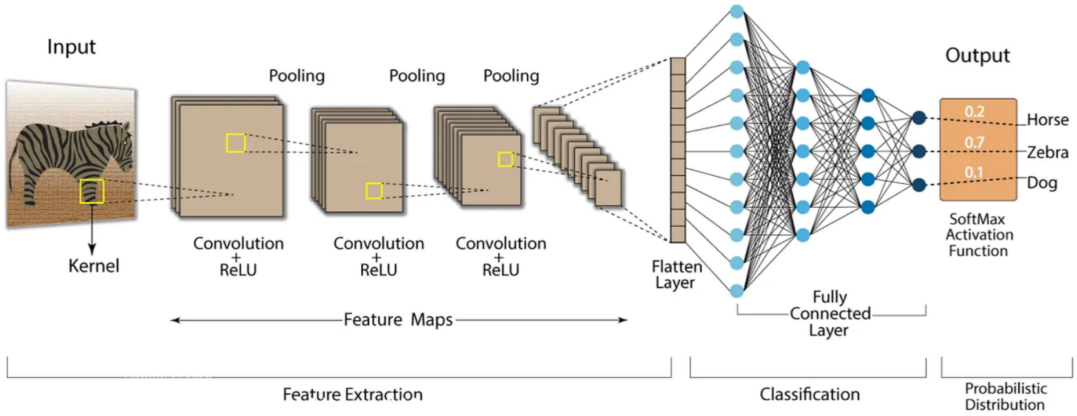


Figure 3.10. CNN architecture [107].

3.2.3.1. YOLO

For this project, the **YOLOv8** (You Only Look Once version 8) model is used, which is a state-of-the-art CNN designed for **real-time object detection, localization** and **segmentation** tasks [108]. It was chosen over other architectures due to its exceptional accuracy and high-speed processing, in addition to its robust community and developer support. YOLOv8 stands out for its ability to simultaneously predict bounding boxes, class labels, and pixel-level segmentation masks. This makes it particularly suitable for

applications that require not just detecting objects but also understanding their shapes and boundaries within the image. For the task of door opening, segmentation is critical, as it allows the robot to accurately identify and differentiate between the door, the handle, and the surrounding environment, enabling precise manipulation.

YOLOv8, developed by Ultralytics [109], continues the legacy of the YOLO series, which began in 2015. The model divides images into grids and predicts the presence of objects within these grids [110]. These models are pre-trained on the Common Objects in Context (COCO) dataset, which includes 330.000 images and 80 different object categories. This **pre-trained weights** can be taken as a starting point to **fine-tune** the model with a custom dataset to obtain better results and faster implementation than if it was trained from scratch. Several hyperparameters can be set for the training, such as the number of **epochs**, which is the number of cycles the neural network trains over the entire dataset. Different YOLO models are available for **specific vision problems** such as segmentation (`yolo8vs-seg`), classification (`yolo8vs-cls`), detection (`yolo8vs`), tracking (`yolo8vs-obb`) and pose estimation (`yolo8vs-pose`), shown in Figure 3.11.

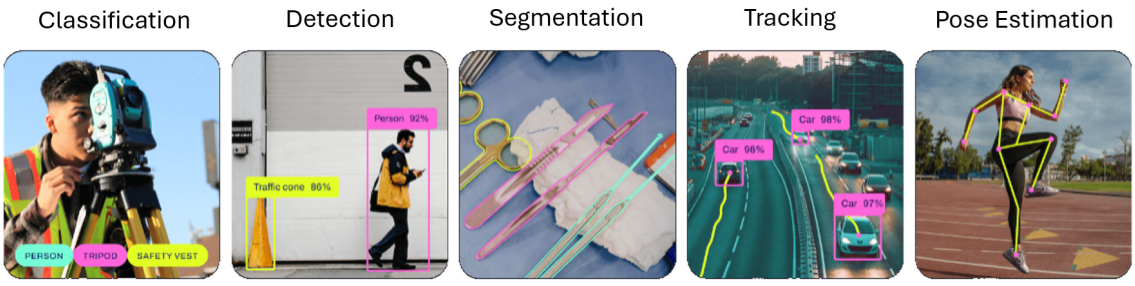


Figure 3.11. Computer vision problems [109].

For each of these specific tasks, different **model sizes** (`n`, `s`, `m`, `l`, and `x`) are available depending on their number of parameters. Normally, a higher number of parameters implies greater accuracy but also higher computational load and lower speed. As an example, Table 3.2 compares the five segmentation YOLOv8 model sizes [111]. The concept of Mean Average Precision (mAP) is explained in Section 3.2.3.3.

Table 3.2. Characteristics of different YOLOv8 segmentation model sizes [111].

Model	Size (pixels)	mAPbox 50-95	mAPmask 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	Params (M)	FLOPs (B)
`YOLOv8n-seg`	640	36.7	30.5	96.1	1.21	3.4	12.6
`YOLOv8s-seg`	640	44.6	36.8	155.7	1.47	11.8	42.6
`YOLOv8m-seg`	640	49.9	40.8	317.0	2.18	27.3	110.2
`YOLOv8l-seg`	640	52.3	42.6	572.4	2.79	46.0	220.5
`YOLOv8x-seg`	640	53.4	43.4	712.1	4.02	71.8	344.1

3.2.3.2. Work Pipeline

The implementation of YOLOv8 is powered by several libraries and tools. **Ultralytics** provides the main framework for managing the YOLOv8 model, offering functionalities for training, inference, and performance evaluation [109]. **PyTorch** serves as the deep learning backend, handling the computational aspects of the model training and optimization processes [112]. Figure 3.12 shows the followed vision sub-system project pipeline.

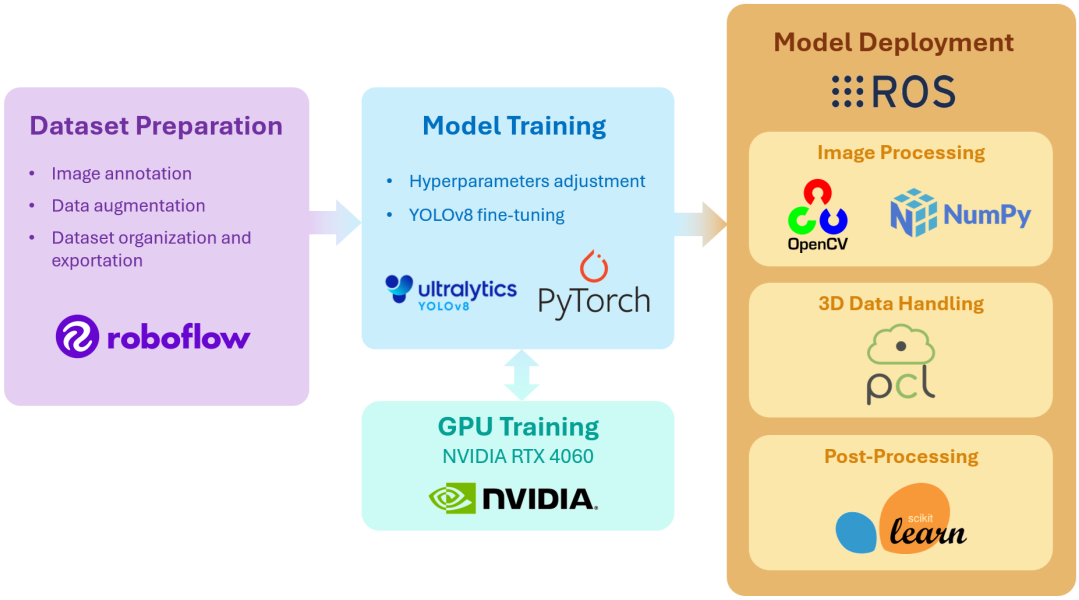


Figure 3.12. Vision model work pipeline schematic.

To prepare the dataset, **Roboflow** was utilized to manage and augment the data [113]. Roboflow offers an intuitive online platform for labeling images, performing data augmentation, and exporting datasets in formats compatible with various machine learning frameworks. Techniques like rotation, scaling, and color adjustments were applied to enhance the dataset's diversity, improving the model's ability to generalize to different scenarios. However, the actual training of the YOLOv8 model was carried out locally using the Ultralytics framework and Pytorch with a NVIDIA RTX 4060 GPU, ensuring a tailored and optimized training process.

3.2.3.3. Performance Metrics for Vision Models

To quantify how well a model performs in detecting and classifying objects, it is essential to understand the key metrics used to measure accuracy and reliability [110]. These metrics are derived from the count of true positives, TP, false positives, FP, and false negatives, FN. Some of the most relevant are:

- **Precision:** Precision measures the accuracy of the model's positive predictions. It is defined as the ratio of true positive detections to the total number of positive

predictions made by the model. In other words, it indicates the proportion of correctly identified instances among all instances classified as positive:

$$p = \frac{TP}{TP + FP} \tag{3.7}$$

- **Recall:** Also known as sensitivity or true positive rate, recall measures the model's ability to identify all relevant instances within a dataset. It is calculated as the ratio of true positive detections to the total number of actual positive instances. This metric reflects the proportion of actual positives that were correctly identified:

$$r = \frac{TP}{TP + FN} \tag{3.8}$$

- **mAP:** The mean average precision is a comprehensive metric that combines both precision and recall across different confidence thresholds. It evaluates the model's performance by integrating the precision-recall curve over varying level of detection confidence. The average precision (AP) is first calculated for each confidence threshold, and then these values are averaged:

$$\mathrm{AP}_n = \int_0^1 p_n(r)\, dr \tag{3.9a}$$

$$\mathrm{mAP} = \frac{1}{N} \sum_{n=0}^{N} \mathrm{AP}_n \tag{3.9b}$$

Where $p_n(r)$ represents the precision as a function of recall, r, at a specific confidence level, indexed by n. Typically, n varies from zero to one in increments of 0.1, resulting in $N = 11$ different confidence levels.

3.2.3.4. Additional Packages and Libraries

In addition to the primary tools, several other packages and libraries were utilized to enhance the functionality and efficiency of the vision algorithms and overall system.

- **OpenCV:** It is an open-source library originally developed by Intel for real-time computer vision applications. It supports multiple programming languages like C++ and Python and is platform-independent, making it versatile for various projects. OpenCV provides optimized algorithms for tasks such as image processing, feature detection, and video analysis. In this project, it is used primarily for processing images generated by the YOLOv8 model [114].
- **NumPy:** It is a core library for numerical computing in Python, offering support for large arrays and matrices, along with numerous mathematical functions. It is

essential for data manipulation and integration with other libraries like OpenCV. In this project, NumPy is used for handling and transforming image data during the preprocessing and post-processing stages [115].

- **Point Cloud Library (PCL):** It is an open-source library specialized in processing 3D point clouds, offering tools for filtering, segmentation, and surface reconstruction. Though its use in this project is limited, PCL is key for managing depth data and enhancing 3D perception capabilities in the robot's tasks [116].
- **Scikit-learn:** It is a machine learning library for Python, providing tools for data mining, classification, and dimensionality reduction. In this project, it is used for tasks like Principal Component Analysis (PCA) and RANSAC, which help refine the vision algorithms' outputs by reducing data dimensionality and ensuring robust model fitting [117].

3.2.4. Camera Calibration and 3D Data Analysis

This section details the algorithms and techniques employed for 3D vision data analysis, critical to the project's vision component. It first explains the key camera parameters obtained from the calibration process that are essential for accurate 3D point cloud reconstruction. Following this, RANSAC and PCA algorithms are covered, since they are pivotal in the vision algorithms developed in this work.

3.2.4.1. Camera Calibration and Parameters

Camera calibration is a foundational aspect of computer vision and robotics, involving the estimation of both intrinsic and extrinsic parameters that define the optical characteristics and positioning of a camera system. This process is essential for achieving high accuracy in applications that rely on visual data for navigation, 3D reconstruction, and interaction with the environment.

The **intrinsic parameters** of a camera describe its internal optical characteristics. In the case of the employed depth cameras, they come pre-calibrated directly from the manufacturer with factory-set intrinsic parameters tuned for general use out of the box. These parameters are encapsulated in the camera intrinsic matrix, often denoted as $\boldsymbol{K}$. The matrix includes the focal lengths, the principal point, and the skew coefficient, which characterizes the angle between the x and y pixel axes. The general form of the intrinsic matrix $\boldsymbol{K}$ is given by:

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.10}$$

where f_x and f_y are the focal lengths expressed in pixel units along the x and y axes, respectively; c_x and c_y are the coordinates of the principal point (which is ideally the image center); and s is the skew coefficient, typically zero in modern cameras where the pixel axes are perpendicular. These parameters are critical because they allow for the conversion from 2D image coordinates to 3D world coordinates; which is pivotal for tasks such as point cloud generation, where depth information must be accurately mapped to spatial dimensions. Figure 3.13 illustrates these intrinsic parameters [87].

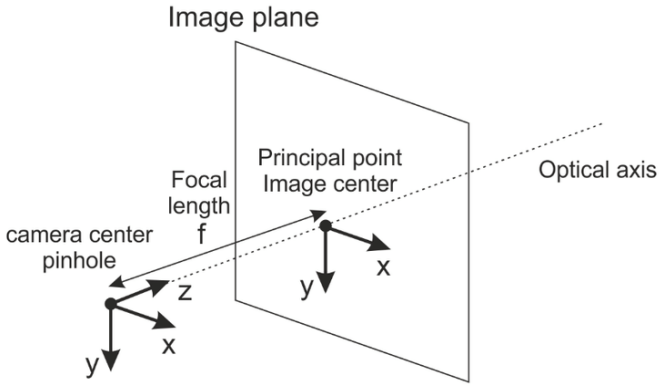


Figure 3.13. Camera intrinsic parameters [87].

With respect to the extrinsic parameters, they consist on a rotation matrix and a translation vector that describe the position and orientation of the camera in a world coordinate system.

3.2.4.2. Point Cloud Construction

A point cloud is a set of data points in space where each point represents a location. The point collectively commonly represent the external surfaces of visible objects in the environment. Generating point clouds from depth images involves mapping each pixel in the depth image into a coordinate in the 3D space based on its depth value and the camera's intrinsic parameters. Knowing the depth, z, for each pixel value, the pixel coordinates (u, v) can be converted into camera-centric coordinates using the following equations:

$$\begin{cases} x = (u - c_x) \cdot \dfrac{z}{f_x} & \text{(3.11a)} \\ y = (v - c_y) \cdot \dfrac{z}{f_y} & \text{(3.11b)} \end{cases}$$

3.2.4.3. RANSAC Algorithm

RANSAC is a robust statistical method used to estimate the parameters of a mathematical model from a dataset that contains outliers, as it can be seen in Figure 3.14a. This

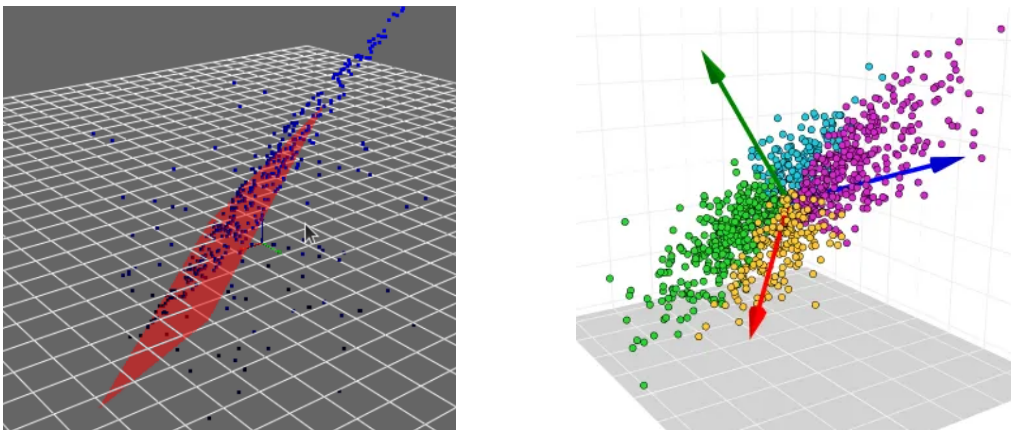
algorithm is particularly effective in the fields of computer vision and image processing where the data may be corrupted with noise or contain points that do not fit the desired model. Its pseudocode is presented in Algorithm 1.

Algorithm 1 RANSAC Algorithm

Require: Pointcloud $\mathcal{P}$, Number of iterations N, Distance threshold T
Ensure: Best model amongst all iterations $\hat{M}$

```
bestInliers ← 0
M̂ ← null
for i ← 1 to N do
    subset ← randomSubset(P)
    model ← fitModel(subset)
    inliers ← []
    for each point in P do
        if distance(point, model) < T then
            inliers.append(point)
    if size(inliers) > bestInliers then
        bestInliers ← size(inliers)
        M̂ ← reEstimateModel(inliers)
return M̂
```

It iteratively selects random subsets of data to fit a model, typically requiring just enough points to define the model, such as two points for a line. During each iteration, RANSAC identifies inliers—points that fit the model within a predefined tolerance. If a model has more inliers than any previously found model, it is considered a better fit and becomes the new best model. After a set number of iterations, the algorithm selects the model with the highest number of inliers as the most robust estimate.



(a) **(b)**

Figure 3.14. Data analysis algorithms: (a) 3D plane RANSAC fit [118], (b) PCA applied to a 3D dataset [119].

3.2.4.4. PCA Algorithm

PCA is a statistical procedure that transforms a set of possibly correlated variables into a set of linearly uncorrelated variables known as principal components. It helps in detecting patterns in data based on the correlation between features, facilitating efficient data compression and interpretative insights. Its pseudocode is presented in Algorithm 2.

Algorithm 2 Principal Component Analysis (PCA)

Require: Data matrix X with dimensions $m \times n$ (m samples, n features)
Ensure: Principal components matrix Y and the variances explained by each component Λ_k
1: Standardize columns of X to have zero mean and unit variance
2: Compute covariance matrix $\Sigma = \frac{1}{m-1} X^T X$
3: Perform eigenvalue decomposition $\Sigma = Q \Lambda Q^T$
4: Sort eigenvalues Λ and corresponding eigenvectors Q in descending order
5: Select k largest eigenvalues Λ_k and corresponding eigenvectors Q_k
6: Create projection matrix W from the selected eigenvectors Q_k
7: Transform the original data matrix X to obtain the principal components matrix $Y = XW$
8: **return** Principal components matrix Y and the variances explained by each component Λ_k

The algorithm starts by standardizing the data to have a mean of zero and standard deviation of one. PCA then computes the covariance matrix of the data, followed by its eigenvectors and eigenvalues. These eigenvectors, which define new axes, and eigenvalues, which measure the variance captured by each axis, are used to orient the data. The principal components are ordered so that the first few retain most of the variation present in the original dataset. By projecting the original data onto these few principal components, PCA achieves dimensionality reduction while preserving as much variability as possible. An example of PCA application to a 3D dataset in shown in Figure 3.14b.

Chapter 4

Development

This chapter presents the comprehensive development process undertaken in this project, covering the core components that underpin the research objectives. It begins with an exploration of the designed system architecture, followed by detailed sections on simulation modeling, the implementation of the NMPC controller from the base repository in both simulated and real-world settings, and the development of sophisticated vision algorithms for door and handle detection. Additionally, the chapter discusses the planning algorithms employed and concludes with insights into the system integration process.

4.1. System Architecture

The system architecture was designed to make the quadruped manipulator robotic system be able of performing obstacle traversal and door opening taking advantage of the base NMPC controller repository. The system's high level architecture can be seen in Figure 4.1, where **three modules** are shown: the vision module, the planner module, and the control module.

The **vision module**, `qm_vision`, is not used in this work for the obstacle traversal tasks, for which the robot relies only on the controller's state estimation without visual feedback (excluding the visual odometry). In this case, the vision module has been designed to generate visual references for performing the **door opening** tasks. These references enable the robot to position and orient itself to optimally open the door and increase the success rate. Additionally, it provides the robot with 3D information of the handle location to perform the loco-manipulation task. Future work could include incor-

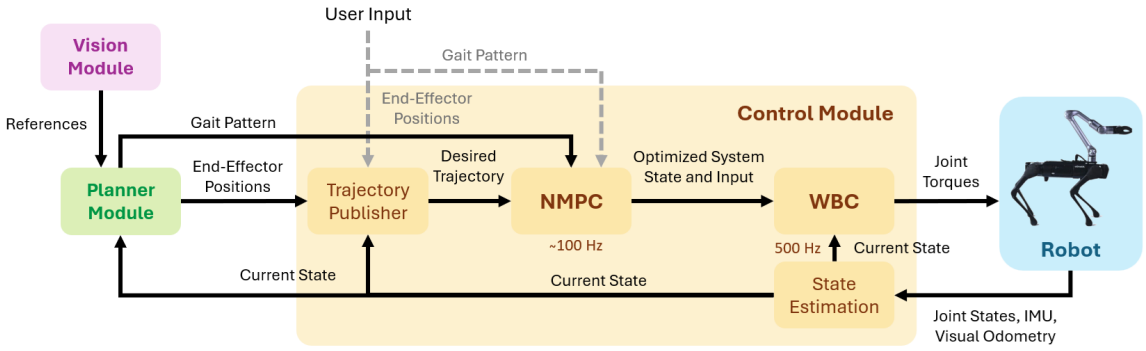


Figure 4.1. System architecture.

porating visual information extracted from the vision module into the NMPC controller as it was done in [67], possibly improving significantly the performance in very complex and difficult SAR scenarios. This module was developed in Python following **Object-Oriented Programming (OOP)** practices thanks to the fast iterative development possibilities this programming language offers, specially in computer vision. Additionally, it was developed as a ROS package, enabling easy integration with the rest of the modules.

With respect to the **planner module**, `qm_planner`, it was also developed from scratch as a ROS package but in this case it was written in C++ for improved performance. This module is basically a substitution of the user input interface the original repository provided to enable the robotic system to perform tasks more autonomously, without constant user input. This is done by getting position references from the vision module and the current state from the Kalman filter state estimator from the GitHub repository. With this information the planner is able to determine at which state of the performed task (e.g., door opening) the system is, and generate appropriate gait schedule and end-effector position commands to feed the NMPC controller. These instructions were originally manually sent through the terminal and the RViz interactive marker.

It is primarily used for the door opening task, though it was also employed for higher repeatability along multiple experiment iterations in simulated unstructured environments. Future work would include integrating it within the planning and navigation frameworks of previous work conducted by the lab for SAR scenarios were doors were significant impediments [1].

Finally, the control module is implemented in C++ in several ROS packages. The packages' structure from the original repository was not modified and it groups different types of files according to their functionality in the control program such as the description files (`qm_description`), the state estimation (`qm_estimation`), the gazebo simulation (`qm_gazebo`), the formulation and resolution of the NMPC optimization problem (`qm_interface`) or the WBC controller (`qm_wbc`). The main control loop that manages

the whole control architecture and the user interface is programmed in `qm_controllers`. Two additional packages are used for common code used across different packages, and for including a wrapper[1] for defining the QP problems: `qm_common` and `qpoases_catkin`, respectively. Though the base repository presented a good foundation for developing the controller, it has several limitations that have been addressed in this work. The controller was adapted for the Unitree Aliengo and Z1 robots in specific and the original capabilities of the quadruped manipulator were augmented as described in Section 4.3. The detailed integration of the whole architecture is presented in the last section of this chapter.

4.2. Simulation Modeling

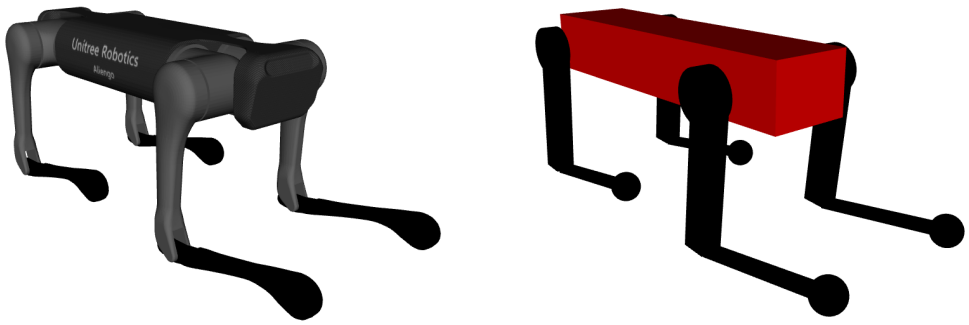
To develop this software, it was chosen to use **Gazebo simulation platform** to mimic the physical environment where the robot would be performing. The simulation modeling process is a crucial step in testing and validating the robotic system's functionality before real-world implementation. Gazebo was chosen over other alternatives such as Unity due to the fact it had already been implemented in the original repository, and because it is within the ROS ecosystem, which eases the development process.

To facilitate simulations within Gazebo, it is essential to first construct the **URDF files**. These files provide a detailed description of the robot's physical attributes, such as its geometry, kinematics, dynamics, and visual elements. Once the URDF files are prepared, additional configuration files and simulation environments are set up to create a comprehensive model that interacts within Gazebo.

4.2.1. Robot modeling

The first step was to modify the robot URDF description files from the repository since these were the ones corresponding to the AlienGo with the Kinova Jaco v2 arm. The URDF files are created from **`.xacro` files** that enable to create shorter and more readable XML files than the whole URDF. For example, in the case of the AlienGo, there is one file with the model mass and inertia parameters, and others for the materials, leg sub-assembly, transmissions, IMU and Gazebo configuration. This enables to create a unique URDF description file of the quadruped manipulator from the description files of the quadruped and the manipulator. It is also in these files where the collision bodies geometries are defined. Figure 4.2 shows the model of the quadruped and its simplified collision bodies model.

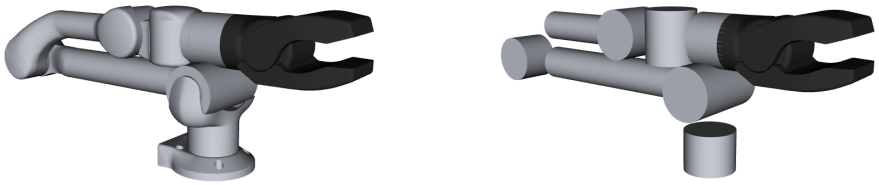
[1] Layer of code that facilitates interaction with a software library, simplifying its use or adapting it for compatibility with different programming environments.



(a) **(b)**

Figure 4.2. AlienGo quadruped URDF model: (a) Geometric model, (b) Collisions model.

In order to implement the manipulator, the geometry mesh files corresponding to each link of the Z1 robot were retrieved from the official `unitree_ros` GitHub repository from Unitree Robotics [120]. The Z1 arm geometric and collision bodies models are shown in Figure 4.3. An additional invisible spherical link in the grasping position was created as required by the NMPC controller.



(a) **(b)**

Figure 4.3. Z1 manipulator URDF model: (a) Geometric model, (b) Collisions model.

Finally, to setup the robot model it is necessary to add the Intel RealSense SR305. In order to do this, a **support was designed in Autodesk Inventor** to fix the camera to the Z1 arm. It was decided to add it in the last arm joint to have a close look when manipulating the door handle. It is composed of three pieces, a lower half-ring, an upper half-ring, and the camera fixation support. The lower half-ring is fixed to the upper-ring through two bolts and the design enables to modify both the height and the orientation angle of the camera through a third bolt that links the upper ring and the camera fixation support, as shown in Figure 4.4. In addition to this, a protection cover was designed for the camera.

These designed components were added to the URDF of the quadruped manipulator as links of the Z1 manipulator. To finish with the robot URDF configuration, the camera models were created. Two depth camera Gazebo sensors were included in the URDF by configuring the camera parameters to be the ones of Intel's datasheets [87][121], such as the horizontal field of view, the resolution, and the minimum and maximum range.

(a) (b)

Figure 4.4. Camera support model: (a) Support model, (b) Protection cover model.

By using the Gazebo plugin `librealsense_gazebo_plugin.so`, the SR305 and D435 cameras were simulated to publish the ROS messages in the topics the real devices would be using. In particular, the cameras were set to publish the color image, the depth image, the color and depth camera information messages containing camera metadata essential for various vision related tasks, and the 3D point cloud of the whole image (published by the real camera thanks to a program in the Intel RealSense SDK [122]). Two virtual links, camera link and camera optical link, are created for the Gazebo camera plugin to work properly. Once this is done, the final robot URDF is generated from the associated `.xacro` files. Figure 4.5 shows the robot link tree-like architecture and Figure 4.6 shows the nomenclature used for each joint of the quadruped manipulator model.

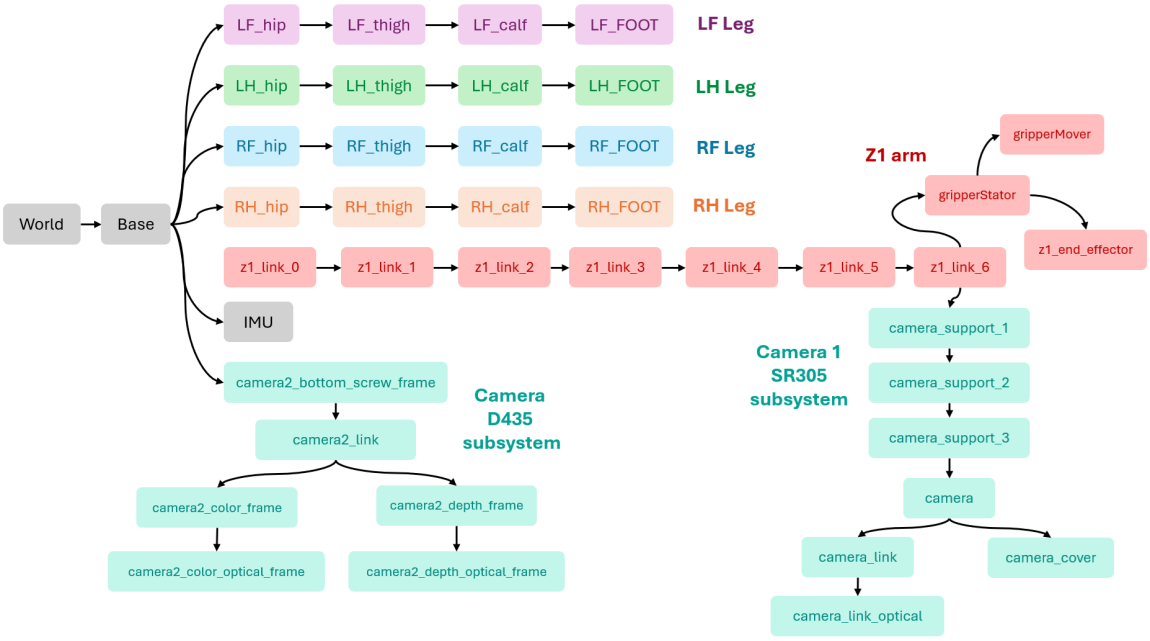


Figure 4.5. Robot link tree-like architecture (ROS TF tree).

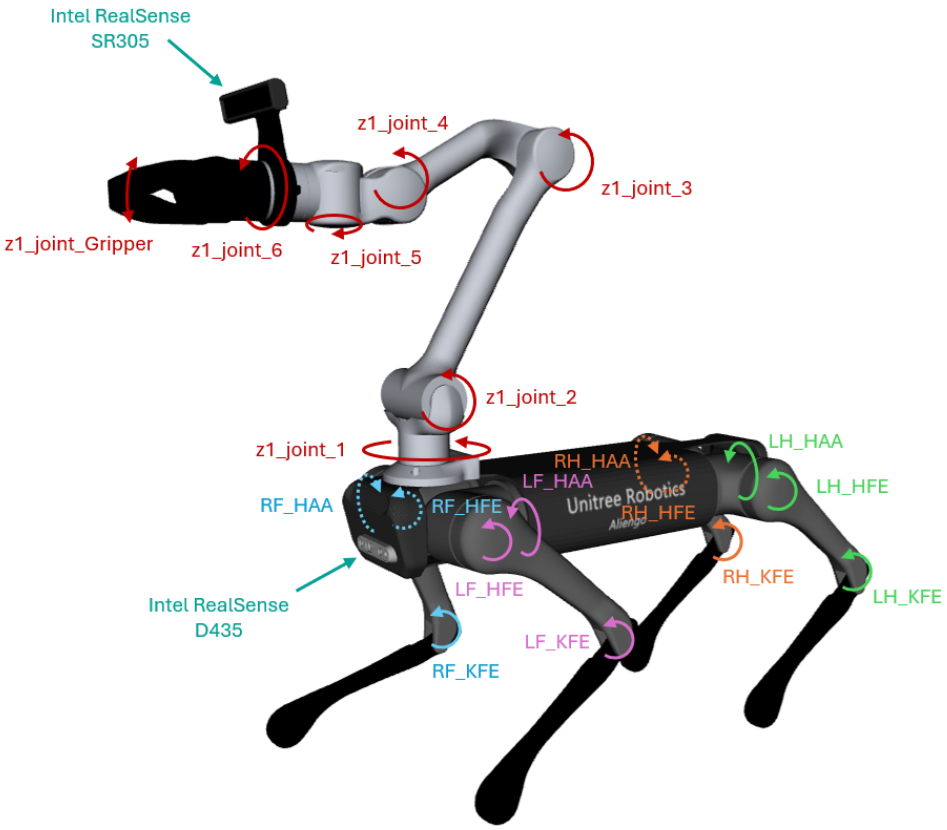


Figure 4.6. Robot assembly and joint configuration.

4.2.2. Environments Modeling

This section focuses on the modeling of the environments utilized in the simulation of robotic tasks, specifically addressing the challenges posed by unstructured SAR scenarios and door opening operations. For each environment, detailed models were constructed to replicate real-world conditions as closely as possible to evaluate the system performance in simulation.

4.2.2.1. Unstructured SAR Environments

To effectively test the features and robustness of the robotic controller under various challenging conditions, five distinct types of unstructured environments that could be found in SAR missions have been developed:

1. **Variable height pallets:** This environment consists of a series of pallets with varying heights, including one of 20 cm height that exceeds the maximum height of 18 cm specified by the robot's manufacturer for its standard controller [46]. The characteristics of the pallets and their distribution can be seen in Figure 4.7.
2. **Tunnel and irregular terrain:** This model features a tunnel of 60 cm height with irregular terrain and restricted height, testing the robot's ability to operate in

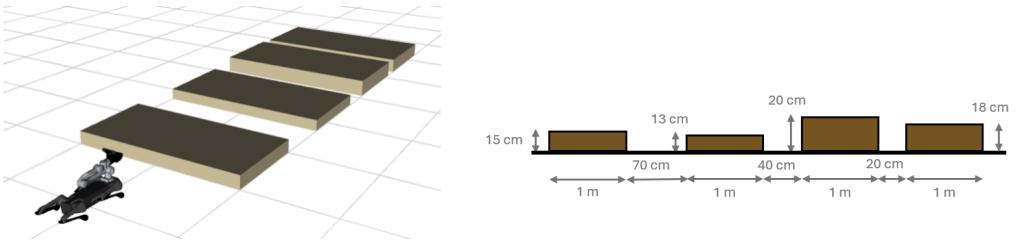


Figure 4.7. Variable height pallets environment.

confined spaces with non-uniform ground conditions. Figure 4.8 shows the designed environment. This irregular terrain was built using the surface modeling tool from Autodesk Inventor.

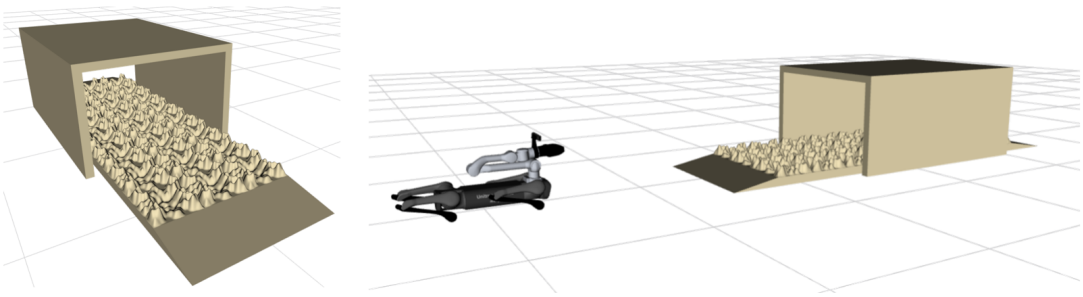


Figure 4.8. Tunnel and irregular terrain environment.

3. **Unstable platform:** An environment featuring an unstable platform supported by two 10 cm spheres can be seen in Figure 4.9a. This scenario assesses the robot's balance control and adaptability to maintain functionality on dynamic, uneven surfaces. The absence of this type of reactive control behavior is one of the main disadvantages of simpler control methods like CPG installed by default in Unitree quadrupeds.
4. **Stair Navigation:** It comprises an obstacle course that includes ascending stairs, a ramp, and descending stairs as it can be seen in Figure 4.9b. Figure 4.9c shows the dimensions of the designed environment. This environment evaluates the robot's mobility and stability control across different inclinations and step configurations.
5. **Maze navigation:** Four types of mazes were designed modularly as part of a different Master's Thesis tackling the control problem with RL. Each block constituting the maze is 0.75×0.75 m. The labyrinths were built physically but the NMPC could not be tested in this environment for time constraints.

Each environment is designed to progressively be able to test the capabilities of the robot controller, trying to mimic unpredictability typical of SAR operations, but certainly limited in comparison with the Red Zone classification according to the NIST. These environments would correspond to an **orange zone** according to this classification.

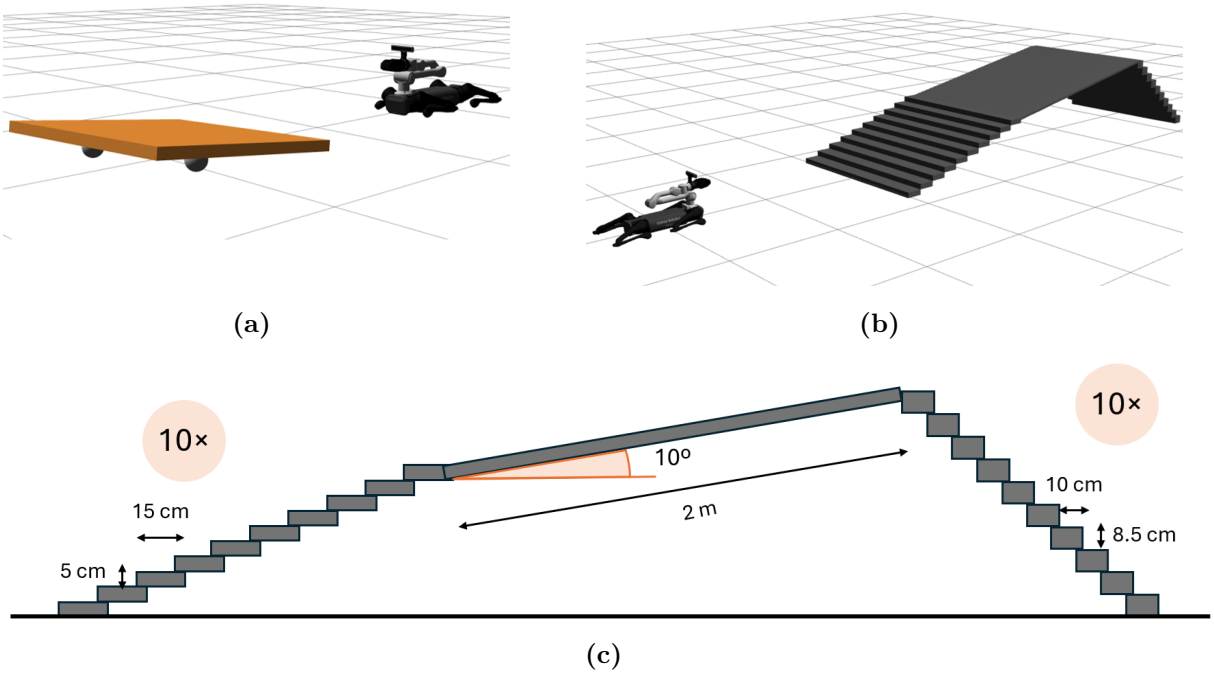


Figure 4.9. Modeled environments: (a) Unstable platform, (b) Stairs and ramp, (c) Stairs environment dimensions.

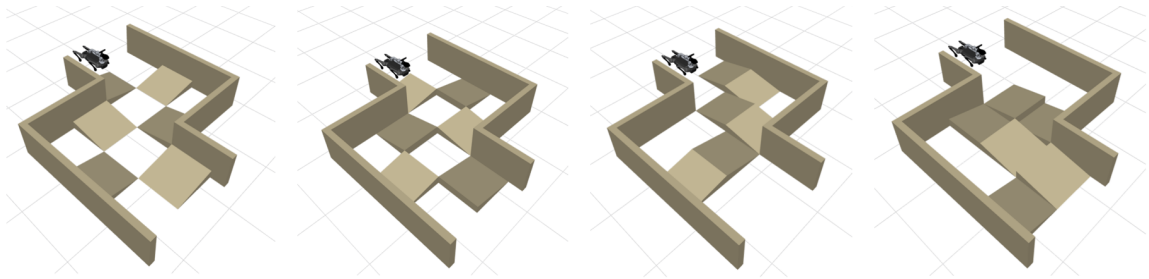
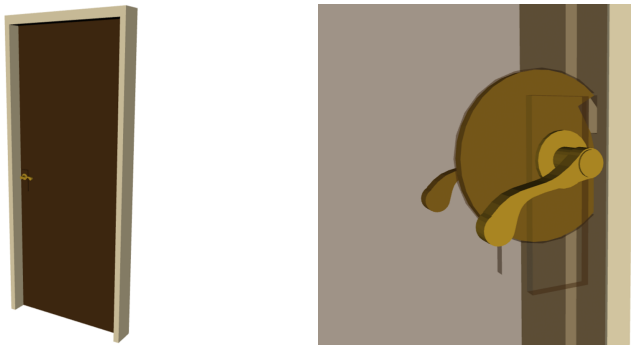


Figure 4.10. Maze environment configurations.

4.2.2.2. Door Opening Environments

In order to create the door opening simulation environment, a URDF model was created based on [123]. The model is constituted by three parts: the door frame, the door, and the handle; and has 2 DOF since both the door and the handle can rotate. Figure 4.11 shows the model and a detail view of the handle mechanism. The door has initially been configured to be a non-spring door. The handle joint can be modeled as an elastic joint in Gazebo by defining the spring stiffness and damping parameters, though initially these parameters were left to zero to start with the most simple behavior.

Two door opening environments have been created locating the door in different positions for **push and pull opening** simulations. These environments are launched from Gazebo launch files which enable to launch several object spawning nodes all at once



(a) (b)

Figure 4.11. Door model: (a) Overview, (b) Door handle detail.

(e.g., quadruped manipulator, door, walls, etc.). The door is interpreted as a 2-DOF non-actuated robot continuously publishing its states in order to be able to visualize it in RViz. Two extra static walls were added to both sides of the door.

4.3. NMPC Controller

The NMPC controller implemented in this project was initially based on a GitHub repository that provides a good foundation work for quadruped manipulators application [95]. While the repository offered a solid starting point, several limitations were identified that necessitated significant modifications to adapt the controller to the specific application designed in this work and improve its performance.

4.3.1. Limitations of the Base Repository

The base repository provides a NMPC framework of the AlienGo robot with the Kinova Jaco v2 arm. It includes basic functionalities for gait generation, state estimation, and solving the optimization problem. The repository includes specific branches for ensuring whole-body **compliance under actuation saturation**, considering **force disturbance** on the manipulator's end-effector and **real robot hardware** implementation. These last two branches were of specific interest for the door opening application and real-world deployment. The repository additionally offers the possibility to control the quadruped manipulator both as a **combined system** and as a **separate system**. However, there were several identified limitations:

- **Kinova Jaco arm support:** The base repository was originally designed to support the Kinova Jaco manipulator arm. However, this project required the integration of the Unitree Z1 arm, which has different kinematic and dynamic characteristics. This limitation meant that the original description and control parameters files were not directly applicable.

- **Inability to handle vertical obstacles, stairs or ramps:** The original repository was not configured to handle vertical references effectively, making the robot incapable of climbing stairs or overcoming significant obstacles. This was because the base height restriction was imposed with respect to the world frame and when the robot faced a vertical obstacle it tried to compensate it by lowering the body till the point of crashing with the ground.

- **Complex initial position configuration:** To change the initial position of the robot in the simulation, six different files needed to be modified manually with several correlated parameters. This cumbersome process introduced potential for errors and made it difficult to perform rapid testing and iterations during development.

- **Manual gait pattern and end-effector setting:** The gait pattern for the robot, crucial for adapting to different terrains and tasks, must be manually set by the user via the terminal. This lack of automation requires constant user intervention, which can be inefficient. The same happens with the end-effector position, critical for tasks involving manipulation, which must be manually set through RViz by the user. This manual adjustment limits the system's autonomy and makes it difficult to perform tasks that require dynamic or real-time end-effector adjustments.

- **Gripper controller limitations:** The finger gripper controller included in the repository could not be operated simultaneously with the NMPC controller unless the system was in separated systems mode. The restriction hindered the robot's ability to perform continuous manipulation tasks while maintaining locomotion, which is essential for seamless operation in dynamic environments.

4.3.2. Modifications and Enhancements

To address these limitations, several modification and enhancements were implemented to improve the controller towards SAR and door opening applications:

- **Integration of the Z1 manipulator:** The NMPC controller was modified to support the Unitree Z1 arm, replacing the original Kinova Jaco configuration. This involved updating the robot's URDF files to reflect the Z1's geometry, kinematics, and dynamics, and modifying the NMPC to control the new arm joints effectively. Small modifications were made in several files to account for the new joint naming, and Proportional-Integral-Derivative (PID) controllers. Some NMPC controller's parameters were modified to improve the system performance in certain tasks as it will be later described.

- **Base height estimation redefinition:** To enable the robot to handle vertical obstacles, stairs and ramps, the NMPC controller was modified to calculate the base height relative to the position of the feet in contact with the ground instead of the world frame. This was done by using Pinnochio's library [100] to determine the position of the feet end effectors. With their position and the boolean determined by each foot's force contact sensor, it was possible to calculate the average height of the feet in contact with the ground.

- **Simplified initial position configuration:** The process of setting the robot's initial position was streamlined by consolidating the configuration parameters into a single file in the `qm_controllers` package. This change reduced the risk of errors and made it easier to configure and deploy the robot in different scenarios.

- **Automated gait pattern selection and end-effector positioning:** A subscriber to the gait pattern topic was implemented in the NMPC controller to avoid the need for defining the gait schedule from the terminal. Something similar was done with the end-effector positioning, making it possible to define its position and orientation directly through a ROS topic without the need to touch the interactive RViz marker, though this can still be done.

- **Enhanced gripper control integration:** Some progress was done towards integrating the gripper controller seamlessly with the NMPC controller, allowing for simultaneous operation in combined systems. However, resource conflicts were found in the ROS controller manager and this improvement is still under development. In the meantime to enable door opening testing, a **custom piece was designed** to be attached to the closed gripper to enable **safe manipulation**.

With respect to the NMPC controller **parameters**, the ones used for the Kinova arm showed to **work relatively well** for the Z1 arm, so they were not initially modified to focus on the application development. However, after the implementation of the force compliance feature, which is described in the next section, parameter tuning was required since high instability occurred in some of the manipulator joints. The tuning process for these parameters is described in Section 4.3.4.

The **SRBD dynamics model** was chosen due to is lower computational cost compared to the FCD model, and reasonably good performance. Other model parameters which were modified for the different simulation tests were the surface static friction coefficient, the base height reference, linear and angular target velocities or the leg swing height, which can be set in `task.info` and `reference.info` configuration files from the `qm_controllers` package.

4.3.3. Force Compliance for Door Opening

The initially used controller—used in the unstructured environment locomotion tests—**did not include a force compliance constraint** in the NMPC problem. This limitation led to **instability** when interacting with external objects, such as doors, because the controller was unable to regulate the applied forces to maintain a stable interaction. Specifically, when the robot's end-effector came into contact with the door, it could not adjust the applied force to ensure a smooth and compliant motion. This resulted in crashes or failure to maintain the desired trajectory.

To address this issue, force compliance was incorporated into the NMPC optimization problem by **adding a dedicated constraint** for the end-effector force. This constraint ensures that the force applied by the end-effector is controlled according to the desired values from the controller. The equality constraint is defined as:

$$-\boldsymbol{f}_{\text{ee}}(t) + \boldsymbol{u}_{\text{input}} = 0 \tag{4.1}$$

Where $\boldsymbol{f}_{\text{ee}}(t)$ represents the force exerted by the end-effector, and $\boldsymbol{u}_{\text{input}}$ corresponds to the force commands generated by the NMPC system. By integrating this force constraint into the optimization process, the controller ensures that the robot can respond to external disturbances, such as interacting with a door, by adjusting the forces applied to maintain stability and complete the task effectively.

Additionally, a **low-priority task** for force tracking was incorporated into the WBC controller, as shown in Table 3.1. This allowed the system to monitor and track the forces applied to the end-effector, ensuring the robot could react compliantly during contact, like when pushing or pulling a door.

4.3.4. Model Parameter Tuning

After the bad performance of the quadruped manipulator when the force constraint was implemented, it was decided to **tune the control parameters**. These included the weight matrices for the NMPC and the PD gains for the WBC. This tuning aimed to ensure that the system could handle tasks like door opening, force compliance, and general locomotion with precision, stability, and energy efficiency.

4.3.4.1. NMPC: $\boldsymbol{Q}$ and $\boldsymbol{R}$ Matrices

$\boldsymbol{Q}$ matrix penalizes deviations from the desired state, while $\boldsymbol{R}$ matrix penalizes the magnitude of control inputs. These matrices were tuned to strike a balance between performance and control effort:

- ***Q* matrix:** Higher values were assigned to state variables that required greater accuracy, such as the position of the base and end-effector. This ensured that the system prioritized these states. For instance, the weights for the leg joint positions were kept relatively low since slight deviations in joint angles were acceptable. On the contrary a high weight was given to the base height parameter. The values defining the ***Q*** matrix can be seen in Table B.1 in Appendix B.

- ***R* matrix:** The ***R*** matrix was designed to limit the control effort, ensuring that the system used energy efficiently without overloading the actuators. For instance, the feet contact forces were weighted much less that the arm joint velocities. This balance ensured smooth control while preventing aggressive inputs that could destabilize the system. The values defining the ***R*** matrix can be seen in Table B.2 in Appendix B.

4.3.4.2. WBC: K_p and K_d parameters

The proportional and derivative gains in the WBC determine the responsiveness and stability of the system. They are detailed in Table B.3 in Appendix B.

- K_p **and** K_d **for base control:** The values for linear and angular base control were set to 400 and 140, respectively. These gains ensured that the robot could stabilize its base while maintaining smooth movements, especially during complex maneuvers like door opening. Higher K_p values lead to quicker corrections, while the moderate K_d values dampened oscillations, preventing overshooting.

- K_p **and** K_d **for swing leg control:** The gains for the swing legs were set to 350 and 37, respectively. These values were tuned to ensure the legs could follow the desired trajectories accurately during locomotion while maintaining stability.

- K_p **and** K_d **for arm joints:** The arm joint parameters were higher, with proportional gain values between 4000 and 6000 and derivative values around 75. This reflected the need for high precision and force in manipulation tasks like door handling, while the lower K_d values allowed for smoother movements without excessive force.

The tuning process was **iterative**, with initial values based on theoretical concepts, followed by empirical adjustments through simulation. Probably real-world testing would lead to a different set of controller parameters since the model does not fully coincide with the real robot. Additionally, due to the iterative nature of the process and the significant amount of time required for adjusting all the parameters properly, further optimization is highly recommended for enhancing performance.

4.4. Vision Algorithms

The ability of the quadruped manipulator to interact effectively with its environment hinges significantly on robust vision systems. This section delves into the specialized vision algorithms to be able to recognize and manipulate doors, based on four ROS nodes. Note however that these **algorithms are limited** in the current stage of development **to controlled settings** and there is still a lot of work to do to implement them in an active navigation algorithm for the robot to be able to interact autonomously with doors in SAR tasks.

To give a general perspective of the vision subsystem development, first of all, two YOLOv8 models, one for door detection and another one for handle classification and segmentation, were trained. The first node uses the door detection model to detect the door and calculates its center with respect to the camera to estimate references depending on the orientation and position of the quadruped with respect to the door. This node additionally uses the D435 camera to estimate the normal distance and vector to the door wall plane.

The handle model is used in a handle classification and segmentation node that identifies and isolates the door handle from visual data. Following this, in the same node depth data is used to generate a point cloud of the segmented handle. A third node subscribes to this point cloud topic and calculates the optimal grasping point and orientation. Finally, one last node estimates the door radius from the information published from the rest, which is one of the main manipulation-relevant door parameters.

4.4.1. Dataset Preparation and Model Training

At the outset of this project, it was decided to train two distinct YOLO models to cater to the specific requirements of the robotic door interaction system. By splitting the tasks between two models, each can be individually optimized for its specific purpose without overburdening the system with unnecessary information processing. The datasets for both models were prepared in Roboflow and both models were GPU-trained locally as it was shown in Figure 3.12.

- **Door Object Detection Model:** The primary function of this model is to serve as a visual feedback mechanism for planning the robot's approach towards a door.
- **Door Handle Segmentation Model:** Once the robot has positioned in a proper starting location, the next step involves the precise interaction with the door handle to open it. This task requires a higher level of detail to accurately segment and iden-

tify the specific features of different handles, including their shapes, positions, and orientations. A dedicated model for handle segmentation allows for a more focused and detailed analysis, which is necessary for precise grasping and manipulation.

4.4.1.1. Door Object Detection Model

The door object detection dataset [124] consists of 1561 uniform resolution 640×640 resolution pixel images, which is optimal for processing while maintaining enough detail for accurate feature recognition. The images were box-annotated to identify the door class. To ensure the effectiveness of the model and mitigate issues like model overfitting and bias, the dataset was divided into **three subsets**: 80% for the training set, and 10% each for validation and testing. This dataset includes several images which have been data-augmented through modifying its lighting conditions or vertical flipping, which further enhances the model robustness.

Among the different available YOLOv8 detection model sizes, `yolov8n`, the nano model, was chosen. This decision was taken because it is considered to be the fastest alternative, which is critical for real-time applications. The smaller size of `yolov8n`, compared to its larger counterparts like `yolov8s` or `yolov8m`, offers faster processing times, making it suitable for deployment on hardware with limited computational resources without a substantial sacrifice in detection performance for the application. The training model was conducted over 100 epochs.

The evolution of the performance metrics during training can be seen in Figure 4.12. Some examples of the model validation results are shown in Figure 4.13.

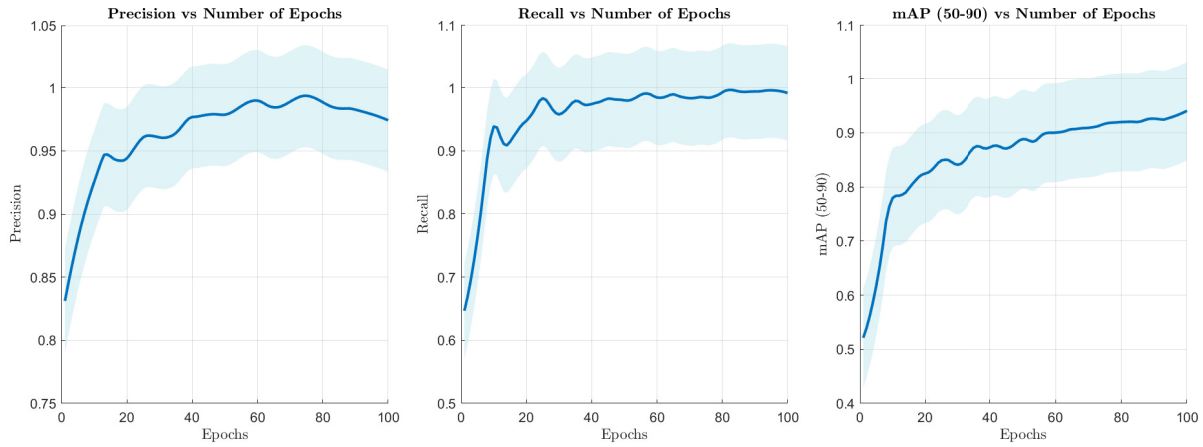


Figure 4.12. Door detection model performance metrics.

Though originally the dataset included only real door pictures, it was decided to **add the frames of a video recorded in simulation** from the D435 camera (around 15%

of the whole dataset divided in training, validation and testing). This was because the initially trained model showed to excel at detecting real doors but missed the simulated ones, which impeded code development in simulation. After this change, the model showed to perform well both in real and simulation environments.

Figure 4.13. Door detection model validation results.

From the performance parameters it can be concluded the model **performs exceptionally well** on both precision and recall, implying it can detect the doors with high accuracy. The mAP values are strong, and the similar magnitude of training and validation losses suggests that the model is not overfitting, which is corroborated by the high precision and recall values.

4.4.1.2. Door Handle Segmentation Model

The door handle segmentation dataset [125] consists of 2153 uniform resolution 640×640 resolution pixel images. The images were annotated to identify and segment **two distinct classes: handles and knobs**. The purpose of including these two classes is to trigger different manipulation strategies depending on the recognized class in future work (out of the scope of this project since only handles were treated). In a similar way to the previous case, the dataset is divided into **three subsets**: 80% for the training set, and 10% each for validation and testing.

To further enhance model robustness and ensure effective generalization across various lighting conditions, angles, and object variations, a portion of the 2153 images in the dataset has been generated using **data augmentation** techniques. This included horizontal and vertical flipping, rotations and shearing. In this case, since the segmentation task is slower than detection, it was decided to also use the nano model, `yolo8vn-seg`, specially due to the speed needed for this application.

The training model was conducted similarly over 100 epochs. The evolution of the performance metrics during training can be seen in Figure 4.14. The normalized confusion matrix of the model after a 100 epochs, together with some model validation examples are shown in Figure 4.15.

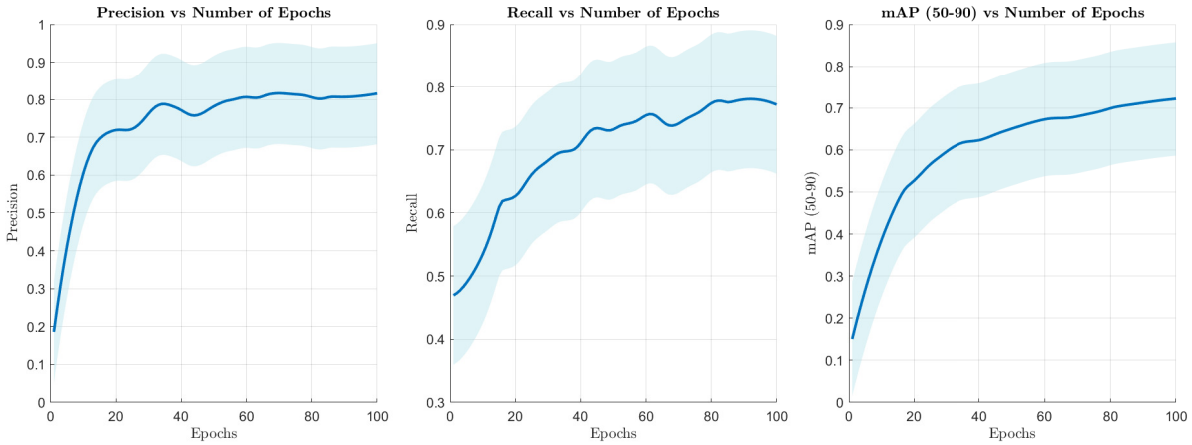
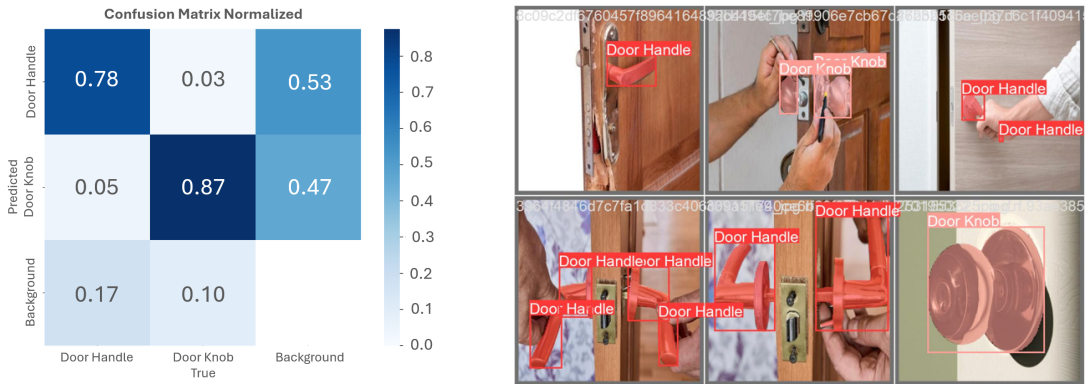


Figure 4.14. Handle segmentation model performance metrics.

From the performance metrics, high precision is observed, indicating that when a class is predicted, it is likely correct. The recall is also relatively high, indicating that the model is good at identifying all the relevant samples within the classes. With respect to the mAP, it suggests robustness in detecting and segmenting the objects across varied overlap thresholds. Finally, with respect to the normalized confusion matrix shown in Figure 4.15a, it is observed the model performs well enough for both door knobs and handles. Though this model could be further improved, in the validation results it is observed in Figure 4.15b to work quite well. Additionally, it has been tested in simulation and real environments and it performs relatively well for the required applications, so it was decided to use these model's weights.



(a) **(b)**

Figure 4.15. Handle segmentation model: (a) Confusion matrix, (b) Validation results.

4.4.2. Door Detection and Reference Generation

The first ROS node, `door_center` is in charge of identifying and localizing the door center vertical line using the D435 camera RGB image. The primary purpose of this node

is to assist the robot in accurately identifying the position and orientation of a door within its environment to be able to reposition for optimal opening. The model currently only detects doors but it could be trained in the future to additionally detect hinges to know if the door is of pull or push type.

This node additionally handles the depth information from the camera to calculate the reference variations in the yaw angle, ψ, and x and y coordinates relative to the door, as it can be seen in Figure 4.16.

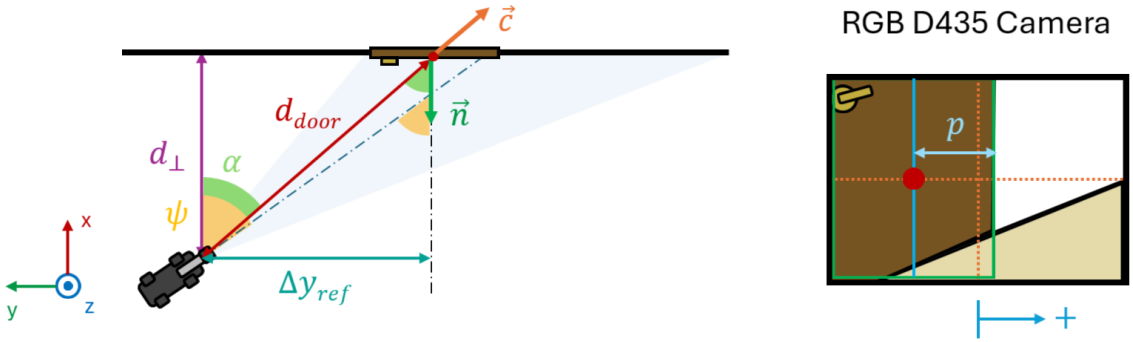


Figure 4.16. Door reorientation reference generation.

The calculation is programmed to be a ROS service so these operations are only performed when requested by a client. The operation consists of calculating the depth value corresponding to the pixel in the center of the detected door box, d_{door}; and the estimation of the normal distance to the door wall plane, $d_\perp$. p is the pixel distance (positive or negative) between the image center and the detected door center. The yaw variation is calculated from the scalar product of the camera direction vector, $\hat{\boldsymbol{c}}$, and the estimation of the vector normal to the door wall plane, $\hat{\boldsymbol{n}}$. The camera direction vector is constant in the camera frame where the calculations are made, with value $\hat{\boldsymbol{c}} = [0, 0, 1]^T$. The x reference command can be directly calculated from d_{door} by knowing the safety distance it is desired to maintain with the door in the normal direction before trying to open it, $d_{\perp_0}$. Knowing this, these equations enable to calculate the reference values.

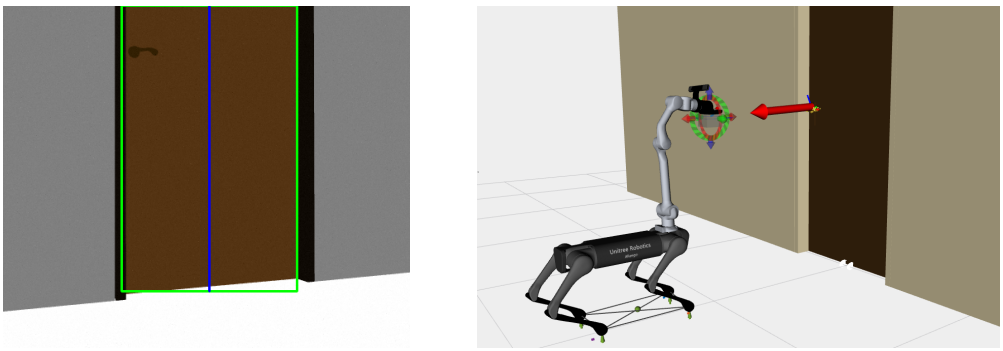
$$\begin{cases} \psi = \text{sign}(\hat{n}_x) \cdot [\pi - \arccos(\hat{\boldsymbol{c}} \cdot \hat{\boldsymbol{n}})], \quad \psi \in (-90°, 90°) & \text{(4.2a)} \\ \Delta x_{\text{ref}} = d_\perp - d_{\perp_0} & \text{(4.2b)} \\ \Delta y_{\text{ref}} = \text{sign}(p \cdot (\psi - \alpha)) \cdot \sqrt{d_{\text{door}}^2 - d_\perp^2} & \text{(4.2c)} \end{cases}$$

Once the door center is calculated from the corners of the detection box as shown in Figure 4.17a, its depth is calculated from the depth reading associated to the RGB pixel where the vertical line crosses the horizontal axis. In order to do this, the pixel coordinates from the RGB camera are transformed into the corresponding depth camera coordinates using both cameras' intrinsic and extrinsic parameters.

The normal vector, $\hat{\boldsymbol{n}}$, and distance, $d_\perp$, are calculated by selecting a plane-type model and applying **RANSAC** method, fixing a distance threshold of 0.01. Before this, the data is process with the PCL library [116] to remove any corrupted values. If no RANSAC inliers are found, the program logs that no planar model could be estimated. If a plane $ax + by + cz + d = 0$ is found; the node extracts the normal vector and the perpendicular distance to the plane. Note that a, b, and c are the components of the normal vector to the plane. d in the plane equation is the perpendicular distance from the origin to the plane scaled by the magnitude of the normal vector. According to this, the normalized **normal vector**, $\hat{\boldsymbol{n}}$, and the real **distance to the plane**, $d_\perp$ are obtained as:

$$\begin{cases} \hat{\boldsymbol{n}} = \dfrac{\{a, b, c\}^T}{\sqrt{a^2 + b^2 + c^2}} & \text{(4.3a)} \\ d_\perp = \dfrac{d}{\sqrt{a^2 + b^2 + c^2}} & \text{(4.3b)} \end{cases}$$

Additional code was written to represent the normal vector as a marker in Rviz, which can be seen in Figure 4.17b. In order to represent this normal vector the node subscribes to the position of the handle centroid, which is published by a different node since for calculating it the SR305 camera mounted on the Z1 arm is used. A pseudocode of this node is shown in Algorithm 3.



(a) **(b)**

Figure 4.17. Door detection: (a) Door center, (b) Normal vector.

4.4.3. Handle Detection and Point Cloud Segmentation

The weights corresponding to the `yolo_v8n-seg` model are used in the second ROS node, `handle_pc`, for detecting and segmenting door handles. This node additionally processes the image and applies the resulting segmentation binary mask to the depth image. This way, a segmented point cloud corresponding to the handle is obtained.

In first place, the node subscribes to the RGB and depth images from the depth camera and runs **inference** on the trained YOLO model with the RGB image. To improve the

Algorithm 3 Door Detection and Reference Calculation

Require: YOLO weights, D435 camera parameters, point cloud $\mathcal{P}$, handle centroid C
Ensure: Door center references (Δx_{ref}, Δy_{ref}, $\Delta \psi_{\text{ref}}$)
1: Initialize ROS node YOLO model with pre-trained weights
2: Acquire RGB image, depth image, and $\mathcal{P}$
3: $d_{\perp}$, $\hat{\boldsymbol{n}} \leftarrow$ Compute plane from $\mathcal{P}$ using RANSAC
4: *door_box* $\leftarrow$ Detect door in RGB image using YOLO
5: *door_center* $\leftarrow$ Calculate center of detected door in RGB image coordinates
6: Transform *door_center* to depth image coordinates
7: *depth_value* $\leftarrow$ Get depth at transformed coordinates
8: Visualize $\hat{\boldsymbol{n}}_{\text{door}}$ with its origin in the centroid C
9: Calculate (Δx_{ref}, Δy_{ref}, $\Delta \psi_{\text{ref}}$) using *normal_vector*, *door_center*, and *depth_value*
10: **return** (Δx_{ref}, Δy_{ref}, $\Delta \psi_{\text{ref}}$)

model performance the maximum number of detections is fixed to one, so that the robot can only detect one handle at a time. The output binary mask is then applied to the depth image to obtain a **segmented depth image**, as shown in Figure 4.18a. Note that in order to do this, the binary mask is resized and aligned with the depth image resolution using the pre-calibrated intrinsic and extrinsic parameters of the Intel RealSense camera.

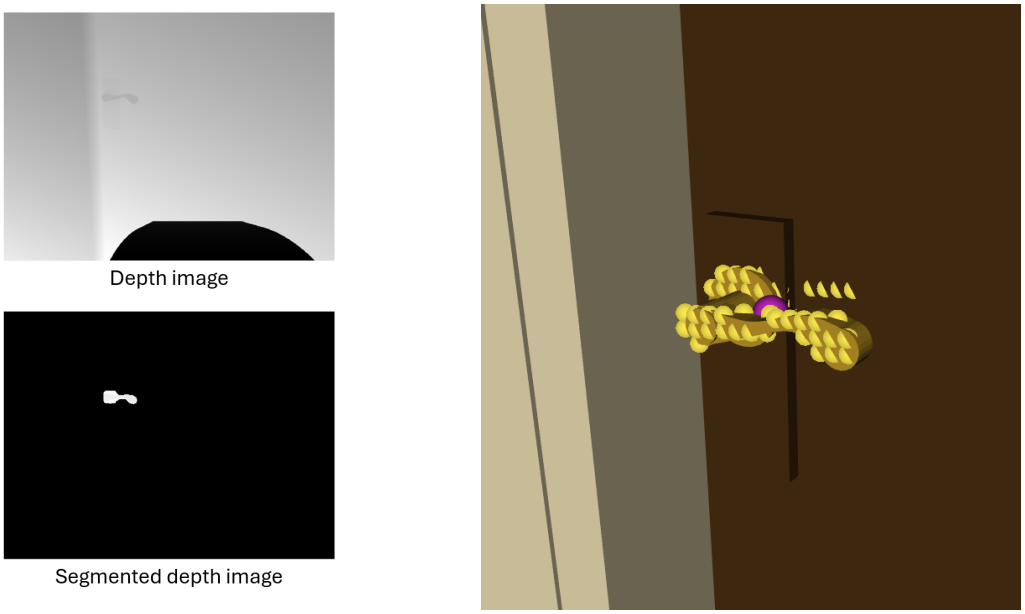


Figure 4.18. Handle detection: (a) Segmented depth image, (b) Segmented point cloud.

Finally, this node **computes the point cloud** derived from the segmented depth image. In order to do this, Equations 3.11a and 3.11b are used. However, in order to significantly reduce the computational load, a **downsample factor** of 4 is applied so that only 25% of the depth image pixels are converted into 3D points, as shown in Figure 4.18b. The point cloud is published in a new topic to be read by other ROS nodes. The pseudocode for this ROS node is presented next.

Algorithm 4 Handle Segmentation and Depth Processing

Require: YOLO weights, SR305 camera RGB and depth images and (f_x, f_y, c_x, c_y)
Ensure: Point cloud $\mathcal{P}_{\text{handle}}$ and masked images
1: Initialize ROS node and YOLO model with pre-trained weights
2: Acquire RGB image I_{RGB} and depth image I_{depth}
3: $D_{\text{mask}} \leftarrow$ YOLO detect mask on I_{RGB}
4: $I_{\text{masked}} \leftarrow I_{\text{RGB}}$ masked with D_{mask}
5: $M_{\text{depth}} \leftarrow I_{\text{depth}}$ masked with D_{mask}
6: $\mathcal{P}_{\text{handle}} \leftarrow$ Generate 3D point cloud from M_{depth} using (f_x, f_y, c_x, c_y)
7: Publish I_{masked}, M_{depth}, and $\mathcal{P}_{\text{handle}}$

4.4.4. Point Cloud Analysis

The ROS node in charge of processing the generated point cloud to extract relevant data is `handle_centroid`. In particular, it applies advanced data analysis techniques such as RANSAC and PCA to obtain the handle centroid and its orientation. The pseudocode of this node is shown in Algorithm 5.

Algorithm 5 Point Cloud Centroid and Orientation

Require: Handle pointcloud $\mathcal{P}_{\text{handle}}$, door normal vector $\hat{\boldsymbol{n}}$
Ensure: Handle centroid C and principal axes visualization
1: Initialize ROS node, subscribers and publishers
2: np_points ← Convert $\mathcal{P}_{\text{handle}}$ to points array np_points
3: $C \leftarrow$ mean(np_points)
4: Publish centroid as PointStamped
5: Fit RANSAC model to np_points
6: Extract inliers inlier_points
7: **if** inlier_points < 2 **then**
8: ⌊ **return**
9: Perform PCA on inliers to get the first principal component x_axis
10: Compute z_axis ← cross_product($\hat{\boldsymbol{n}}$, x_axis)
11: Normalize x_axis and z_axis
12: Compute y_axis ← cross_product(z_axis, x_axis)
13: Ensure axis consistency
14: Publish axes as Marker

First, this node subscribes to the point cloud topic and calculates its **centroid** by converting it to a numpy array and calculating its mean. Then, a **RANSAC regression** algorithm is used to fit the points to a line to remove possible outliers. Next, **PCA** is applied to obtain the principal components of the inlier points. Just the principal component with the largest variance, corresponding to the handle main direction, is normalized and used. Then, the node is subscribed to a topic where the normal vector to the door wall is published. Using these two normalized vectors, the third direction, perpendicular to both the handle direction and the wall normal is obtained. Finally, a **marker** is pro-

grammed to visualize the handle orientation in RViz with its origin in the handle centroid. In order to avoid continuous direction change in the principal component, a consistency evaluation function is introduced to change the sign if necessary. Both the centroid and the orientation axes of the handle can be seen in Figure 4.19.

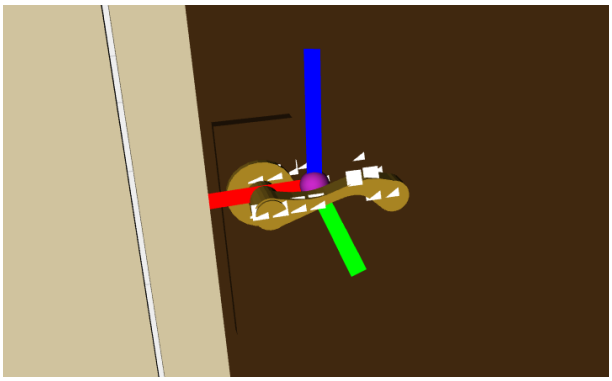


Figure 4.19. Handle centroid and orientation axes calculation.

4.4.5. Door Radius Estimator

The fourth and last node of the vision sub-system, `door_params` incorporates a ROS service that computes and returns the door's swing radius, R_m. In order to calculate it, the normal distance to the door's wall plane from the arm camera, $d_{\perp_{\text{arm}}}$, is required. Though it is true most doors have a similar rotation radius of around 0.7 m, it was decided to implement this code to generalize more the robot's vision capabilities so that it would be able to open any type of handle door.

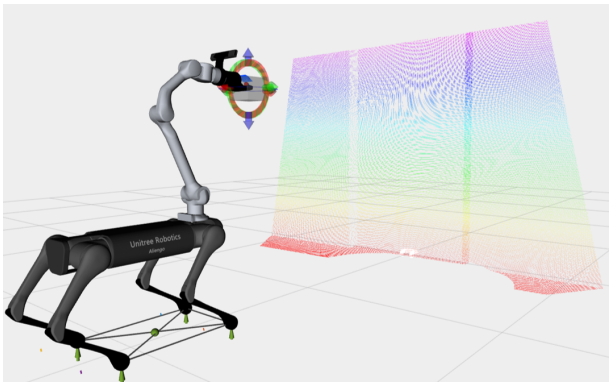


Figure 4.20. Pointcloud for door normal distance estimation.

The normal distance to the door, $d_{\perp_{\text{arm}}}$, is calculated in the same way as it was done in Algorithm 3. In this case, though the normal distance to the arm camera could be estimated from the one detected in the robot base, it was decided that the system would be more robust if calculated from the arm perspective. This is because this is the part of the robotic system in charge of the manipulation tasks that will be closer to the interactive objects, and so, it will provide better measurements.

In order to compute the door radius, R_m, this node subscribes to the handle mask topic and computes its centroid. To be able to compute the rotation radius it is also necessary to calculate the door edge that is opposite to the handle side. In order to do this, the node also subscribes to the RGB image topic and converts it to grayscale, since edge detection algorithms typically operate on this version. Canny edge detection is applied to this image to retrieve the edges.

The **Canny edge detector** algorithm from OpenCV first smoothens the image by applying a Gaussian blur and then finds the gradient of the image intensity at each pixel using the Sobel kernel (whose size was fixed to be 3×3), which transforms the image to something similar to what is shown in Figure 4.21a. Afterwards, non-maximum suppression is applied to the blurred edges image to obtain sharper edges by suppressing all the gradient values except the local maxima (locations with sharpest change of intensity value). Next, double thresholding is applied to the edges resulting in the calculation of weak and strong edges depending on whether they are above one or the two threshold limits (in the code these limits were set to 50 and 150, respectively because they were tested to work reasonably well). Finally, the weak edges that are not connected to strong edges are suppressed, obtaining the final edges image, which can be seen in Figure 4.21b.

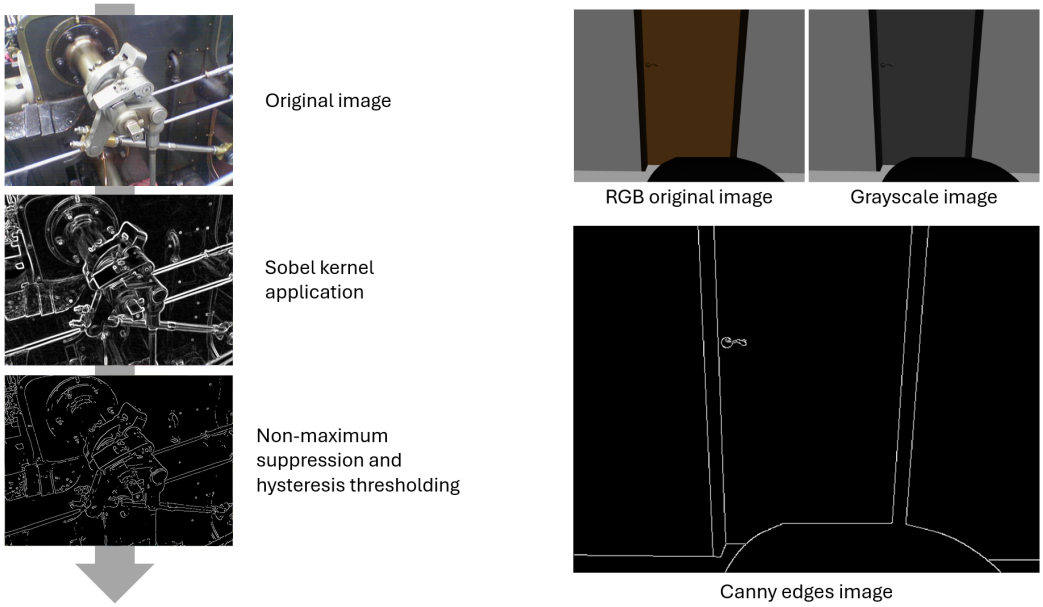
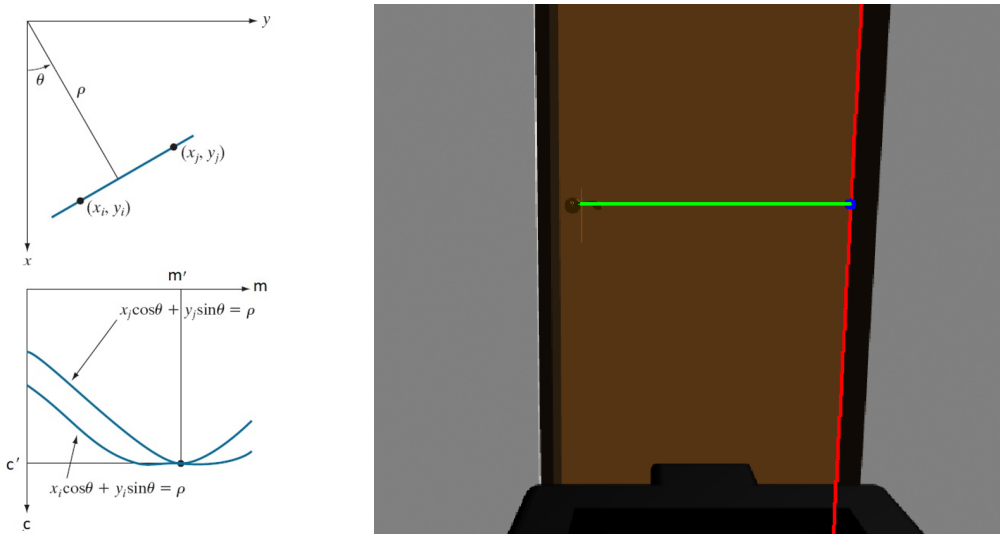


Figure 4.21. Canny edge detector: (a) Sobel kernel [126], (b) Door opening simulation.

Next, the OpenCV **Hough Line Transform** is applied to the edges binary image to isolate straight lines. It works by mapping each edge pixel onto the Hough space, where straight lines are described in polar coordinates by the distance from the origin to the closest point on the straight line (x_0, y_0), ρ; and the angle between the x-axis and the line connecting the origin with its closest point, θ (see Figure 4.22a). In the Hough space

an accumulator array is used to count the occurrences of each (ρ, θ) pair. Each point in the image votes for all lines that could pass through it in the Hough space. The more points (or votes) that accumulate, the higher the likelihood that a line exists there. The minimum number of intersections needed to detect a line was set to 200 with unitary resolutions for both ρ and θ.



(a) **(b)**

Figure 4.22. Hough Line Transform: (a) Hough space [127], (b) Door opening simulation.

These lines are filtered to just keep the **vertical** ones (since it is supposed the robot will estimate the door radius in a stable standard position). This is done by checking that the difference of x-coordinates of two far away points belonging to each straight line do not overpass a certain threshold, fixed in 200 pixels. From the resulting approximately vertical lines, two on the handle side and two on the opposite side; the ones on the handle side and the one on the opposite side corresponding to the door frame are discarded. This is done by taking the second one with the maximum x-axis distance to the previously calculated handle mask centroid (the first one if only one line is detected in the opposite side of the handle), which is shown in red color in Figure 4.22b. This resulting **maximum distance** is the radius in pixels, R_p, which is shown in green color in Figure 4.22b. To convert it to meters, the distance to the door, $d_{\perp_{\text{arm}}}$; and the focal length, f_x, are used as indicated in Equation 4.4:

$$R_m = R_p \cdot \frac{d_{door}}{f_x} \tag{4.4}$$

Additional code has been written to manage possible error situations such as not having the required variables to compute the door radius. The pseudocode corresponding to this node is shown in Algorithm 6.

Algorithm 6 Door Radius Estimation

Require: Point cloud $\mathcal{P}$, RGB image I_{RGB}, focal length f_x, and handle mask D_{mask}
Ensure: Door radius R_m

```
Initialize ROS node, subscribers, and publishers
C_mask ← centroid(D_mask)
gray_img ← img2grayscale(I_RGB)
edges ← canny_edge_detector(gray_img)
hough_lines ← hough_transform(edges)
if C_mask and hough_lines found then
    a ← get_axis(hough_lines)
    R_p ← get_radius(C_mask, a)
    Publish annotated image
Convert P to PCL and apply RANSAC to find plane π
if π found then
    Calculate door normal distance d_⊥arm
R_m ← compute_radius(R_p, f_x, d_⊥arm)
Provide R_m via service response
```

4.5. Planner Algorithm

The planner algorithm in this project plays a crucial role in guiding the robot for specific tasks such as door opening and locomotion. However, it is important to note that the planner developed here is **not a general-purpose navigation planner for obstacle avoidance**, as this was beyond the scope of this project. Instead, the primary focus was on the control aspects of the robot, leveraging the existing NMPC controller for locomotion tasks, and developing a specific planner for **door manipulation**. In general locomotive tasks, such as the SAR environments where the robot will be tested, the trajectory publisher integrated within the NMPC controller assumes the role of the planner, generating references for the base from the end-effector commands. For door-opening tasks, a specialized planner was developed within the `qm_planner` ROS package.

4.5.1. Trajectory Publisher for Locomotion Tasks

In pure locomotion tasks, the NMPC controller's trajectory publisher is used to manage the robot's movement across unstructured terrains. Note that it does not account for obstacle detection through perception, but several previous projects in the lab have already dealt with this, and the solution presented in this work could be easily integrated with them in the future. The robot's trajectories are computed based on the NMPC's state estimation and end-effector position, and it was modified from the one in the original repository to include the foot heights and a boolean for base fixation to handle constrained movement during tasks such as door opening.

It basically accounts with a time-to-target estimator, which calculates the time required for the robot to reach a target position, considering both displacement and rotation. It calculates the maximum of both linear and rotational movements to ensure the robot moves at an appropriate speed based on both displacement and rotation.

With this time estimation, the trajectories publisher generates a trajectory that guides the robot from its current state to the target pose defined from the end-effector position target, using both base and end-effector data. For this, it considers if the boolean "fixed base" flag is activated, to restrict the movement in the y-axis ensuring the robot remains while executing specific tasks. This feature was introduced for door opening to ensure the base is not aligned with the arm position during door traversal since this led to collision with the door frame.

It basically enables to create trajectories discretized in the time-to-target period estimation, through the definition of the position and orientation of both the end-effector and the base. The end-effector trajectories are directly defined from an interpolation between the target final position and orientation and the current one. With respect to the base, roll and pitch are continuously constrained to zero during this work to ensure maximum stability (however, this could be modified for enabling the robot to incline to pick an object for the ground for example). The base height is fixed to the defined CoM height (0.4 m in most cases) plus the average height of the feet in contact with the ground (to allow for vertical objects traversal, such as stairs without undesired collisions). The yaw rotation was left free (but it can be fixed to continuously align with the end-effector yaw or to zero if no base yaw is desired), and the x-position of the base is determined by the one of the end-effector minus the distance in that axis from the base to the arm. Finally, with respect to the y-coordinate, if the fixed base flag is active then the base is fixed in that coordinate to ensure it passes through the door frame safely.

4.5.2. Planner for Door Opening Tasks

For door-opening tasks, a specific planner was developed within the `qm_planner` ROS package. This planner handles the complex task of positioning the robot with respect to a door and autonomously executing the steps necessary to open it.

4.5.2.1. Robot Base Reorientation with Respect to the Door

The reorientation process in the robot's planner module is crucial to ensure that the robot is aligned with the door for the subsequent task of opening it. The robot may start from positions and orientations that are not aligned with the door, and this process ensures that the base of the robot is parallel to the door's wall plane. This ensures an optimal approach to handle and open the door.

The provided algorithm consists of several steps and components to achieve this. The process involves reading the robot's current position, calculating the target pose (position and orientation), and reorienting the robot based on the difference between the current and desired orientation with respect to the door. The pseudocode for the program is presented in Algorithm 7.

Algorithm 7 Robot Base Reorientation with Respect to the Door

Require: Initial target pose, current end-effector pose, door reference values
Ensure: Robot base aligned with door for door opening

```
Initialize ROS node, publishers, subscribers, and service client
Set initial target pose and publish it
Set position_reached to false
while ROS is running do
    Update current end-effector pose and check if target is reached
    if position_reached is true then
        if initial_position_reached is false then
            Set initial_position_reached to true
            Retrieve door reference values
            Publish gait command "trot" and adjust target pose
            Set position_reached to false
        else if door_initial_position_reached is false then
            Publish gait command "stance" and check door position
            if position is acceptable then
                Set door_initial_position_reached to true
            else
                Reposition robot and publish "trot" command
                Set position_reached to false
        else
            Wait for new target
    else
        if new target detected then
            Publish target position and update last pose
    Sleep for loop duration
```

The algorithm begins constantly tracking the robot's current pose and checking whether it is near the target position. This is done through a function that retrieves reference values for the door's position and orientation relative to the robot through a ROS service. Position updates are handled by a callback function that monitors the distance between the robot's current and target poses. When the robot is within a small threshold (0.07 m), it signals that it has reached the desired position and prepares for the next step. Stance command is issued for more accurate reference measurement for better alignment. The loop continues adjusting the robot's position until the door's reference values indicate that the robot is properly aligned. Once the alignment is confirmed, the robot is ready for the door-opening task.

4.5.2.2. Door Detection, Approach, and Opening

The door-opening planner developed in this project allows the robot to autonomously approach and open a push-type door (pull-type door opening was not finally tackled due

to time constraints). The process involves a sequence of steps, such as detecting the door handle, computing the door's opening radius, and commanding the robot to manipulate the door based on its type (left or right handle). Below is a detailed explanation of how the system functions:

1. **End-effector pose and target pose handling:** The robot continuously updates its knowledge of the end-effector's position using data published by the robot's state observation system. A callback function subscribes to the current end-effector position topic and continuously checks if the target position has been reached within a certain threshold.

2. **Centroid detection and transformation:** The planner uses data from the vision module to detect the 3D location of the handle centroid of the door handle. This information is captured via a ROS topic subscription, and the centroid coordinates are transformed from the camera frame to the world frame. These transformed coordinates are used to update the robot's target position, ensuring that the end-effector is correctly aligned with the door handle.

3. **Distance threshold handling:** As the robot approaches the door, a distance threshold is used to stop the centroid update once the robot is sufficiently close to the handle. This prevents the robot from continuously adjusting its position unnecessarily, and also door opening fails provoked by perception issues associated to excessive closeness to the door. This way, the normal distance to the door is continuously monitored to deactivate centroid target updates.

4. **Gait command control:** The robot uses a series of predefined gait commands, such as "trot" and "stance", to control its movement. For instance, when the robot needs to take accurate door radius measurements, the "stance" command is issued, while when the robot needs to approach the door handle, the "trot" command is issued.

5. **Radius calculation and door type detection:** One of the critical steps in the door-opening process is calculating the door's radius (i.e., the distance between the door' hinge and handle). This is done through the previously commented ROS service, which is called once the robot has reached the initial position. The service also returns the door type, indicating whether the handle is on the left or right side. This information is crucial because it determines the direction in which the robot must apply the force to open the door.

6. **Opening the door:** After calculating the radius and reaching the door handle, the robot begins the door-opening process. The robot pushes the door by adjusting

its target position in small increments. During this process the fixed base flag is activated to let the base target position be in the middle of the door frame. Note that this is needed to avoid the collision between the base and the door frame, since the base tends to align with the end-effector without any constraints.

Moreover, this constraint is not imposed from the beginning to the base because it was observed to provoke instability in the door handle approach phase (probably due to the overrestriction of the optimization control problem). the robot adjusts its movement to push the door in the appropriate direction depending on the handle location side. Then the door is gradually opened in small angle increments defining the end-effector target pose. Once the door is opened, the robot moves to its final positions, completing the door-opening task.

7. **Handling different scenarios:** The planner accounts for several scenarios during the door-opening task:

   - **Reaching initial position:** Before starting the door-opening process, the door must reach an initial target position near the door (around 1.5 m away and with the arm positioned at 1 m height from the ground). Once it is confirmed that the initial position is reached, the robot begins the door handle approach phase.
   - **Receiving a new target centroid:** If a new centroid data is received, the robot updates its target position accordingly and starts moving toward the new centroid.
   - **Pushing the door:** If the door is not fully opened the robot continues to push in the correct direction until the final door angle is reached.
   - **Task completion:** Once the door is fully opened, the robot transitions to its final position, confirming that the door-opening task is complete.

8. **Positon command publishing:** Throughout the process, the robot continually publishes its target position. This ensures that the NMPC controller is aware of the desired positions, allowing the robot to move towards its target smoothly.

In typical door-opening tasks involving mobile manipulators, the problem is often simplified by using separate systems**problem is often simplified by using separate systems** for locomotion and manipulation, fixing the robot's base in a stable position before executing the manipulation. While this approach reduces complexity and ensures stability, it lacks dynamism, takes longer to complete, and limits adaptability to unexpected environmental changes. The proposed solution in this work integrates locomotion and manipulation into a single system, allowing for a more fluid and dynamic operation

without the need to fix the base. Although this solution is still in its initial stages and could be improved to handle scenarios like door pulling, spring-loaded doors, or awareness of the door's state during opening, it presents a step toward more autonomous and efficient robotic manipulation tasks.

Algorithm 8 Door Opening Planner

Require: Initial target pose, end-effector state, door handle centroid, door handle type (left or right)
Ensure: Door opened and final position reached

```
Initialize ROS node and publishers/subscribers
Initialize initial target pose
Set position_reached to false and wait for target to be reached
while ROS is running do
    if position_reached is true then
        if initial_position_reached is false then
            Set initial_position_reached to true
            Call service to compute door radius and type
            Publish gait command "trot" and wait 3 seconds
        else if new_centroid_received is true then
            Move to new centroid position and reset new_centroid_received flag
        else if start_opening is true and current_angle < final_angle then
            Adjust end-effector position and orientation to push the door
            if door_type is right handle then
                Adjust y position positively and rotate end-effector clockwise
            else if door_type is left handle then
                Adjust y position negatively and rotate end-effector counterclockwise
            if current_angle ≥ final_angle then
                Set door_opened to true and stop opening
            Increment current_angle by delta
        else if door_opened is true then
            Move end-effector to final position after door is fully opened
        else
            Wait for new target or adjustments
            if distance_threshold_met is false and door is not opened then
                Publish gait command "stance" to push the door further
                if position_reached is true then
                    Publish gait command "trot" and center the base
                    Set start_opening to true
    else
        if first_publish or new target position is detected then
            Publish target position to the positionCommandPublisher
            Update last_target_pose and reset first_publish flag
```

4.6. Software Integration

The software integration phase of this project brought together the key modules—control, vision, and planner—into a unified ROS-based architecture which can be seen in Figure 4.23 extracted using the `rqt_graph` tool. This section will explain how the different components interact and communicate with each other through the ROS ecosystem, focusing on the real-time communication and coordination between the NMPC controller, the vision system, and the high-level planner.

At the core of the architecture is the **control module**, which is responsible for executing the low-level commands generated by the NMPC controller. The controller node receives a series of desired base and end-effector trajectories from the target trajectories publisher node. This node subscribes to the observation from the state of the quadruped manipulator and the interactive RViz marker to finally publish the robot position target.

The **vision module**, which provides visual and depth information, is integrated via the robot's camera sensors: `/camera1`, which is the SR305 camera, and `/camera2`, which is the D435 camera. This module consists of four ROS nodes responsible for segmenting and generating the handle point cloud (`/handle_pc`), determining the handle centroid and orientation (`/handle_centroid`), calculating the door radius (`/door_params`), and determining the door center axis (`/door_center`). These nodes each publish on some topics which are required for the planner, such as the handle centroid, but also have optional publishing (through a publish flag) for visualization of intermediate results, such as the edge detection image of the door.

The **planner module** is responsible for generating the high-level motion plans that guide the robot to perform the door opening task. Two planner nodes are used, though only one is active each time: `/go_to_door` (for repositioning normal to the door) and `/open_door` (for performing the opening task). The planner takes inputs from the vision system and from the current state of the robot to compute the end-effector positions that are used as a reference by the controller's trajectory publisher node from `/planner/cmd`. These planner nodes additionally publish the gait commands in `/gait_command_topic`, which interfaces with the gait topic publisher node of the controller, and a boolean on `/fixed_base` to switch the target trajectories publisher constraints (useful for door traversal).

The overall communication between these three modules is critical to ensuring that the robot can execute complex tasks such as door opening. Additional nodes are used to publish the transforms of the walls and the door,which is considered by ROS to be a 2-DOF robot, to be able to properly visualize them in RViz.

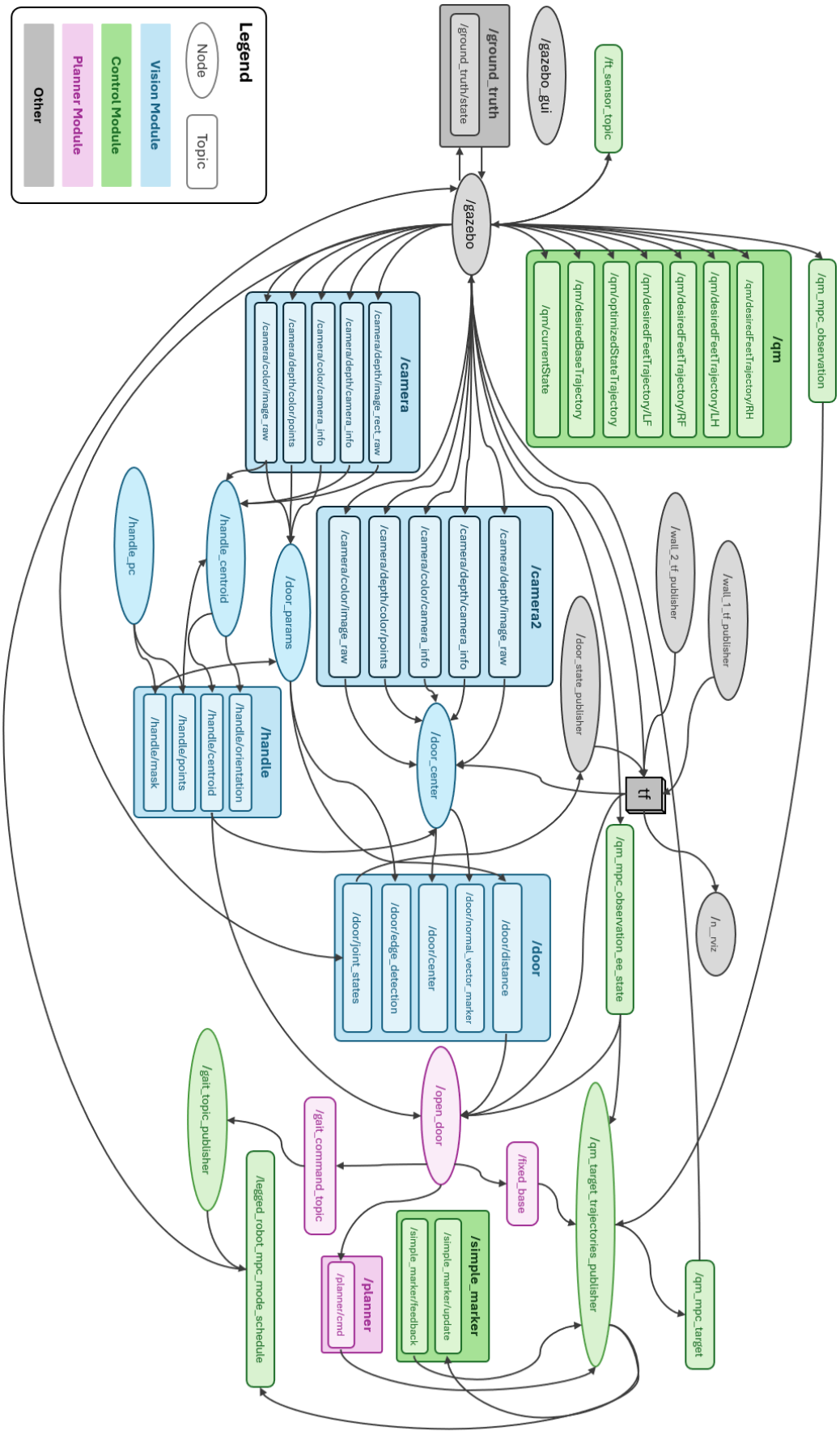


Figure 4.23. ROS nodes and topics program architecture.

4.7. Hardware Integration

Once the software was finished, the whole hardware system was assembled. In order to do this, a 10 mm thick **methacrylate plate was designed** and milled to serve as interface between the Z1 arm and the quadruped since the original manufacturer support was not available in the lab. Four of the holes of this plate are threaded to enable the fixation of the Z1 arm, while the other four are through holes that permit to fix the plate with the arm to the quadruped through sliding nuts located in the two metallic profile rails. The plate design and assembly can be seen in Figure 4.24.

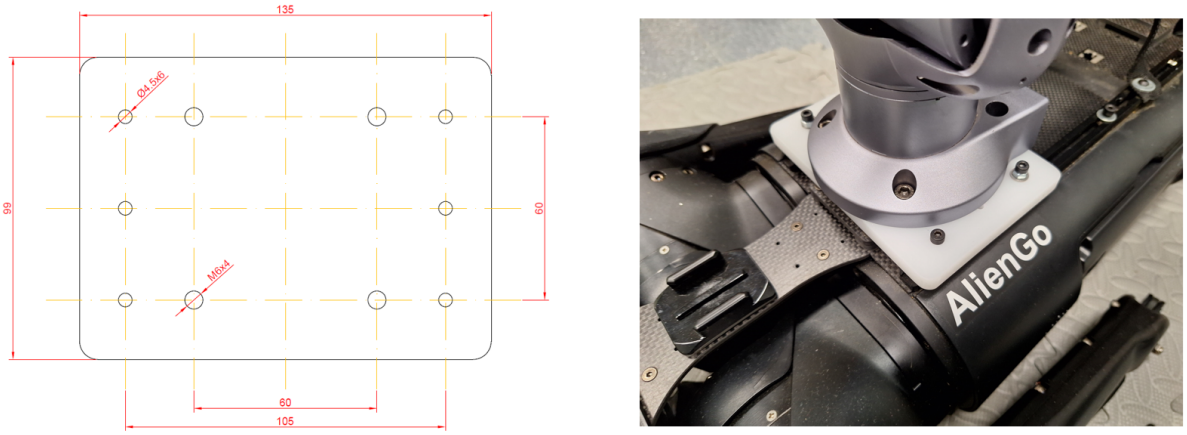


Figure 4.24. Methacrylate plate design and assembly.

Then, the SR305 camera support components and cover were 3D-printed using a Prusa MK4 3D printer [128]. The support pieces were printed in Polylactic Acid (PLA) at 20% of infill, while the camera cover was printed in Thermoplastic Polyurethane (TPU). The resulting pieces can be seen in Figure 4.25.

(a) **(b)**

Figure 4.25. 3D-printed camera components: (a) Support, (b) Cover.

With respect to the connections, the AlienGo quadruped communicates with the Z1 manipulator through an Ethernet cable. The power 24 V connection required for the

Z1 arm can't be directly made to the AlienGo since its maximum output voltage is 19 V. When this was discovered did not make much sense since the arm is supposed to be adapted to the quadruped platform and both are from the same manufacturer. A transformer would be needed for making the connection onboard, but since there was no availability, the connection of the arm was directly made to the room plug through its corresponding charging transformer. This limitation is acceptable for testing purposes since a plug extender can be used, but in the final implementation a transformer would be needed.

Finally, since the Z1 arm may be damaged or damage the door during interaction testing, it was decided to add silicone pads to the gripper to prevent this. These silicone pads were attached through double face tape, resistant but easily removable, as it can be seen in Figure 4.26.

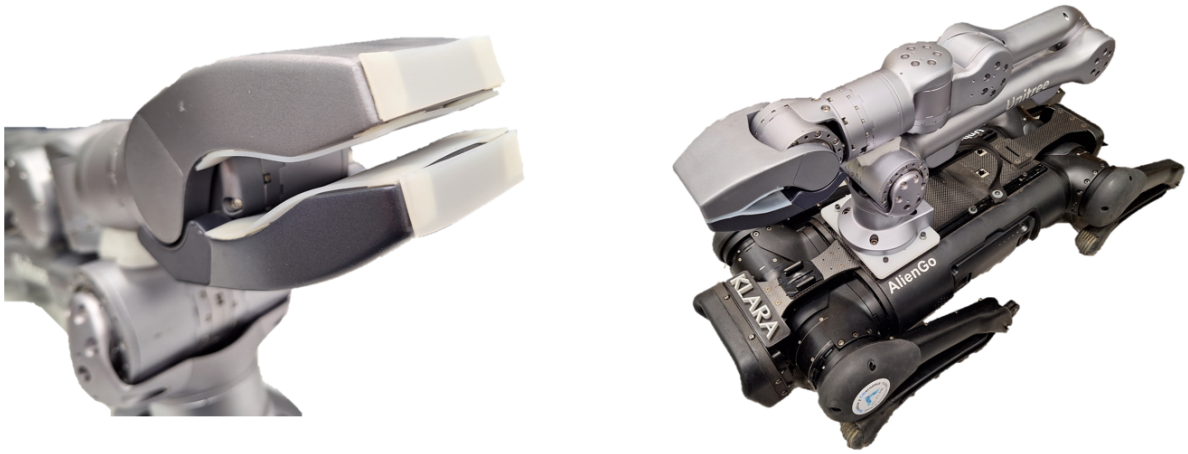


Figure 4.26. Z1 gripper silicone pad protections.

Finally, the **controller code was adapted for real hardware implementation** through the Z1 SDK. The GitHub repository for the Kinova Jaco was taken as a reference and the functions concerning the robotic arm were accordingly substituted with the equivalent ones provided by **Unitree's Z1 SDK**. This hardware adaption of the code is available as a separate branch in the repository since several controller files were modified.

Chapter 5

Results and Discussion

This chapter presents the results of both simulation and real-world testing phases, focusing on the system's performance in various scenarios. During simulation testing, the system's ability to traverse non-structured environments and its door-opening capability were evaluated. The real-world testing phase further assessed the implementation of the NMPC controller on the AlienGo robot and the accuracy of the vision system. The final complete hardware integration with the Z1 was done but there was no time to test the whole-body NMPC controller on the real platform due to time constraints. The discussion highlights the effectiveness, challenges, and improvements observed throughout these tests.

5.1. Simulation Testing

This section outlines the evaluation process carried out in **simulated environments** to validate the performance and robustness of the quadruped manipulator system before transitioning to real-world testing. This testing phase was crucial for assessing the controller's capabilities in handling various complex and unstructured scenarios. Additionally, it enabled to discover and highlight the system limitations for future work.

The simulation environments were carefully designed to evaluate specific features, present in real-world obstacles and situations during SAR. These tests include tracking the end-effector trajectory and navigating through non-structured terrains such as pallet obstacles, unstable platforms, stairs, ramps, tunnels with uneven terrain, and maze-like paths. Additionally, a dedicated simulation scenario for door opening was created to assess the manipulator's precision and effectiveness in handling such tasks.

Through these simulations, the robot's ability to follow precise trajectories, maintain stability on uneven surfaces, and execute complex maneuvers was thoroughly evaluated, providing critical insights into the system's strengths and areas for further improvement before proceeding to real-world testing.

5.1.1. End-Effector Trajectory Tracking

In this simulation test, the objective was to assess the quadruped robot's ability to accurately **track a circular trajectory** with its end-effector. This type of trajectory is commonly used in robotic systems to evaluate the precision and stability of the controller when performing continuous and smooth motions.

The test was designed to have the quadruped's end-effector follow a circular path in a horizontal plane while maintaining constant yaw orientation. The circle has a radius of 1.2 m. The robot' end-effector was required to start at the circle's center and then progressively move along the circular path while maintaining a constant height. The code for this test can be found in the `qm_planner` package.

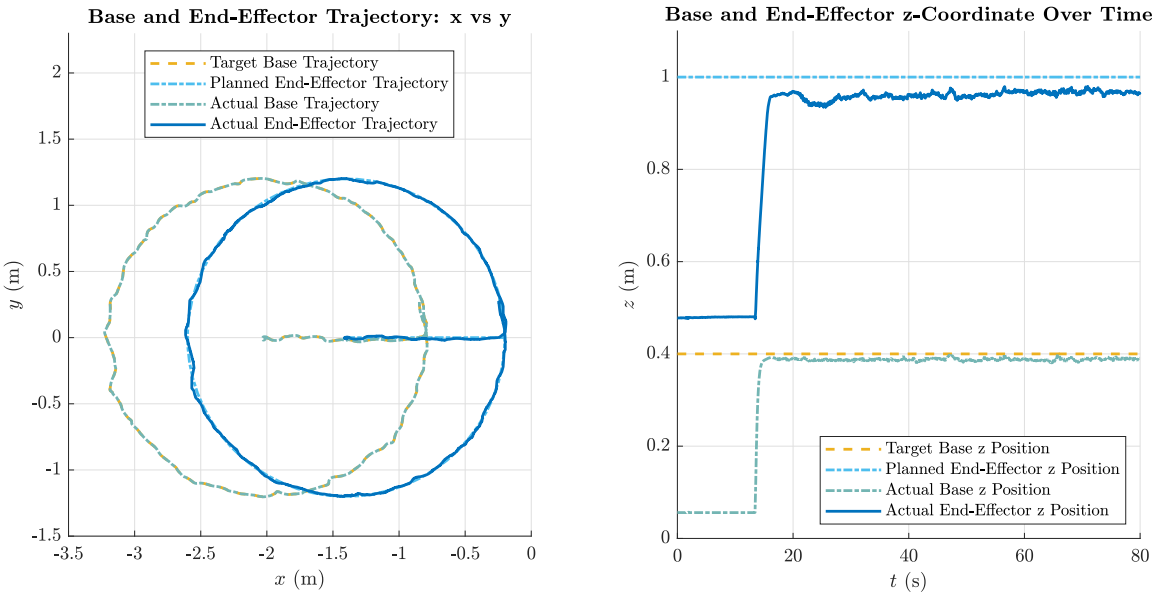


Figure 5.1. Comparison between planned and actual positions of the quadruped manipulator's base and end-effector.

Figure 5.1 shows the base and end-effector trajectories and compares it with the actual followed path. The original trajectory that is introduced into the controller as reference input is the planned end-effector trajectory, which is seen in the figure to be an almost perfect circle. The controller is programmed with an equality constraint in the NMPC optimization problem and with an equality task in the WBC controller to ensure end-effector position tracking. The target base trajectories are calculated by the trajectory publisher from the user input end-effector planned trajectory and the current state of the

system. It can be seen that the controller accurately tracks the desired trajectory. In the z-coordinate it is seen how when the controller starts working the base and the arm move towards their reference heights. The base height is very accurately track while the end-effector's one is less accurate. This is reasoned to be due to its larger kinematic chain to the base which amplifies the existing position errors. Another possible contribution could be the fact that the end-effector position tracking task belongs to a lower hierarchy level that the base height tracking.

To quantitatively assess the performance, the **Root Mean Squared Error (RMSE)** error metric was employed, which is particularly sensitive to large deviations, thereby penalizing heavily significant errors. The RMSE was computed for each coordinate separately to provide a detailed component-wise error analysis.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\text{desired}_i - \text{actual}_i\right)^2} \tag{5.1}$$

	Base			**End-Effector**		
	$\boldsymbol{x}$	$\boldsymbol{y}$	$\boldsymbol{z}$	$\boldsymbol{x}$	$\boldsymbol{y}$	$\boldsymbol{z}$
RMSE (cm)	2.01	1.60	2.68	26.95	7.72	5.53

Table 5.1. RMSE values for base and end-effector trajectories.

The obtained RMSE values corresponding to each of the base and end-effector trajectories components can be seen in Table 5.1. In general **low values** are observed, with higher values for the end effector for the commented reasons. The x component is specially large because of the starting movement phase. In addition to this, it was considered interesting to analyze how well the odometry measures the robot's actual state. As commented previously, this odometry measurements are obtained through a Kalman filter that uses visual odometry from the front camera, together with the motor encoders and the IMU. Figure 5.2 shows a **comparison between the odometry measurement and the ground truth** for each position and orientation component (in Euler angles). The low RMSE values in Table 5.2 show the odometry is reliable.

	Base Position			**Base Orientation**		
	$\boldsymbol{x}$	$\boldsymbol{y}$	$\boldsymbol{z}$	$\boldsymbol{\phi}$	$\boldsymbol{\theta}$	$\boldsymbol{\psi}$
RMSE	2.19 cm	1.91 cm	0.9 cm	0.235°	0.304°	0.516°

Table 5.2. RMSE values for base odometry position and orientation.

Additionally, in Figure 5.2 it can be observed how the robot tends to maintain the base parallel to the ground without rotations due to the imposed constraints.

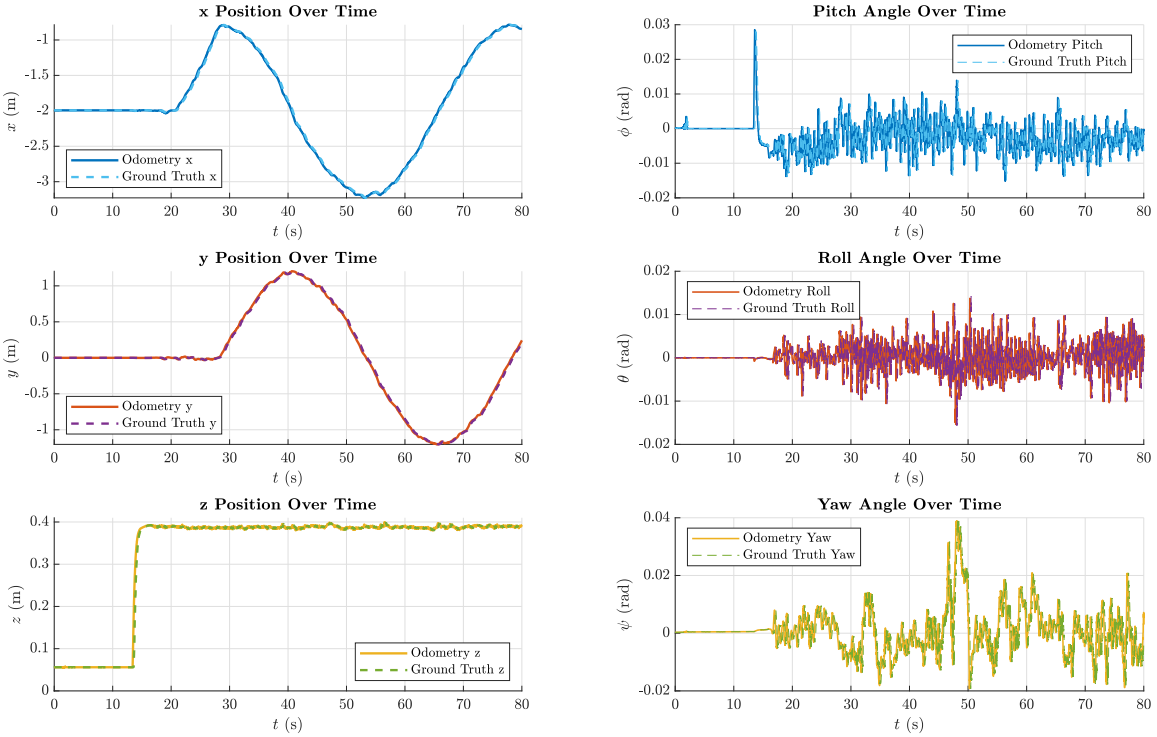


Figure 5.2. Odometry and ground truth base position and orientation comparison.

5.1.2. Non-Structured Environment Traversal

This section details the testing of the quadruped robot's ability to navigate various unstructured environments, simulating the challenges typically encountered in SAR operations. To focus on the robot's locomotion capabilities, the tests employed a separated NMPC controller. This approach was chosen after observing that the added mass and dynamics of the robotic arm caused significant destabilization in more complex environments, making it necessary to isolate the locomotion system for accurate assessment.

While the robotic arm has the potential to enhance the quadruped's stability by dynamically adjusting its position, it was decided to keep the arm inactive to **focus on evaluating the quadruped's locomotion** performance in isolation. This way, the NMPC controller was solely responsible for managing the quadruped's movement, while the arm's joints and gripper were controlled by separate PID controllers. In terms of gait pattern, a standing trot was predominantly used during the tests due to its superior stability, as it ensures all four feet maintain brief contact with the ground during each gait cycle.

5.1.2.1. Pallet Obstacles Path

This test was designed to evaluate the quadruped's ability to navigate a series of obstacles with varying heights. These obstacles simulate uneven terrain that may be

encountered in real-world SAR operations. The focus of this test was to assess the NMPC controller's effectiveness in maintaining stability and smooth movement over **challenging, discontinuous high surfaces**. It must be noted that no perception was included into the NMPC model to optimize the step placement performance. For this test, the feet step height was fixed to 26 cm for ensuring comfortable traversal, the CoM height to 0.36 m, and the reference velocity to 0.1 m/s. Figure 5.3 shows the path followed by the robot to traverse the pallet obstacles.

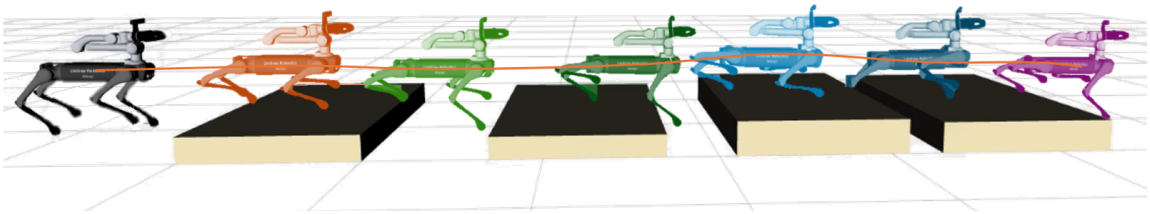


Figure 5.3. Pallets path traversal.

Since the robot lacks the ability to anticipate when a pallet will appear in its path, the performance of the system relies heavily on the fast, **reactive control** provided by the NMPC. This is particularly evident when dealing with the taller pallets, where the feet suddenly drop into the gaps between the pallets. Despite these challenges, the controller performs well, allowing the robot to recover and continue its trajectory, as illustrated in Figure 5.4.

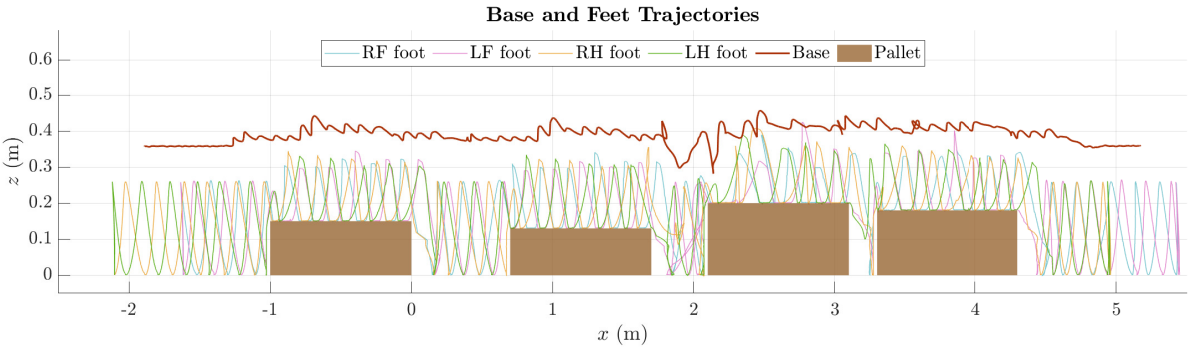


Figure 5.4. Base and feet trajectories during pallet traversal.

The system also demonstrates robust behavior when the robot attempts to climb onto the next pallet and the feet slip downward. However, velocity limitations were encountered; for instance, at a speed of 0.2 m/s, the quadruped became significantly destabilized and eventually fell. This issue could be mitigated, and performance significantly improved, by using the front depth camera to detect and segment optimal foot placement areas. Integrating this data into the NMPC algorithm would enable safer operation and more efficient stepping. The base orientation ground truth Euler angles' evolution during the traversal is shown in Figure 5.5. It can be observed how the NMPC manages to keep them around zero due to the imposed constraints in the trajectory planner.

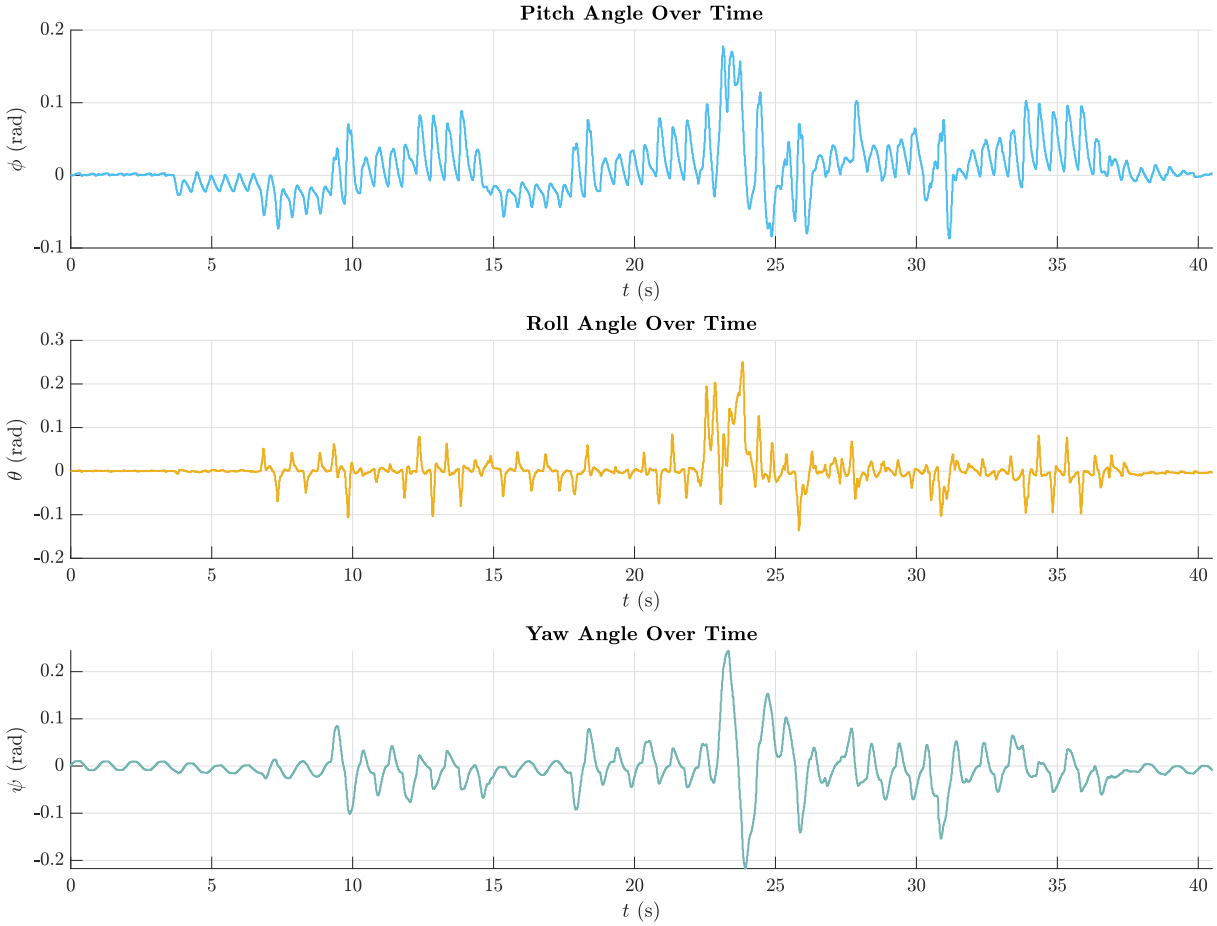


Figure 5.5. Ground truth base orientation during pallet traversal.

5.1.2.2. Unstable Platform

The unstable platform test challenges the robot's balance and adaptability by requiring it to traverse a platform that is deliberately unsteady. This scenario is intended to simulate situations where the ground beneath the robot may shift or tilt unexpectedly, testing the NMPC controller's ability to respond quickly and maintain the robot's equilibrium.

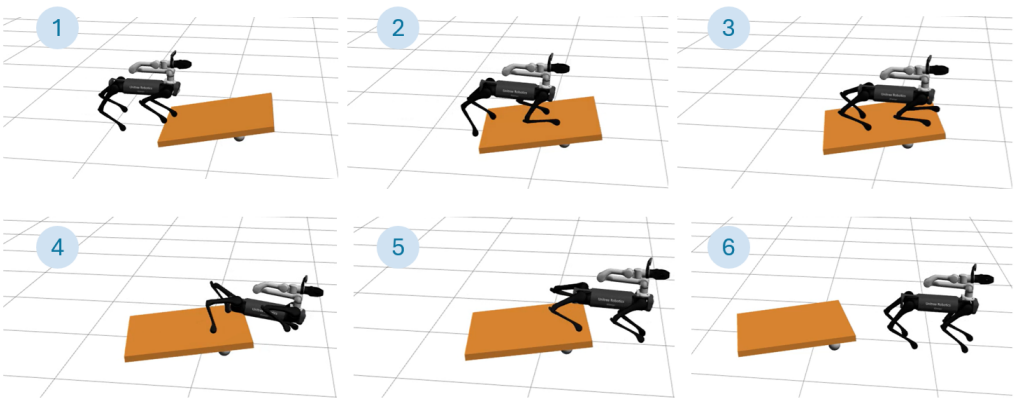


Figure 5.6. Unstable platform path traversal.

For this test the base CoM height was fixed to 0.4 m, with 0.15 m height steps and 0.2 m/s standing trot reference velocity. The main interest of this test was to check if the controller is able to maintain equilibrium even without having a model of the object dynamics with which the robot is interacting. Figure 5.7 shows the base positions and orientations' evolution during the test and how the NMPC controller manages to stabilize the robot even when external uncontrolled moving ground disturbs its trajectory.

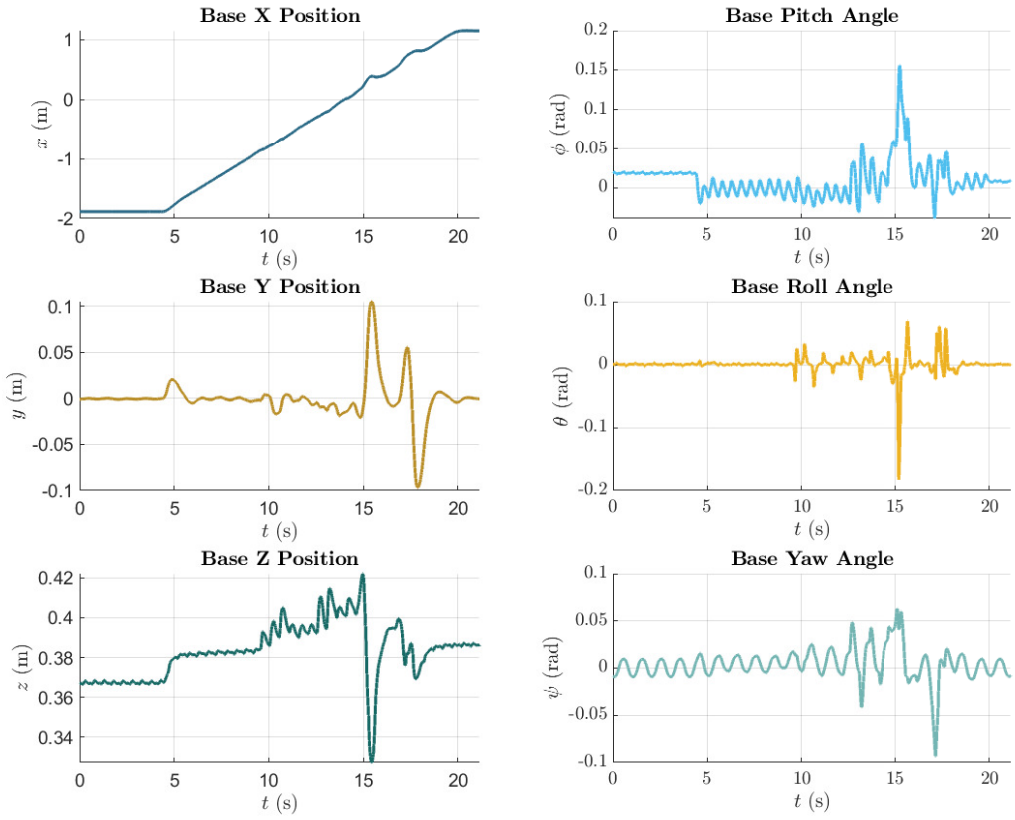


Figure 5.7. Ground truth base positions and orientations during unstable path.

Though it is not easily appreciated in Figure 5.6, the platform was supported by two non-aligned low friction ($\mu = 0.1$) 10 cm spheres; which led to **continuous moving of the platform**. In second 15 a significant disturbance occurs when the center of mass of the robot provokes a change in the torque direction of the platform, making it rotate to the opposite side. In the x-position evolution curve it can be seen how the robot manages to keep an almost constant velocity of 0.2 m/s as depicted by the constant slope. The y position suffers some variations to properly stabilize the robot at every moment.

5.1.2.3. Stairs and Ramp Path

In this stairs and ramp path test, the quadruped is tasked with ascending and descending a series of stairs and ramps. The CoM height was initially set to 0.4 m, the

step height to 0.15 m, and the friction coefficient from the stairs surface to $\mu = 0.5$. This test was crucial for evaluating the robot's capability to handle performance in maintaining stability and ensuring smooth transitions between different inclinations. Figure 5.8 depicts the trajectory followed by the quadruped during the test. Good performance was observed during both the upstairs and ramp tracks. However, at the beginning of the **downstairs stage**, the robot **failed** to maintain stability and collapsed. This behavior was repeatedly seen for different step and base CoM heights and gait patterns.

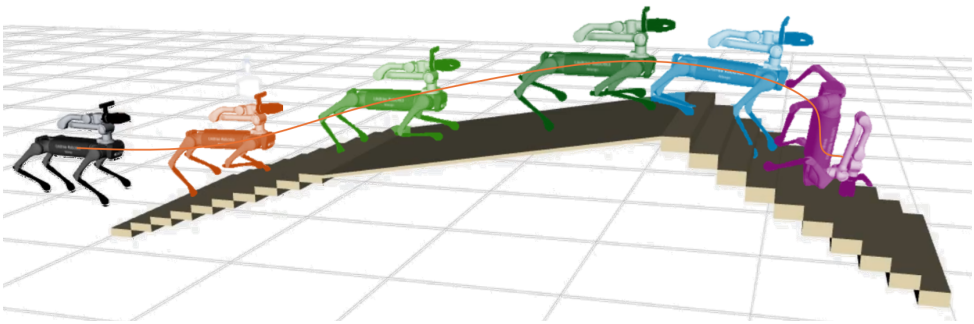


Figure 5.8. Stairs and ramp path traversal.

It was concluded that the failure to descend the stairs was primarily due to the excessive extension of the legs as they reached for the next step. This excessive leg extension elevated the robot's center of mass (CoM) relative to the contact surface, thereby increasing the system's instability. This issue was exacerbated by the small size of the stair steps, which provided limited surface area for secure foot placement. Furthermore, the absence of a perception system to optimize foot placement contributed significantly to the instability, as the robot was unable to adjust its gait or foot positioning dynamically to maintain balance. Consequently, these factors combined led to the robot's inability to descend the stairs effectively, highlighting the need for both mechanical adjustments and the integration of a perception system to improve stability in such scenarios.

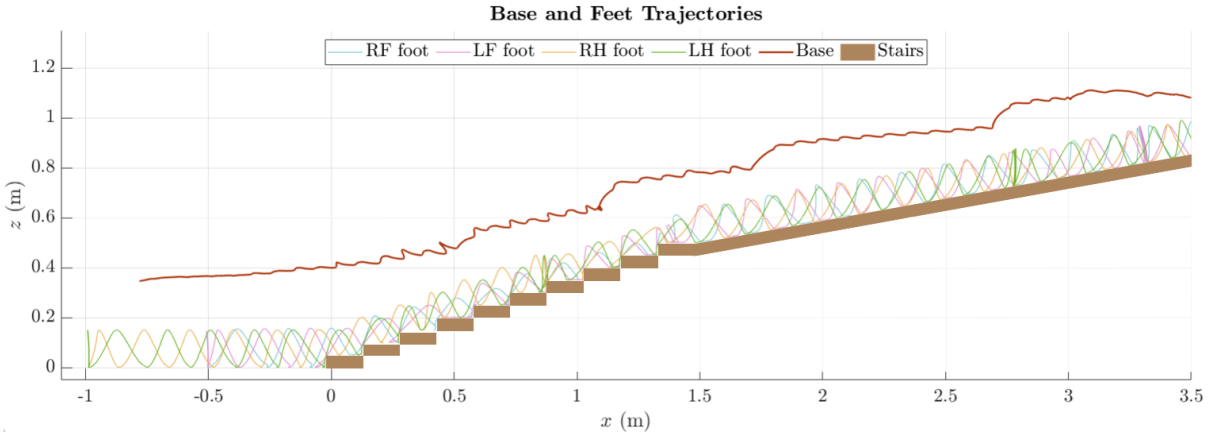


Figure 5.9. Base and feet trajectories during stairs traversal.

Figure 5.9 illustrates the evolution of the base height during the track, along with the feet trajectories leading up to the crash in the downstairs section. The figure shows a robust behavior during the climbing stage and how the height continuously increases to maintain the distance with the contact feet. Another factor that is thought to influence the instability of the system when going downstairs is the position of the robotic arm, which weights around 4 kg. A possible solution to this problem could be to modify the base pitch angle reference when the robot is going to go down stairs or ramps. This could work, specially when combined with the perception system, because that way the robot's CoM would keep closer to the surface, increasing its stability.

5.1.2.4. Tunnel with Uneven Terrain

This test simulates a confined and irregular environment, forcing the robot to navigate through a narrow tunnel with an uneven floor. This test assesses the quadruped's ability to maneuver in tight spaces while dealing with unpredictable ground conditions, highlighting the NMPC controller's capability in maintaining precise control under such constraints.

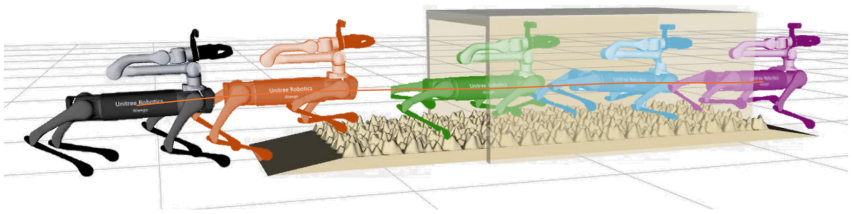


Figure 5.10. Tunnel path traversal.

For this test, the CoM base height was set to 0.25 m to be able to pass underneath a 0.6 m tall tunnel (without considering the uneven terrain elevation) with the manipulator arm and its depth camera support structure. The target velocity was set to 0.2 m/s with standing trot gait schedule. Figure 5.11 shows the robot's trajectories.

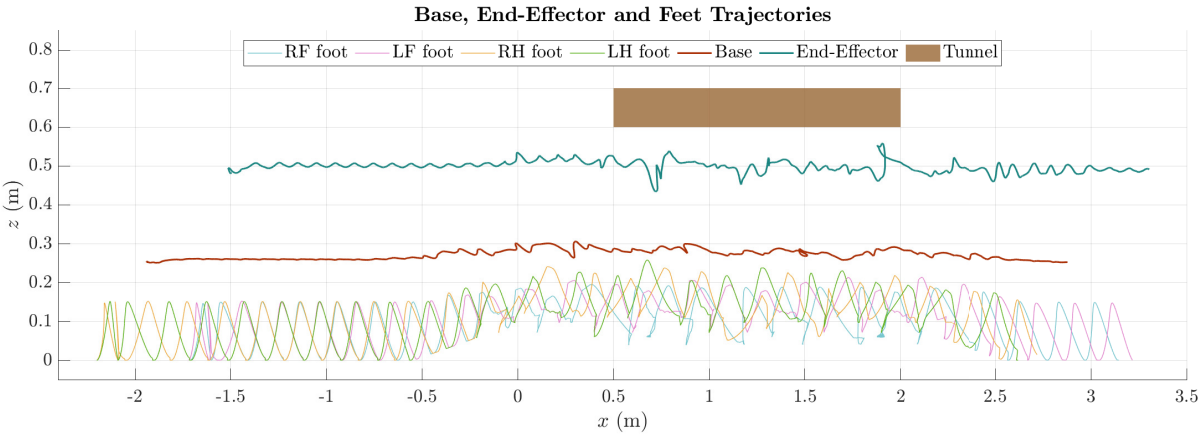
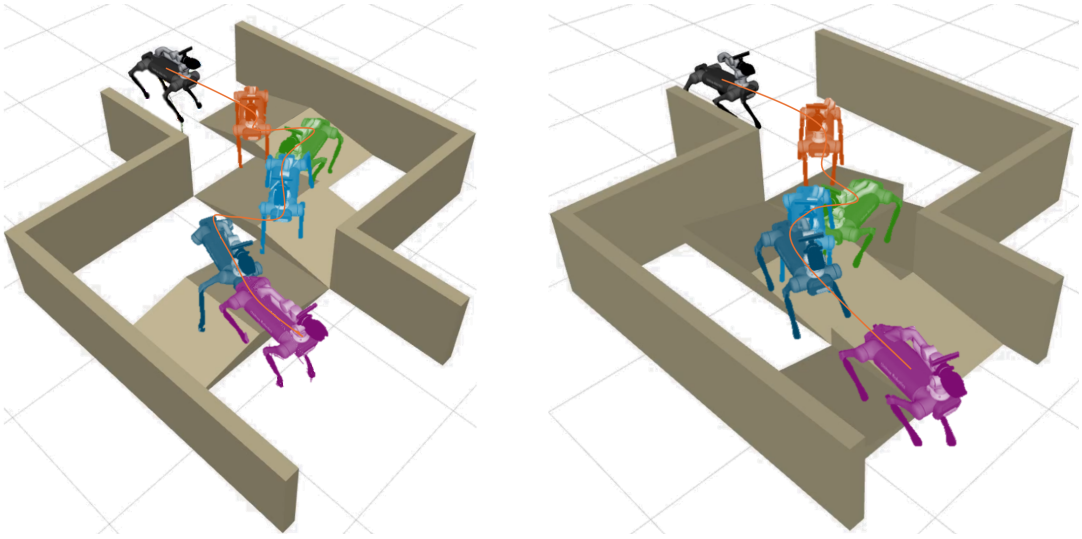


Figure 5.11. Base, end-effector and feet trajectories during uneven terrain tunnel traversal.

In the figure it can be observed how the end effector does not collide with the tunnel ceiling and how the controller manages to satisfy the CoM constraint successfully while traversing the uneven terrain. Looking at the feet trajectories, it is observed they are highly irregular due to multiple slips during the robot's motion, but still the robot manages to reach the end of the tunnel with good stability. T, highlighting the controller's effectiveness. This result demonstrates the quadruped's ability to navigate a confined and irregular environment using the proposed NMPC system.

5.1.2.5. Maze Traversal

Finally, the maze traversal test is based on four simple, modularly-designed mazes, each with different block configurations but the same overall shape. These mazes, constructed by another laboratory member, present a controlled yet challenging environment for the robot. The purpose of this test is to evaluate the NMPC controller's effectiveness in navigating through complex, constrained paths in different directions. By analyzing the performance in the four configurations, the robustness and adaptability of the controller can be better understood. Figure 5.12 shows the maze traversal for the first and fourth environments.



(a) **(b)**

Figure 5.12. Mazes traversal: (a) First maze, (b) Fourth maze.

The robot was guided through command velocities to traverse each of the four mazes multiple times, and the controller demonstrated sufficient robustness, resulting in no falls across all experiments. It was observed that when instabilities occurred, the NMPC quickly corrected the base orientation to prevent a fall, similar to its performance in other tests.

5.1.3. Force Compliance

As it was commented before, for quadruped manipulators, force compliance is crucial to ensure stable interaction with the environment. In the initial version of the NMPC controller, when the robot touched a surface with its end-effector, it had no mechanism to respond with appropriate force to maintain its original trajectory. This often resulted in crashes or destabilization during tasks that required physical contact, such as pushing doors or manipulating objects. To address this, the NMPC was modified to incorporate a force constraint into the optimization problem. As described in the previous chapter, this force constraint was also integrated as a task in the WBC controller, enabling the robot to respond with a controlled force when the end-effector makes contact with an object.

To validate this force compliance capability, a test was designed to simulate a scenario where continuous forces are applied to the end-effector in three sequential directions. The goal of this test was to evaluate the system's ability to maintain stable control and respond to these external forces while maintaining its base position and orientation. Forces of 20 N were applied in each principal direction during 3 seconds, as shown in Figure 5.13.

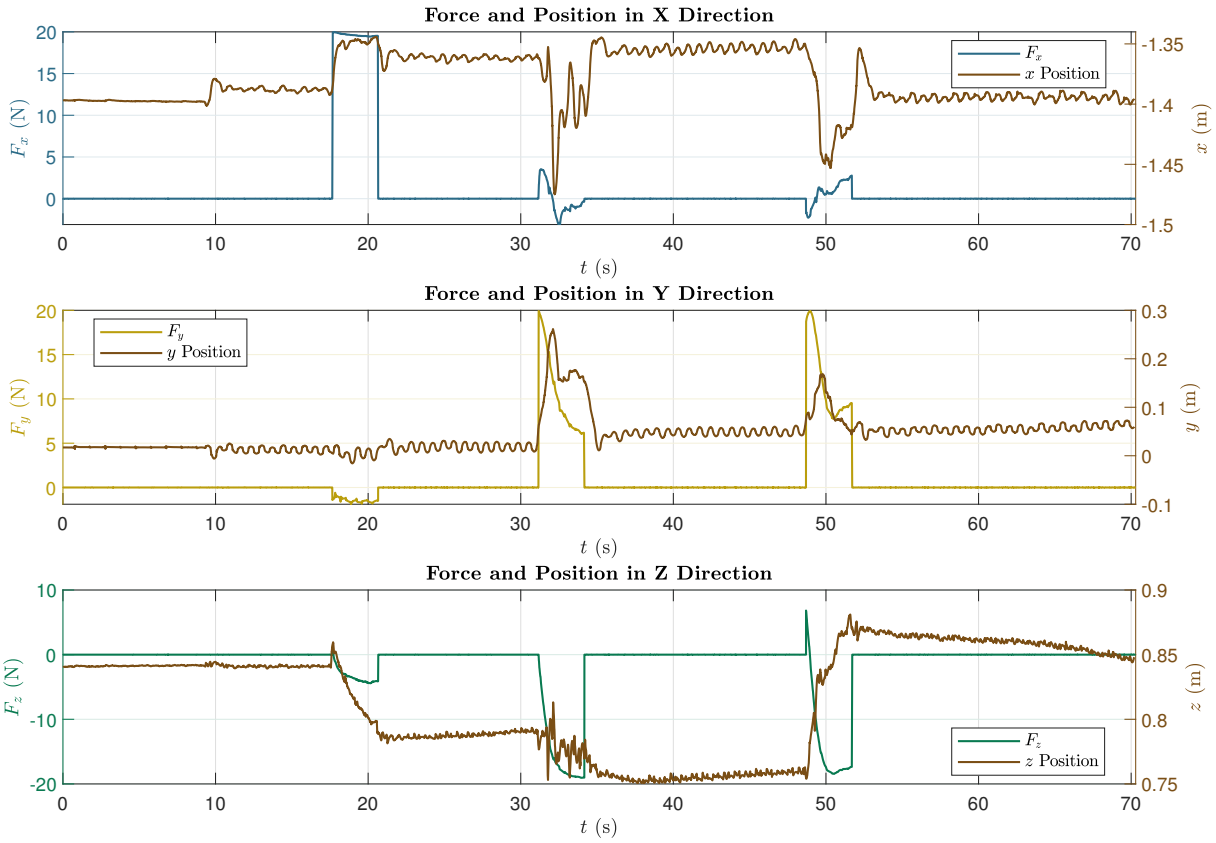


Figure 5.13. End-effector forces and positions evolution.

In the figure, it can be observed that the first applied force pulse corresponds to the x-axis component, while the second and third force pulses show contributions from both the y and z components. This occurs because the initial x-axis force pulse slightly alters

the orientation of the end-effector, causing subsequent force pulses to result in forces being distributed across multiple axes. The 20 N forces applied with respect to the world frame are thus measured as components across different axes by the end-effector force sensor. The robot successfully responds by quickly correcting its end-effector position in response to each force disturbance. However, a small residual error remains after stabilization, which could be attributed to the tuning of the $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{K}_p$, and $\boldsymbol{K}_d$ parameters. These parameters directly influence the control system's responsiveness and the ability to maintain precise tracking after disturbances.

5.1.4. Door Opening

The ability to autonomously open doors is a key feature for the quadruped robot, particularly in SAR scenarios where access to obstructed or confined spaces is critical. The door-opening tests focused on evaluating the system's ability to first align itself with the door and then execute the necessary maneuvers to successfully open it. The process was divided into two main phases: reorientation and positioning, followed by the actual door-opening procedure. Both phases were conducted using the NMPC and vision systems to guide the robot in accurately approaching and manipulating the door.

5.1.4.1. Reorientation and Positioning

In this test, the robot was placed in a position and orientation that was not aligned with the door, simulating a realistic scenario where precise initial positioning is not guaranteed, as shown in Figure 5.14. Several initial positions were tested, for instance, the figure shows a 30° rotated initial position 1 m away from the door center normal line.

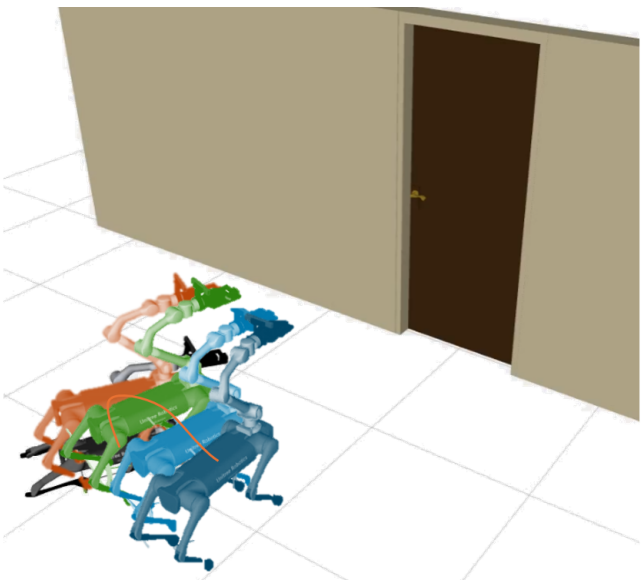


Figure 5.14. Quadruped manipulator repositioning maneuver.

In this case, using the repositioning planner algorithm described in the previous chapter, and the NMPC controller, relating the end-effector target trajectories with the ones of the base, the robot was able to reorient itself to achieve proper alignment with the door frame. This test was **assessed qualitatively** since it is not critical for the system to function properly as long as the robot is more or less normal to the door.

The algorithm enables to modify the normal distance to the door at which the door opening operation is desired to be started. During the test, it was observed that due to the nature of the equations, both higher inclination angles with respect to the door wall, and higher distances led to bigger positioning errors, especially for Δy_{ref}, since the noise of the measurements used for Δx_{ref} and ψ calculations were minimized thanks to the RANSAC plane algorithm.

It must be noted that in this work, the absence of obstacles in this repositioning task has been assumed. For a more generic application, this reorientation task would be performed by the navigation algorithm of the robot.

5.1.4.2. Opening Process

The door opening tests were conducted with a 5 kg push-type door, measuring 0.85 m in width and 2 m in height. This weight represents the lower end of the spectrum for interior doors, which typically weigh between 10 and 20 kg or more, depending on material and size. A lighter door was chosen for testing to avoid system instability, as the current state of the gripper controller is not fully integrated with the NMPC controller, making proper gripping more challenging for heavier doors. The door's hinge was modeled with low damping and friction coefficients (0.005 N·m·s and 0.0001, respectively). The handle was assigned a 0.3 kg mass, similar damping and friction coefficients, and a low spring stiffness of 10 N/m. Figure 5.15 illustrates the phases of the door opening process.

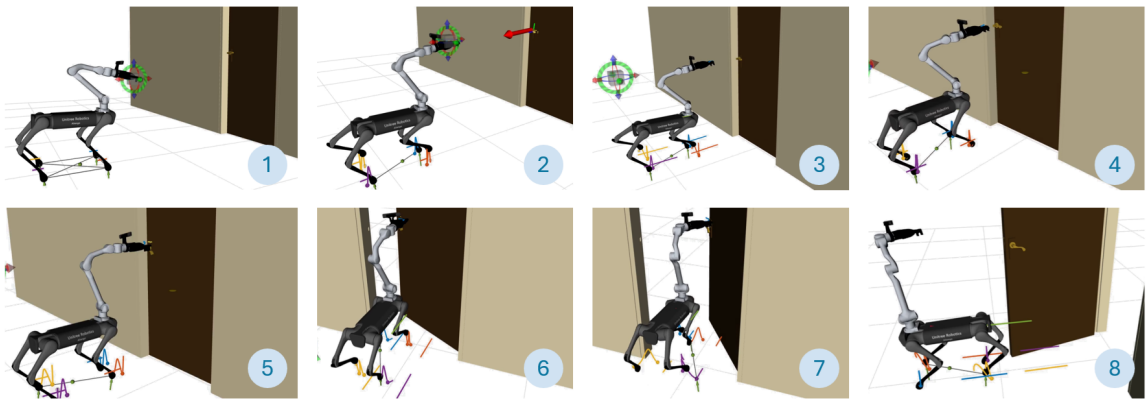


Figure 5.15. Push-door opening maneuver.

As shown in Figure 5.15, several phases can be identified. In the first phase, the robot successfully went to the initial position and got the visual information required for the

handle approximation phase. Next, the target position was set to a coordinate close to the estimated handle centroid with certain offsets to prevent from colliding. Then, the handle was lowered and the door was slightly pushed. Finally, the base moved towards the center of the door thanks to the imposed constraints and the manipulator accompanied the door rotation movement.

Figure 5.16 shows the plane positions followed by the quadruped manipulator's base CoM and its end-effector, respectively. It can be observed that at the end of the process the manipulator does not seem to be able to come back to its original position and this is thought to be due to the fact the NMPC controller together with the door exerted forces led the arm's third joint to singularity. This could be perhaps avoided in future research by imposing other constraints in the optimization problem. Additionally, it can be seen the base trajectory is not as smooth as the one of the end-effector. This could be perhaps solved by tuning the weights given in the optimization problem to each of the tracking tasks.

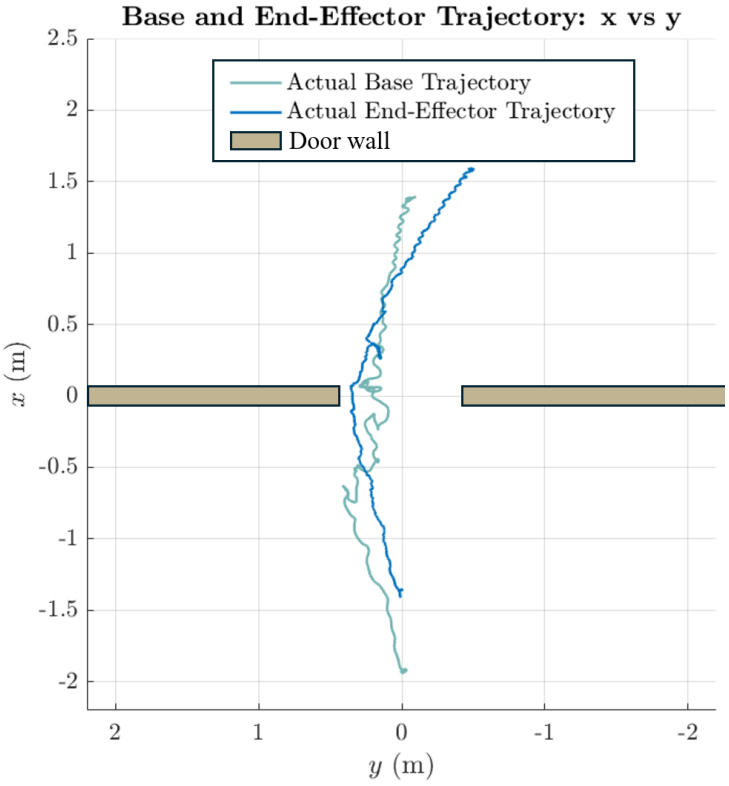


Figure 5.16. Base and end-effector plane positions during push-door opening maneuver.

In any case, though this was one of the successful tests, the system was seen in general to **lack robustness**. The door opening process was not always completed due to sudden instabilities when manipulating the door. Probably these issues could be solved with further parameter tuning and highly improved in case the door model was added into the NMPC optimization problem. This would imply the found opening solutions could be much more optimized than the manually-planned trajectories presented in this work.

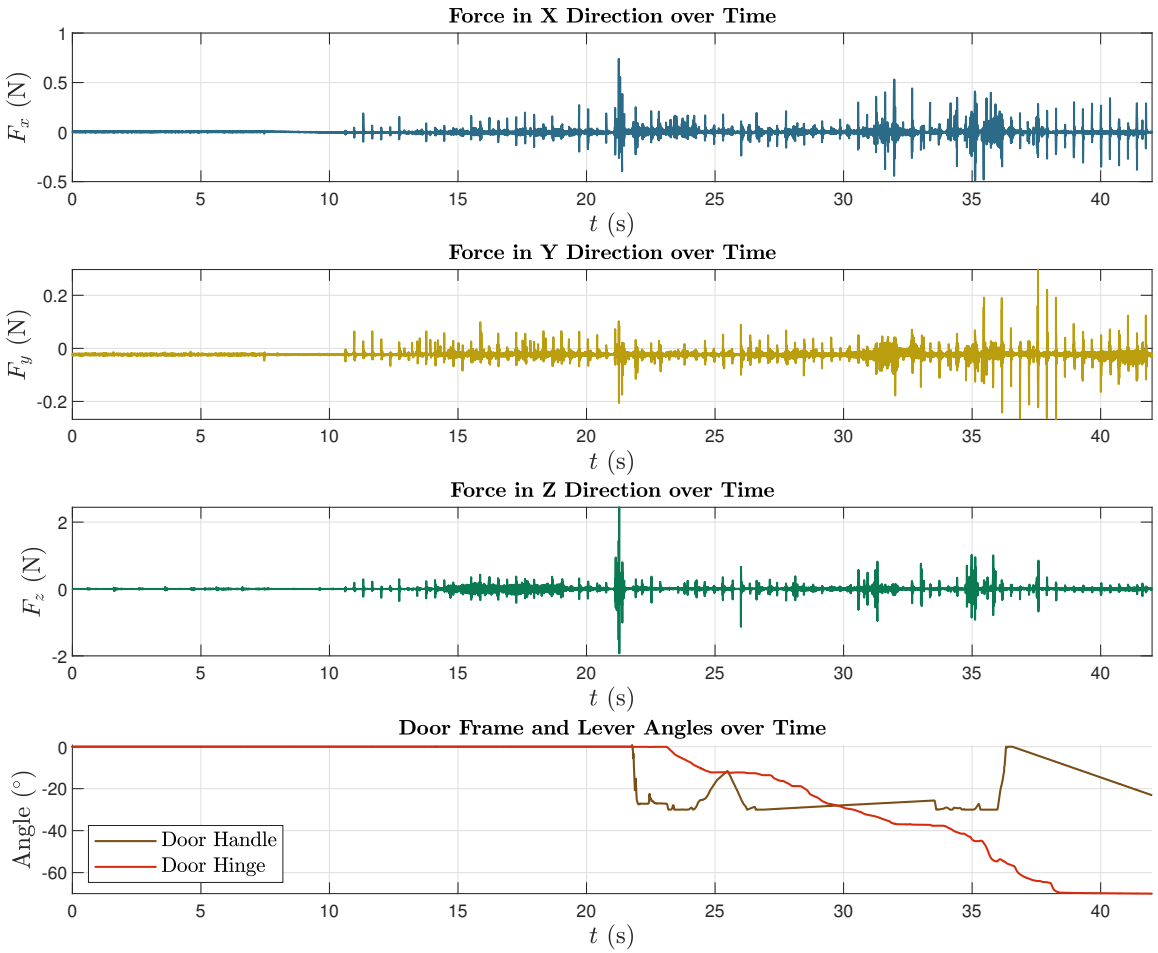


Figure 5.17. End-effector forces and door position evolution during door opening task.

Finally, Figure 5.17 shows the **forces measured on the end-effector** together with the evolution of the two joints of the door, namely the hinge and the lever. It can be observed how a peak vertical force is measured when the lever is lowered for enabling the door opening. The final rotation angle of the door was fixed in the algorithm to 70° which almost coincides with the final door hinge value. The x and y force components are seen to have highly variable values along the whole door opening process. This is thought to be due to the nature of the small continuous changes of the end-effector position. With respect to the spring door hinge, it behaves abnormally towards the end of the simulation. No explanation has been found.

5.2. Real Testing

The real-world testing phase aimed to validate the system's performance under practical conditions, extending beyond the controlled environment of simulations. This section provides an overview of the vision system's performance in real-world scenarios, an in-depth analysis of the results from implementing the NMPC on the quadruped robot

(without the robotic arm), and initial steps toward hardware integration of the complete system. These tests were instrumental in identifying both the system's strengths and areas for improvement, ensuring the final implementation meets the intended operational requirements for real-world applications.

5.2.1. Vision Module Testing

The real-world testing of the vision module assessed its ability to detect doors, handles, and estimate the rotation axis. Results showed both strengths and challenges.

The handle mask generation algorithm performed well under most conditions due to the robustness of the trained model. However, the estimation of the door radius **struggled in complex scenarios**. In some cases, vertical lines on the door misled the system, causing incorrect rotation axis detection, as seen in Figure 5.18. Additionally, minimal contrast between the door and frame resulted in axis detection errors, where the frame was identified instead of the door.

Figure 5.18. Door rotation axis detection errors.

With respect to the depth processing, the system **excelled in point cloud generation** for the handle at the SR305 camera's operating range. The handle's centroid and orientation were accurately captured, as shown in Figure 5.19. It can be observed how the hinge plate that threads the mechanism to the door though included in the mask, is correctly removed from the point cloud thanks to the RANSAC algorithm.

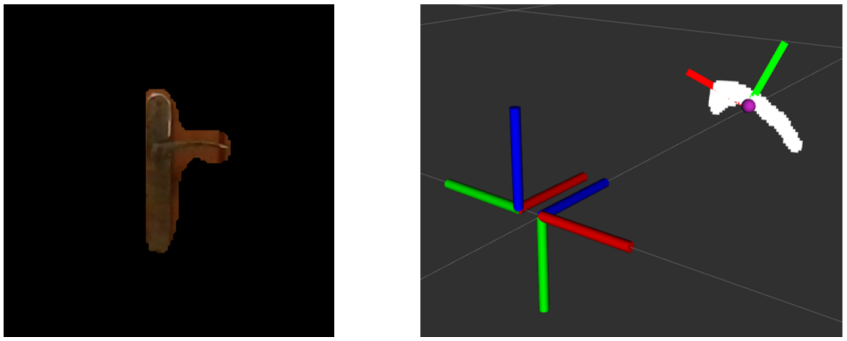


Figure 5.19. Real door handle generated point cloud.

5.2.2. NMPC Quadruped Control

Before integrating the NMPC with the full robotic system, initial tests were conducted using **only the quadruped**. This approach allowed for a focused evaluation of the NMPC controller's performance in managing the complex dynamics of the quadruped robot without the additional challenges introduced by the robotic arm. The controller for non-manipulator quadrupeds from [93], on which the manipulator quadruped repository from [95] was based, was employed to perform the tests.

The primary objective of these initial tests was to validate the NMPC controller's ability to maintain stability in various unstructured scenarios. The controller was responsible for optimizing the robot's body posture while navigating through different obstacles. The chosen environment for its availability within the lab was a set of obstacles built in the form of 40×40 cm modules with different types of terrains, as it can be seen in Figure 5.20.

Figure 5.20. Obstacle modules used during testing.

The conducted test consisted on traversing a 1.20 m long obstacle path both forward and backward. In this test, the NMPC controller was not installed on the robot platform itself; instead, the computer acted as the master, communicating all the necessary information to the robot over Wi-Fi, with the robot functioning as the slave. Though the obstacles are not challenging enough for considering them an orange area according to NIST classification, they enabled to extract key insights from the real-world controller functioning and validate the control system for low-complexity real-world unstructured environments.

The results of the real-world controller functioning demonstrate its effectiveness in low-complexity unstructured environments. The robot's gait was set to trot for all tests, with maximum linear and angular velocity references of 0.13 m/s and 0.15 rad/s, respectively. The CoM height was maintained at 0.4 m. The end position commands were issued via RViz, and for safety, the robot was manually supported by a textile handle, as depicted in Figure 5.21b. Despite this precaution, no external force was required to correct the robot's behavior during the experiments.

(a) (b)

Figure 5.21. AlienGo hardware testing: (a) Followed path, (b) Safety textile handle.

The velocity commands were issued solely in the x direction, both **forward and backwar**d, and for yaw rotation, as illustrated in Figure 5.22.

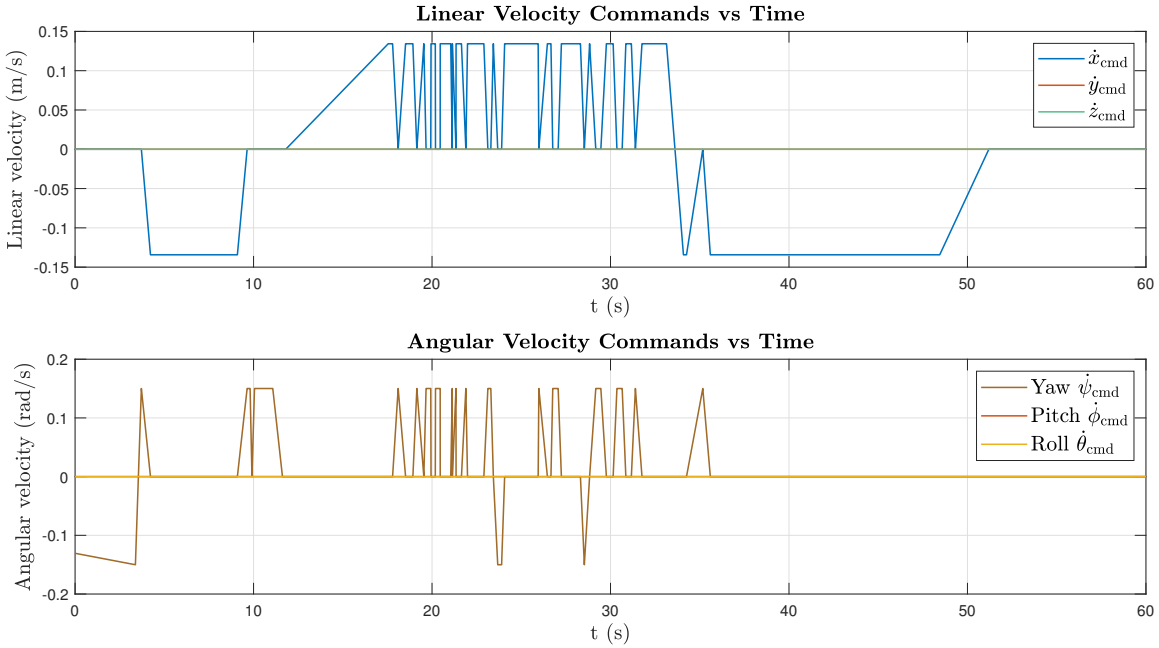


Figure 5.22. Comparison between velocity commands and velocities filtered estimations.

During these tests, the NMPC effectively predicted the robot's future states and adjusted control inputs accordingly, ensuring smooth and stable locomotion, as shown in Figure 5.23. The figure highlights how the desired base trajectories were closely followed by the controller, even while crossing the obstacle path.

The RMSE values obtained between during the 60 s test were 4.41 cm for the x component, 4.24 cm for the y component, and 2.81 cm for the z component. These low RMSE values indicate a high level of accuracy in trajectory tracking, confirming the controller's reliability and precision in maintaining the desired trajectory within the tested scenario. These results suggest that the controller is well-suited for more advanced and

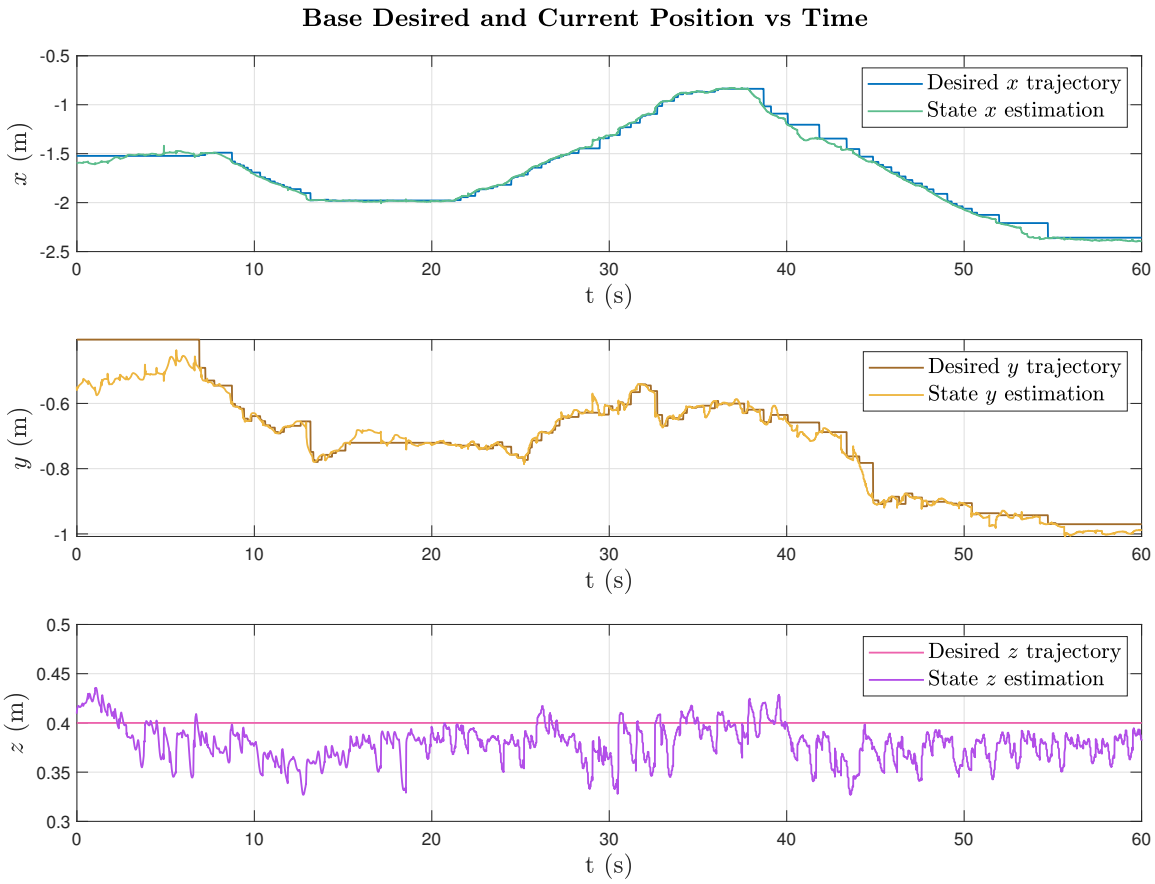


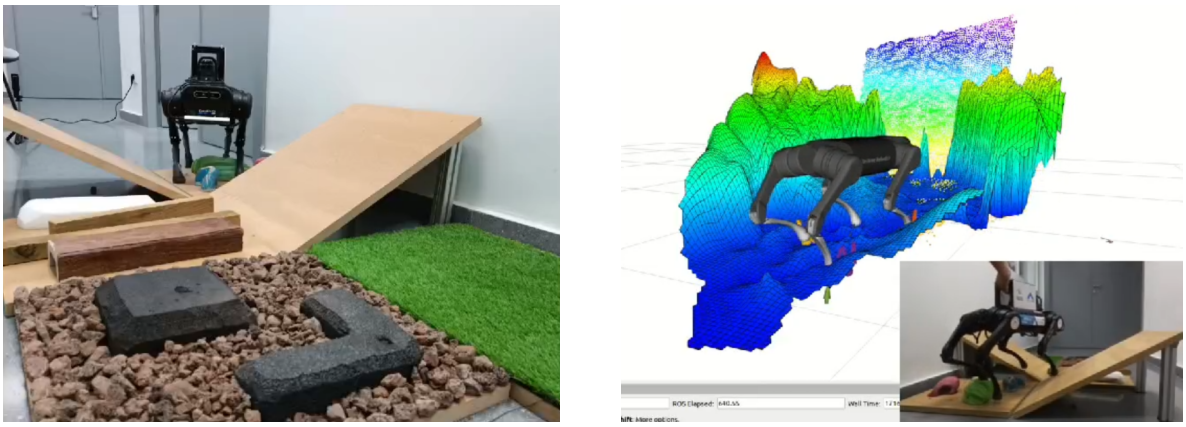
Figure 5.23. Comparison between velocity commands and velocities filtered estimations.

complex testing, pushing towards its physical limits, since the obstacles in this case were relatively low with a maximum height of around 10 cm. Actually, this same obstacles path was tested and also overcome by the default AlienGo controller, so it does not have enough complexity to showcase the advanced capabilities of the developed NMPC controller versus the gait pattern generator one.

Although not critically impairing, the computational demands of the NMPC posed challenges when operating the system in a master-slave configuration from an external computer. The MPC control loop frequency was slightly lower than desired, around 90 Hz, which may have contributed to reduced stability. Achieving more stable performance likely requires a higher control frequency. This underscores the importance of implementing the controller directly on the onboard Jetson TX2 in future work to ensure real-time performance and enhance overall system stability.

The NMPC controller was additionally tested as part of an unrelated parallel research project conducted by other members of the lab on the ARTU-R Unitree A1 quadruped platform [129], which focused on environment reconstruction through perception sensor integration. The NMPC controller was qualitatively assessed to perform well, enabling the robot to traverse obstacles without falling—an improvement over the previous CPG

controller, which frequently led to falls.



(a) (b)

Figure 5.24. Unitree A1 quadruped NMPC controller testing [129]: (a) Obstacle track, (b) Elevation map generation.

5.2.3. Steps Towards Testing the Quadruped Manipulator

Due to time constraints, the testing and validation of the entire quadruped manipulator system could not be completed, although the hardware integration was successfully finalized. The software was adapted satisfactorily from simulation for real hardware implementation using Unitree's SDK. However, the limited availability of the Z1 arm, which was being used in parallel for an agricultural robotics project, combined with the tight schedule, made it unfeasible to complete this task.

Additionally, one of the last challenges encountered during the software integration phase was related to the communication between the computer master and the Z1 arm, specifically involving the User Datagram Protocol (UDP) protocol. The issue arose because the UDP address of the arm was not being correctly detected, which led to difficulties in establishing its connection.

It is likely that this issue could be resolved by correcting some protocol addresses or by running the program directly on the onboard Jetson TX2 computer. Localizing the control loop on the same hardware platform could additionally improve the overall stability and performance of the system. Unfortunately, due to the aforementioned time constraint and hardware availability, this solution could not be fully implemented and tested. However, addressing this issue is recommended for future work to ensure reliable operation of the quadruped manipulator system.

Chapter 6

Conclusions and Future Work

The final chapter summarizes the key findings and accomplishments of the project. It reflects on the project's outcomes and the overall success in meeting the initial objectives. Furthermore, this chapter outlines potential future work and areas for further research and development, aimed at enhancing the robot's capabilities.

6.1. Main Conclusions

The conclusions drawn from this project highlight the development and implementation of a quadruped manipulator system with a focus on NMPC for handling locomotion and manipulation tasks. Through extensive testing, mainly in simulation but also in real-world environments, several key findings have emerged.

First, the quadruped's locomotion system, driven by the NMPC controller, exhibited **strong performance across various unstructured environments**, including pallet obstacles, unstable platforms, and complex terrains such as stairs and ramps. The controller demonstrated **effective trajectory tracking**, maintaining stability even when confronted with uneven surfaces and sudden perturbations. The **low RMSE** values observed in these tests confirm the accuracy of the system's performance, and the reactive control behavior ensures recovery from destabilizing conditions. However, **challenges remain** in environments like **stair descents**, where leg extension and the absence of **perception-based foot placement** optimization led to failures in simulation. The future integration of perception into the NMPC model will likely mitigate these issues and enhance overall stability.

The vision system also delivered promising results, particularly in detecting doors

and door handles. The system successfully handled both segmentation and recognition tasks, enabling the robot to approach and manipulate doors in simulated environments. While the full hardware integration fo the quadruped and manipulator was completed, time constraints prevented the real-world testing of the integrated system. Despite this, simulation results suggest that the manipulator system is capable of performing push door-opening tasks with a reasonable degree of precision, offering a solid foundation for future tests.

Real-world testing of the quadruped without the manipulator provided valuable insights into the system's practical capabilities. The NMPC controller successfully managed the robot's movements in low-complexity unstructured environments, such as low-height obstacle paths, proving that the system is robust enough to handle more challenging scenarios. The use of an external computer to control the NMPC loop, however, revealed the limitations of this setup in terms of control frequency and overall system stability. To overcome this, it is recommended that future iterations run the NMPC controller directly on the onboard computer to ensure real-time performance and improve the system's responsiveness.

While the manipulation tasks were not fully tested due to hardware limitations, the project has laid a strong foundation for further development. The communication issues encountered during the integration phase, particularly those related to the UDP protocol for the manipulator, suggest that future work should focus on resolving these connectivity problems. Running the entire control loop on the onboard Jetson TX2 is expected ot address these challenges and optimize the overall system's performance.

In summary, this project demonstrates the quadruped manipulator system's **potential to perform complex locomotion and manipulation tasks** in unstructured environments for SAR operations. The successful integration of the NMPC controller and vision system provides a strong framework for future work. The system effectively maintained stability and executed precise maneuvers in challenging terrains, with the vision algorithms supporting detection and manipulation for door-opening tasks. Overall, the results validate the system's robustness and readiness for more advanced real-world testing.

6.2. Future Work Lines

This project has opened up several avenues for future research and development, ranging from short-term improvements to more complex, long-term advancements. These future work lines can be classified into distinct categories that address different aspects of the quadruped manipulator system. Some tasks, such as full integration and real-

time control optimization, are achievable in the near future, while others, like enhancing autonomy and manipulation capabilities, will require more extensive research efforts.

6.2.1. Integration and Real-World Optimization

- **Complete testing of the fully integrated quadruped manipulator:** The full hardware and software integration of the quadruped manipulator system needs to be tested in real-world conditions. This includes resolving communication issues between the robotic arm and the master control system, and testing the NMPC and vision system in conjunction with real-world door-opening tasks.
- **Onboard System Optimization:** Running the control loop and vision processing directly on the robot's onboard computer will likely enhance performance and real-time responsiveness, reducing latency and improving stability.
- **Test interaction with complex real-world objects:** In addition to door-opening tasks, the system should be tested with SAR-specific objects like debris, rubble, or collapsed structures to enhance the robot's manipulation capabilities in disaster scenarios.
- **Field deployment in SAR simulations:** After lab-based real-world testing, it would be interesting to deploy the robot in large-scale SAR training exercises, where it can be tested in realistic disaster scenarios such as earthquake simulations, collapsed structures, or flooded areas to validate its operational readiness.
- **Integration with multi-robot SAR teams:** Explore the potential for integrating the quadruped manipulator with other robots (aerial or ground) for coordinated multi-robot SAR missions, enabling better coverage and faster victim detection and assistance.

6.2.2. Perception and Control Enhancements

- **Incorporation of perception-based foot placement:** Implementing perception-based algorithms to improve foot placement on uneven terrain could significantly reduce the risk of falls in challenging environments, especially when navigating stairs, ramps, or high randomly distributed pallets.
- **Integration with room navigation for active exploration pipeline:** By combining the door-opening capabilities from this project with the navigation framework for victim active exploration that enables to avoid obstacles, the robot would be able to explore more rooms in complex environments. This would improve its effectiveness in SAR missions.

- **Adaptive control for dynamic environments:** Enhance the quadruped's control system to allow for dynamic adjustment of gait parameters, such as step height and stride length, in real-time based on terrain feedback. For example, as the robot encounters obstacles or varying terrain types like rubble, uneven ground, or debris, the system can automatically increase or decrease the step height to maintain stability and prevent tripping. This dynamic control can be particularly beneficial in SAR scenarios where the terrain is unpredictable and constantly changing, improving the robot's ability to traverse complex environments without falling or losing balance.

- **Door model NMPC integration:** Integrating the door model directly into the NMPC optimization problem could significantly improve the robot's door-opening capabilities by accounting for the dynamics of the door during manipulation. This would allow the NMPC to generate more accurate trajectories for the manipulation and base, addressing current issues in the planning phase, such as collisions or suboptimal movements.

  Additionally, incorporating the door model into the optimization process would not only enhance real-time control but also provide an **excellent framework for generating realistic door-opening trajectories**. These trajectories could be used as training data for RL policies, enabling the robot to learn more adaptive strategies for door manipulation in a variety of scenarios.

- **Door hinge class identification:** Enhancing the object detection You Only Look Once (YOLO) model by incorporating door hinges as an additional class would allow the system to automatically determine whether a door is a pull-type (with visible hinges) or a push-type (with hidden hinges). This classification would enable the robot to adapt its manipulation strategy accordingly, triggering the appropriate control actions for the specific door type.

- **Perception system robustness enhancement:** Ensure all vision and depth sensors are calibrated for challenging conditions like smoke, dust, or darkness, which are common in disaster scenarios, ensuring reliable locomotion performance in low-visibility environments.

- **Develop real-time safety features:** Implement emergency stop features, obstacle avoidance systems, and safe interaction protocols to ensure safe operation in hazardous environments protecting both the robot and any trapped individuals it encounters.

6.2.3. Advanced Manipulation and Autonomy

- **Improvement of door manipulation abilities:** Future work should extend the door-opening capabilities to include pull-type doors and more complex door mechanisms, such as knobs, enhancing the robot's manipulation repertoire in real-world environments.
- **Robust grasping algorithms:** Future work will include the integration of an independent controller for the gripper within the NMPC combined robotic system. Refining the grasping capabilities of the Z1 manipulator by integrating advanced algorithms which consider force feedback could improve performance in handling different objects. For example, a door of unknown type could be tried to open first as if it was a push-door type, and if a metric relating the measured torques on the arm overpass a certain threshold, then it could try the opposite direction. This would enable to avoid damaging both the robot and the door with which the interaction is performed.

Bibliography

[1] J. F. García-Samartín, C. C. Ulloa, J. del Cerro, and A. Barrientos, "Active robotic search for victims using ensemble deep learning techniques", *Machine Learning: Science and Technology*, vol. 5, no. 2, p. 025 004, Apr. 2024. DOI: 10.1088/2632-2153/ad33df.

[2] C. C. Ulloa, D. Domínguez, A. Barrientos, and J. del Cerro, "Design and Mixed-Reality Teleoperation of a Quadruped-Manipulator Robot for SAR Tasks", in *Robotics in Natural Settings*, Cham: Springer International Publishing, 2023, pp. 181–194. DOI: 10.1007/978-3-031-15226-9_19.

[3] C. C. Ulloa, D. D. J. del Cerro, and A. Barrientos, "A Mixed-Reality Tele-Operation Method for High-Level Control of a Legged-Manipulator Robot", *Sensors*, vol. 22, no. 21, 2022, ISSN: 1424-8220. DOI: 10.3390/s22218146.

[4] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A Unified MPC Framework for Whole-Body Dynamic Locomotion and Manipulation", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021. DOI: 10.1109/LRA.2021.3068908.

[5] H. Chai, Y. Li, R. Song, G. Zhang, Q. Zhang, S. Liu, J. Hou, Y. Xin, M. Yuan, G. Zhang, and Z. Yang, "A survey of the development of quadruped robots: Joint configuration, dynamic locomotion control method and mobile manipulation approach", *Biomimetic Intelligence and Robotics*, vol. 2, no. 1, p. 100 029, 2022, ISSN: 2667-3797. DOI: https://doi.org/10.1016/j.birob.2021.100029.

[6] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots", *Science Robotics*, vol. 4, no. 26, Jan. 2019, ISSN: 2470-9476. DOI: 10.1126/scirobotics.aau5872.

[7] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, *Sim-to-Real: Learning Agile Locomotion For Quadruped Robots*, 2018. DOI: 10.15607/RSS.2018.XIV.010.

[8] *NFPA 1670: Standard on Operations and Training for Technical Search and Rescue Incidents*, Quincy, MA: National Fire Protection Association (NFPA), 2017.

[9] A. L. Adams and others, "Search Is a Time-Critical Event: When Search and Rescue Missions May Become Futile", *Wilderness & Environmental Medicine*, vol. 18, no. 2, pp. 95–101, 2007, ISSN: 1080-6032. DOI: 10.1580/06-WEME-OR-035R1.1.

[10] H. Haq, *Three survivors pulled alive from earthquake rubble in turkey, more than 248 hours after quake*, Feb. 2023. [Online]. Available: https://www.cnn.com/2023/02/16/europe/turkey-syria-earthquake-rescue-efforts-intl/index.html (visited on 12/08/2024).

[11] R. R. Murphy, S. Tadokoro, and A. Kleiner, "Disaster Robotics", in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., Cham: Springer International Publishing, 2016, pp. 1577–1604, ISBN: 978-3-319-32552-1.

[12] R. Murphy, J. Casper, J. Hyams, M. Micire, and B. Minten, "Mobility and Sensing Demands in USAR", in *2000 26th Annual Conference of the IEEE Industrial Electronics Society. IECON 2000. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation. 21st Century Technologies*, Nagoya, Japan: IEEE, 2000.

[13] M. R. Blackburn, H. R. Everett, and R. T. Laird, "After Action Report to the Joint-Program Office: Center for the Robotic Assisted Search and Rescue (CRASAR) Related Efforts at the World Trade Center", Space and Naval Warfare Systems Center San Diego CA, Technical Report, 2002.

[14] R. Eguchi, K. Elwood, E. K. Lee, and M. Greene, "The 2010 Canterbury and 2011 Christchurch New Zealand Earthquakes and the 2011 Tohoku Japan Earthquake", Earthquake Engineering Research Institute, Technical Report, 2012.

[15] J. Whitman, N. Zevallos, M. Travers, and H. Choset, "Snake Robot Urban Search After the 2017 Mexico City Earthquake", in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2018, pp. 1–6. DOI: 10.1109/SSRR.2018.8468633.

[16] C. C. Ulloa, "Quadrupedal Robots in Search and Rescue : Perception and Teleoperation", Mar. 2024. DOI: 10.20868/UPM.thesis.81769.

[17] A. Jacoff, E. Messina, B. A. Weiss, S. Tadokoro, and Y. Nakagawa, "Test Arenas and Performance Metrics for Urban Search and Rescue Robots", in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, vol. 4, IEEE, 2003, pp. 3396–3403.

[18] NIST, *National institute of standards and technology*, 2022. [Online]. Available: https://www.nist.gov/ (visited on 12/08/2024).

[19] J. Scholtz, B. Antonishek, and J. Young, "Evaluation of Human-Robot Interaction in the NIST Reference Search and Rescue Test Arenas", National Institute of Standards and Technology, Gaithersburg, MD 20899, Tech. Rep., 2004.

[20] H. Chitikena, F. Sanfilippo, and S. Ma, "Robotics in Search and Rescue (SAR) Operations: An Ethical and Design Perspective Framework for Response Phase", *Applied Sciences*, vol. 13, no. 3, 2023, ISSN: 2076-3417. DOI: 10.3390/app13031800.

[21] D. Falanga, K. Kleber, S. Mintchev, D. Floreano, and D. Scaramuzza, "The Foldable Drone: A Morphing Quadrotor That Can Squeeze and Fly", *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 209–216, 2019. DOI: 10.1109/LRA.2018.2885575.

[22] J. Dufek and R. Murphy, "Visual Pose Estimation of Rescue Unmanned Surface Vehicle From Unmanned Aerial System", *Frontiers in Robotics and AI*, vol. 6, 2019, ISSN: 2296-9144. DOI: 10.3389/frobt.2019.00042.

[23] E. Kelasidi, P. Liljebäck, K. Pettersen, and J. Gravdahl, "Experimental investigation of efficient locomotion of underwater snake robots for lateral undulation and eel-like motion patterns", *Robotics and Biomimetics*, vol. 2, Dec. 2015. DOI: 10.1186/s40638-015-0029-4.

[24] R. Murphy, "Human-robot interaction in rescue robotics", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 2, pp. 138–153, 2004. DOI: 10.1109/TSMCC.2004.826267.

[25] J. P. Queralta and others, "Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision", *IEEE Access*, vol. 8, pp. 191 617–191 643, 2020. DOI: 10.1109/ACCESS.2020.3030190.

[26] W.-Y. G. Louie and G. Nejat, "A Victim Identification Methodology for Rescue Robots Operating in Cluttered USAR Environments", *Advanced Robotics*, vol. 27, no. 5, pp. 373–384, 2013. DOI: 10.1080/01691864.2013.763743.

[27] J. A. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, A. J. Ijspeert, D. Floreano, L. M. Gambardella, R. Y. Siegwart, and D. Scaramuzza, "The current state and future outlook of rescue robotics", *Journal of Field Robotics*, vol. 36, pp. 1171–1191, 2019. DOI: 10.1002/rob.21887.

[28] J. Roldán, P. García, M. Garzon, J. De León, J. del Cerro, and A. Barrientos, "Heterogeneous Multi-Robot System for Mapping Environmental Variables of Greenhouses", *Sensors*, vol. 16, p. 1018, Jul. 2016. DOI: 10.3390/s16071018.

[29] B. Yamauchi, "PackBot: A Versatile Platform for Military Robotics", *Proc. SPIE*, vol. 5422, Sep. 2004. DOI: 10.1117/12.538328.

[30] J. Tordesillas, J. del Cerro, A. Barrientos, and J. De León, "Modelo cinemático de un robot hexápodo con C-legs", Sep. 2016. DOI: 10.17979/spudc.9788497498081.0352.

[31] M. H. Raibert and E. R. Tello, "Legged robots that balance", *IEEE Expert*, vol. 1, no. 4, pp. 89–89, 1986. DOI: 10.1109/MEX.1986.4307016.

[32] R. Mosher, "Test and evaluation of a versatile walking truck", *Proceedings of Off-Road Mobility Research Symposium, Washington DC, 1968*, pp. 359–379, 1968. [Online]. Available: https://cir.nii.ac.jp/crid/1570291225044499840.

[33] "BigDog, the Rough-Terrain Quadruped Robot", *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 822–10 825, 2008, 17th IFAC World Congress, ISSN: 1474-6670. DOI: 10.3182/20080706-5-KR-1001.01833.

[34] A. Bouman, M. Ginting, N. Alatur, M. Palieri, D. Fan, T. Touma, T. Pailevanian, S.-K. Kim, K. Otsu, J. Burdick, and A.-A. Agha-Mohammadi, "Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion", Oct. 2020, pp. 2518–2525. DOI: 10.1109/IROS45743.2020.9341361.

[35] M. Hutter, C. Gehring, A. Lauber, F. Günther, D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Blösch, H. Kolvenbach, M. Bjelonic, L. Isler, and K. Meyer, "ANYmal - toward legged robots for harsh environments", *Advanced Robotics*, vol. 31, pp. 918–931, 2017. DOI: 10.1080/01691864.2017.1378591.

[36] D. Bellicoso, M. Bjelonic, L. Wellhausen, K. Holtmann, F. Günther, M. Tranzatto, P. Fankhauser, and M. Hutter, "Advances in real-world applications for legged robots", *Journal of Field Robotics*, pp. 1311–1326, Oct. 2018. DOI: 10.1002/rob.21839.

[37] C. G. Atkeson, P. W. B. Benzun, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. P. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, T. Padir, F. Polido, G. G. Tighe, and X. Xinjilefu, "What Happened at the DARPA Robotics Challenge Finals", in *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*. Cham: Springer International Publishing, 2018, pp. 667–684. DOI: 10.1007/978-3-319-74666-1_17.

[38] B. Lindqvist, S. Karlsson, A. Koval, I. Tevetzidis, J. Haluška, C. Kanellakis, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Multimodality robotic systems: Integrated combined legged-aerial mobility for subterranean search-and-rescue", *Robotics and Autonomous Systems*, vol. 154, p. 104 134, 2022, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2022.104134.

[39] S. Halder, K. Afsari, E. Chiou, R. Patrick, and K. A. Hamed, "Construction inspection & monitoring with quadruped robots in future human-robot teaming: A preliminary study", *Journal of Building Engineering*, vol. 65, p. 105 814, 2023, ISSN: 2352-7102. DOI: https://doi.org/10.1016/j.jobe.2022.105814.

[40] Y. Hu, L. Jiang, K. Kang, D. Cao, Z. Zhou, J. Liu, Y. Wang, M. Li, and H. Hu, "A Navigation and Control Framework of Quadrupedal Robot for Autonomous Exploration in Cave Environments", in Oct. 2023, pp. 49–58, ISBN: 978-981-99-6479-6. DOI: 10.1007/978-981-99-6480-2_4.

[41] X. Wang, J. Liu, T. Zhang, and J. Zhang, "Applied Research of Agricultural Quadru-ped Robots", Apr. 2024, pp. 954–957. DOI: 10.1109/CSNT60213.2024.10545756.

[42] S. Owoeye, F. Durodola, P. Adeniyi, I. Abdullahi, and B. Adesanya, "Design and development of a quadruped home surveillance robot", *IAES International Journal of Robotics and Automation (IJRA)*, vol. 13, p. 232, Jun. 2024. DOI: 10.11591/ijra.v13i2.pp232-246.

[43] Y. Gong, G. Sun, A. Nair, A. Bidwai, R. C. S., J. Grezmak, G. Sartoretti, and K. A. Daltorio, "Legged Robots for Object Manipulation: A Review", *Frontiers in Mechanical Engineering*, vol. 9, p. 1 142 421, 2023. DOI: 10.3389/fmech.2023.1142421.

[44] M. Lopes, "Quadruped manipulator for potential agricultural applications", Ph.D. dissertation, Jul. 2023. DOI: 10.13140/RG.2.2.14303.84643.

[45] Deep Robotics, *Jueying x20*, 2022. [Online]. Available: https://www.deeprobotics.cn/en/products.html (visited on 12/08/2024).

[46] Unitree Robotics, *Aliengo*, 2020. [Online]. Available: https://shop.unitree.com/products/aliengo (visited on 12/08/2024).

[47] Boston Dynamics, *Spot Arm Specifications and Key Concepts*, Mar. 2023. [Online]. Available: https://support.bostondynamics.com/s/article/Spot-Arm-specifications-and-concepts (visited on 12/08/2024).

[48] Unitree Robotics, *Unitree Robotics Aliengo + Z1 for Fire Rescue*, 2022. [Online]. Available: https://www.youtube.com/watch?v=7CVwGY65In8 (visited on 12/08/2024).

[49] J. Li, H. Gao, Y. Wan, J. Humphreys, C. Peers, H. Yu, and C. Zhou, "Whole-Body Control for a Torque-Controlled Legged Mobile Manipulator", *Actuators*, vol. 11, no. 11, 2022, ISSN: 2076-0825. DOI: 10.3390/act11110304. [Online]. Available: https://www.mdpi.com/2076-0825/11/11/304.

[50] Y. Ma, F. Farshidian, T. Miki, J. Lee, and M. Hutter, "Combining Learning-Based Locomotion Policy With Model-Based Manipulation for Legged Mobile Manipulators", *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2377–2384, 2022. DOI: 10.1109/LRA.2022.3143567.

[51] C. D. Bellicoso, K. Krämer, M. Stäuble, D. Sako, F. Jenelten, M. Bjelonic, and M. Hutter, "ALMA - Articulated Locomotion and Manipulation for a Torque-Controllable Robot", in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8477–8483. DOI: 10.1109/ICRA.2019.8794273.

[52] J. Guo, H. Chai, Y. Li, Q. Zhang, Z. Wang, J. Zhang, Q. Zhang, and H. Zhao, "Research on the Autonomous System of the Quadruped Robot with a Manipulator to Realize Leader-Following, Object Recognition, Navigation, and Operation", *CSEE Systems and Control Engineering*, vol. 4, no. 4, pp. 376–388, Dec. 2022. DOI: 10.1049/csy2.12069.

[53] S. Zimmermann, R. Poranne, and S. Coros, "Go Fetch! - Dynamic Grasps using Boston Dynamics Spot with External Robotic Arm", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 4488–4494. DOI: 10.1109/ICRA48506.2021.9561835.

[54] T. Sandakalum and M. H. Ang, "Motion Planning for Mobile Manipulators—A Systematic Review", *Machines*, vol. 10, no. 2, 2022, ISSN: 2075-1702. DOI: 10.3390/machines10020097. [Online]. Available: https://www.mdpi.com/2075-1702/10/2/97.

[55] C. C. Ulloa, L. Sánchez, J. D. Cerro, and A. Barrientos, "Deep Learning Vision System for Quadruped Robot Gait Pattern Regulation", *Biomimetics*, vol. 8, no. 3, 2023, ISSN: 2313-7673. DOI: 10.3390/biomimetics8030289.

[56] C. Dario Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization", in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 558–564. DOI: 10.1109/HUMANOIDS.2016.7803330.

[57] M. Vukobratović and B. Borovac, "Zero-moment point — Thirty five years of its life", *International Journal of Humanoid Robotics*, vol. 1, no. 01, pp. 157–173, 2004. DOI: 10.1142/S0219843604000083.

[58] I. Poulakakis, E. Papadopoulos, and M. Buehler, "On the Stability of the Passive Dynamics of Quadrupedal Running with a Bounding Gait", *International Journal of Robotics Research*, vol. 25, no. 7, pp. 669–687, 2006. DOI: 10.1177/0278364906066768.

[59] M. H. Raibert, H. B. B. Jr., M. Chepponis, J. Koechling, and J. K. Hodgins, "Dynamically Stable Legged Locomotion", Massachusetts Institute of Technology, Artificial Intelligence Lab, Tech. Rep., Sep. 1989, p. 207.

[60] J. Pratt, P. Dilworth, and G. Pratt, "Virtual Model Control of a Bipedal Walking Robot", in *Proceedings of the International Conference on Robotics and Automation*, vol. 1, IEEE, 1997, pp. 193–198. DOI: 10.1109/ROBOT.1997.620037.

[61] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant Quadruped Locomotion Over Rough Terrain", in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 814–820. DOI: 10.1109/IROS.2009.5354681.

[62] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2245–2252. DOI: 10.1109/IROS.2018.8593885.

[63] M. Neunert, M. Stauble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds", *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, Jul. 2018, ISSN: 2377-3774. DOI: 10.1109/lra.2018.2800124.

[64] G. Bledt, "Policy regularized model predictive control framework for robust legged locomotion", Jan. 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:196054916.

[65] F. Farshidian and others, *OCS2: An open source library for Optimal Control of Switched Systems*, GitHub repository, 2024. [Online]. Available: https://github.com/leggedrobotics/ocs2 (visited on 12/08/2024).

[66] M. Johansson, "Modeling and control of a crushing circuit for platinum concentration", Ph.D. dissertation, Jan. 2018. DOI: 10.13140/RG.2.2.31508.60805.

[67] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive Locomotion Through Nonlinear Model-Predictive Control", *Trans. Rob.*, vol. 39, no. 5, pp. 3402–3421, May 2023, ISSN: 1552-3098. DOI: 10.1109/TRO.2023.3275384.

[68] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation", in *Nonlinear Model Predictive Control: Towards New Challenging Applications*, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 391–417, ISBN: 978-3-642-01094-1. DOI: 10.1007/978-3-642-01094-1_32.

[69] D. Kim, J. Lee, J. Ahn, O. Campbell, H. Hwang, and L. Sentis, "Computationally-Robust and Efficient Prioritized Whole-Body Controller with Contact Constraints", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–8. DOI: 10.1109/IROS.2018.8593767.

[70] C. Dario Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots", in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada: IEEE Press, 2017, pp. 3359–3365. DOI: 10.1109/IROS.2017.8206174.

[71] A. J. Ijspeert, "Biorobotics: Using robots to emulate and investigate agile locomotion", *Science*, vol. 346, no. 6206, pp. 196–203, 2014. DOI: 10.1126/science.1254486.

[72] L. R. Palmer and D. E. Orin, "Intelligent Control of High-Speed Turning in a Quadruped", *Journal of Intelligent and Robotic Systems*, vol. 58, no. 1, pp. 47–68, 2010. DOI: 10.1007/s10846-009-9345-7.

[73] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control", *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018, ISSN: 0730-0301. DOI: 10.1145/3197517.3201366.

[74] M. Silva, "Quadruped Robot Optimization Using a Genetic Algorithm", Sep. 2011. DOI: 10.1142/9789814374286_0091.

[75] Y. Song, S. Kim, and D. Scaramuzza, *Learning quadruped locomotion using differentiable simulation*, 2024.

[76] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, *Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning*, 2021.

[77] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning", in *Conference on Robot Learning*.

[78] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating", in *Conference on Robot Learning*, PMLR, 2020, pp. 66–75.

[79] G. Kang, H. Seong, D. Lee, and D. H. Shim, "A versatile door opening system with mobile manipulator through adaptive position-force control and reinforcement learning", *Robotics and Autonomous Systems*, vol. 180, p. 104 760, 2024, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2024.104760.

[80] Y. Sun, L. Zhang, and O. Ma, "Force-Vision Sensor Fusion Improves Learning-based Approach for Self-Closing Door Pulling", *IEEE Access*, vol. PP, pp. 1–1, 2021. DOI: 10.1109/ACCESS.2021.3118594.

[81] M. Arduengo, C. Torras, and L. Sentis, "Robust and Adaptive Door Operation with a Mobile Robot", *Intelligent Service Robotics*, vol. 14, pp. 409–425, 2021. DOI: 10.1007/s11370-021-00366-7.

[82] L. Mochurad and Y. Hladun, "Neural Network-Based Algorithm for Door Handle Recognition Using RGBD Cameras", *Scientific Reports*, vol. 14, p. 15 759, 2024. DOI: 10.1038/s41598-024-66864-7.

[83] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Versatile multicontact planning and control for legged loco-manipulation", *Science Robotics*, vol. 8, no. 81, eadg5014, 2023. DOI: 10.1126/scirobotics.adg5014.

[84] Y. Wang, L. Wang, and Y. Zhao, "Research on Door Opening Operation of Mobile Robotic Arm Based on Reinforcement Learning", *Applied Sciences*, vol. 12, no. 10, 2022. DOI: 10.3390/app12105204.

[85] S. Dalibard, A. Nakhaei, F. Lamiraux, and J.-P. Laumond, "Manipulation of Documented Objects by a Walking Humanoid Robot", in *Proceedings of the 10th IEEE-RAS International Conference on Humanoid Robots*, Nashville, United States, 2010, pp. 518–523.

[86] Unitree Robotics, *Unitree Z1*, 2020. [Online]. Available: https://shop.unitree.com/products/unitree-z1 (visited on 12/08/2024).

[87] Intel Corporation, *Intel®RealSense™Depth Camera SR300 Series Product Family Datasheet*, Revision 002, Jun. 2019. [Online]. Available: https://www.intelrealsense.com/wp-content/uploads/2019/07/RealSense_SR30x_Product_Datasheet_Rev_002.pdf (visited on 12/08/2024).

[88] Open Robotics, *ROS - Robot Operating System*, 2024. [Online]. Available: https://www.ros.org/ (visited on 12/08/2024).

[89] E. Ackerman and E. Guizzo, "Wizards of ROS: Willow Garage and the Making of the Robot Operating System", *IEEE Spectrum*, Nov. 2017. [Online]. Available: https://spectrum.ieee.org/wizards-of-ros-willow-garage-and-the-making-of-the-robot-operating-system.

[90] O. Robotics, *Gazebo Simulation Environment*. [Online]. Available: https://gazebosim.org/home (visited on 12/08/2024).

[91] D. Hershberger, J. Faust, D. Gossow, and W. Woodall, *RViz Documentation*, 2024. [Online]. Available: http://wiki.ros.org/rviz (visited on 12/08/2024).

[92] O. Robotics, *rosbag*. [Online]. Available: https://wiki.ros.org/rosbag (visited on 12/08/2024).

[93] Q. Liao, *legged_control: A Control Framework for Legged Robots*, GitHub repository, 2024. [Online]. Available: https://github.com/qiayuanl/legged_control (visited on 12/08/2024).

[94] T. Flayols, A. Del Prete, P. Wensing, A. Mifsud, M. Benallegue, and O. Stasse, "Experimental evaluation of simple estimators for humanoid robots", in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Birmingham, United Kingdom: IEEE Press, 2017, pp. 889–895. DOI: 10.1109/HUMANOIDS.2017.8246977.

[95] T. Zhang, F. Lin, X. Peng, X. Xiong, and Y. Lou, *qm_control: A Control Framework for Legged Manipulator Robots*, GitHub repository, 2024. [Online]. Available: https://github.com/skywoodsz/qm_control (visited on 12/08/2024).

[96] T. Zhang, F. Lin, X. Peng, X. Xiong, and Y. Lou, "Whole-body Compliance Control for Quadruped Manipulator with Actuation Saturation of Joint Torque and Ground Friction", in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[97] N. Rathod, "Model Predictive Control for Legged Robots", Ph.D. dissertation, Jun. 2023. DOI: 10.13140/RG.2.2.35910.65601.

[98] G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020, 21st IFAC World Congress, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.073.

[99] J. Carpentier, F. Valenza, N. Mansard, and others, *Pinocchio: Fast forward and inverse dynamics for poly-articulated systems*, 2021. [Online]. Available: https://stack-of-tasks.github.io/pinocchio (visited on 12/08/2024).

[100] 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 Integrations (SII)*, 2019.

[101] Eigen Development Team, *Eigen: a C++ template library for linear algebra*, Official Website, 2024. [Online]. Available: https://eigen.tuxfamily.org/ (visited on 12/08/2024).

[102] Flexible Collision Library Contributors, *Flexible Collision Library (FCL)*, GitHub repository, 2024. [Online]. Available: https://github.com/flexible-collision-library/fcl (visited on 12/08/2024).

[103] Humanoid Path Planner Contributors, *HPP-FCL: Humanoid Path Planner Flexible Collision Library*, GitHub repository, 2024. [Online]. Available: https://github.com/humanoid-path-planner/hpp-fcl (visited on 12/08/2024).

[104] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space", *IEEE Journal of Robotics and Automation*, vol. 4, pp. 193–203, Apr. 1988.

[105] A. H. Reynolds, *Convolutional Neural Networks (CNNs)*, 2019. [Online]. Available: https://anhreynolds.com/blogs/cnn.html (visited on 12/08/2024).

[106] K. O'Shea and R. Nash, *An Introduction to Convolutional Neural Networks*, 2015. [Online]. Available: https://arxiv.org/abs/1511.08458 (visited on 12/08/2024).

[107] N. Shahriar, *What is Convolutional Neural Network – CNN (Deep Learning)*, Medium, 2023. [Online]. Available: https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5 (visited on 12/08/2024).

[108] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[109] Ultralytics, *Ultralytics GitHub Repository*, GitHub repository, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics (visited on 12/08/2024).

[110] R. Kaur and S. Singh, "A comprehensive review of object detection with deep learning", *Digital Signal Processing*, vol. 132, p. 103 812, 2023, ISSN: 1051-2004. DOI: https://doi.org/10.1016/j.dsp.2022.103812.

[111] Ultralytics, *Instance Segmentation*, Ultralytics Official Website, 2024. [Online]. Available: https://docs.ultralytics.com/tasks/segment/ (visited on 12/08/2024).

[112] PyTorch, *Getting Started with PyTorch Locally*, 2024. [Online]. Available: https://pytorch.org/get-started/locally/ (visited on 12/08/2024).

[113] Roboflow, *Roboflow: Optimize Your Computer Vision Workflow*, 2024. [Online]. Available: https://roboflow.com/ (visited on 12/08/2024).

[114] OpenCV, *Open Source Computer Vision Library*, 2024. [Online]. Available: https://opencv.org/ (visited on 12/08/2024).

[115] NumPy, *NumPy*, 2024. [Online]. Available: https://numpy.org/ (visited on 12/08/2024).

[116] Point Cloud Library, *Point Cloud Library (PCL)*, 2024. [Online]. Available: https://pointclouds.org/ (visited on 12/08/2024).

[117] scikit-learn, *scikit-learn: Machine Learning in Python*, 2024. [Online]. Available: https://scikit-learn.org/stable/ (visited on 12/08/2024).

[118] MRPT, *RANSAC C++ examples*, GitHub repository, 2024. [Online]. Available: https://docs.mrpt.org/reference/latest/tutorial-ransac.html (visited on 12/08/2024).

[119] C. Cheng, *Principal Component Analysis (PCA) Explained Visually*, Medium, 2022. [Online]. Available: https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d (visited on 12/08/2024).

[120] Unitree Robotics, *Unitree ROS*, GitHub repository, 2024. [Online]. Available: https://github.com/unitreerobotics/unitree_ros (visited on 12/08/2024).

[121] Intel Corporation, *Intel®RealSense™Depth Camera D400 Series Product Family Datasheet*, Revision 005, Jan. 2019. [Online]. Available: https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf (visited on 12/08/2024).

[122] Intel RealSense, *Intel RealSense SDK*, GitHub repository, 2024. [Online]. Available: https://github.com/IntelRealSense/librealsense (visited on 12/08/2024).

[123] J. Jaramillo, *URDF-door*, GitHub repository, 2019. [Online]. Available: https://github.com/JoseJaramillo/URDF-door (visited on 12/08/2024).

[124] D. Sotelo, *Door Object Detection Dataset*, Open Source Dataset, Jun. 2024. [Online]. Available: https://universe.roboflow.com/daniws/door-object-detection-5snyd (visited on 12/08/2024).

[125] D. Sotelo, *Door Handle Segmentation Dataset*, Open Source Dataset, Jun. 2024. [Online]. Available: https://universe.roboflow.com/daniws/handle-detection-r4gr5/dataset/5 (visited on 12/08/2024).

[126] H. Rana and H. Sirohia, "Comparative Study Between Canny and Sobel Edge Detection Techniques", Aug. 2020.

[127] S. Guan and D. Qi, "Defects Description in Blockboard by Hough Transform and Minimum-Perimeter Polygons", *International Journal of Advancements in Computing Technology*, vol. 4, pp. 365–375, Dec. 2012. DOI: 10.4156/ijact.vol4.issue23.43.

[128] *Original Prusa MK4 3D Printer*, 2024. [Online]. Available: https://www.prusa3d.com/product/original-prusa-mk4s-3d-printer-5/ (visited on 12/08/2024).

[129] C. Cruz, J. del Cerro, and A. Barrientos, "Perception Sensor Integration for Improved Environmental Reconstruction in Quadruped Robotics", in *Jornadas de Automática*, vol. 45, 2024. DOI: 10.17979/ja-cea.2024.45.10830. [Online]. Available: https://doi.org/10.17979/ja-cea.2024.45.10830.

[130] D. Sotelo, *Quadruped Manipulator NMPC for SAR and Door Opening Tasks*, GitHub repository, 2024. [Online]. Available: https://github.com/danisotelo/qm_door (visited on 12/08/2024).

Appendix A

Code of Interest

This appendix provides an overview of the software architecture and codebase hosted on GitHub at . The repository is well-documented through a `README.md` file but some key comments will be given here. The repository is structured as a ROS package and includes a comprehensive suite of sub-packages, each tailored for specific functionalities required for simulation and operation on real hardware [130].

Repository Structure

The code repository is organized into three main branches:

- **Quadruped Manipulator Simulation Branch:** This branch contains all necessary configurations and enhancements for running simulations on the quadruped manipulator in SAR scenarios. It enables to control the quadruped manipulator both as a combined and as a separate system.
- **Quadruped Manipulator with Force Compliance Simulation Branch:** This branch is similar to the previous one but incorporates end-effector's force compliance. This branch includes the door opening simulation. It does not support separate system control.
- **Quadruped Manipulator Real Hardware Branch:** Dedicated to deployment on the actual system, this branch includes specific adjustments and optimizations to ensure compatibility with the physical components of the quadruped manipulator.

Sub-packages Overview

The repository comprises 11 sub-packages, each serving a distinct role within the broader system architecture:

- **`qm_common/`:** This sub-package includes the contact sensor and hybrid joint hardware interfaces used across other sub-packages.
- **`qm_controllers/`:** It contains and manages the whole control architecture responsible for managing the locomotion and manipulation tasks executed by the robot.

- **`qm_description/`:** Houses the URDF models and mesh files that describe the physical and visual elements of the robot and simulation environments.
- **`qm_estimation/`:** Implements the Kalman filter estimation algorithm for accurate perception of the robot's position and orientation relative to its environment.
- **`qm_gazebo/`:** This sub-package is specifically geared towards integrating with the Gazebo simulation environment, containing plugins and launch and configuration files for simulation.
- **`qm_interface/`:** Sets up the NMPC optimal control problem, manages the model dynamics, reference trajectories, state estimations, constraints and cost functions.
- **`qm_msgs/`:** Defines the ROS messages used for communication within the sub-packages and external nodes.
- **`qm_planner/`:** Contains the planning algorithm that computes the strategy for navigation and manipulation tasks.
- **`qm_vision/`:** Dedicated to vision processing, this sub-package implements algorithms for the door opening task.
- **`qm_wbc/`:** Houses the WBC control algorithms that coordinate complex movements and interactions of the robot and solves the hierarchical QP problems.
- **`qpoases_catkin/`:** A wrapper for the qpOASES library, which provides optimization routines used particularly in the control algorithms.

Ongoing development aims to enhance the capabilities of the ROS package to extend the functionalities to cover more advanced tasks. Contributions and feedback from the community are welcomed to drive further innovations and improvements.

Appendix B

NMPC and WBC Controller Parameters

B.1. NMPC Parameters

Table B.1. State Weight Matrix $\boldsymbol{Q}$.

Index (i,j)	Parameter	Description	Weight
(0,0)	v_{com_x}	Normalized Centroidal Momentum (linear)	50.0
(1,1)	v_{com_y}	Normalized Centroidal Momentum (linear)	50.0
(2,2)	v_{com_z}	Normalized Centroidal Momentum (linear)	300.0
(3,3)	L_x/M	Centroidal Angular Momentum	10.0
(4,4)	L_y/M	Centroidal Angular Momentum	30.0
(5,5)	L_z/M	Centroidal Angular Momentum	30.0
(6,6)	p_{base_x}	Base Pose (position)	1000.0
(7,7)	p_{base_y}	Base Pose (position)	1000.0
(8,8)	p_{base_z}	Base Pose (position)	3000.0
(9,9)	ψ	Base Pose (orientation)	1000.0
(10,10)	ϕ	Base Pose (orientation)	2000.0
(11,11)	θ	Base Pose (orientation)	2000.0
(12,12)	`LF_HAA`	Leg Joint Positions (LF)	5.0
(13,13)	`LF_HFE`	Leg Joint Positions (LF)	5.0
(14,14)	`LF_KFE`	Leg Joint Positions (LF)	2.5
(15,15)	`LH_HAA`	Leg Joint Positions (LH)	5.0
(16,16)	`LH_HFE`	Leg Joint Positions (LH)	5.0
(17,17)	`LH_KFE`	Leg Joint Positions (LH)	2.5
(18,18)	`RF_HAA`	Leg Joint Positions (RF)	5.0
(19,19)	`RF_HFE`	Leg Joint Positions (RF)	5.0
(20,20)	`RF_KFE`	Leg Joint Positions (RF)	2.5
(21,21)	`RH_HAA`	Leg Joint Positions (RH)	5.0
(22,22)	`RH_HFE`	Leg Joint Positions (RH)	5.0
(23,23)	`RH_KFE`	Leg Joint Positions (RH)	2.5
(24,24)	p_{j_1}	Arm Joint Positions	5.0
(25,25)	p_{j_2}	Arm Joint Positions	5.0
(26,26)	p_{j_3}	Arm Joint Positions	5.0
(27,27)	p_{j_4}	Arm Joint Positions	5.0
(28,28)	p_{j_5}	Arm Joint Positions	0.0
(29,29)	p_{j6}	Arm Joint Positions	0.0

Table B.2. Control Weight Matrix $\boldsymbol{R}$.

Index (i,j)	Parameter	Description	Weight
(0,0)	$f_{x_{LF}}$	Feet Contact Forces (LF)	5.0
(1,1)	$f_{y_{LF}}$	Feet Contact Forces (LF)	5.0
(2,2)	$f_{z_{LF}}$	Feet Contact Forces (LF)	5.0
(3,3)	$f_{x_{RF}}$	Feet Contact Forces (RF)	5.0
(4,4)	$f_{y_{RF}}$	Feet Contact Forces (RF)	5.0
(5,5)	$f_{z_{RF}}$	Feet Contact Forces (RF)	5.0
(6,6)	$f_{x_{LH}}$	Feet Contact Forces (LH)	5.0
(7,7)	$f_{y_{LH}}$	Feet Contact Forces (LH)	5.0
(8,8)	$f_{z_{LH}}$	Feet Contact Forces (LH)	5.0
(9,9)	$f_{x_{RH}}$	Feet Contact Forces (RH)	5.0
(10,10)	$f_{y_{RH}}$	Feet Contact Forces (RH)	5.0
(11,11)	$f_{z_{RH}}$	Feet Contact Forces (RH)	5.0
(12,12)	f_x	Arm Force	0.0
(13,13)	f_y	Arm Force	0.0
(14,14)	f_z	Arm Force	0.0
(15,15)	w_x	Arm Force	0.0
(16,16)	w_y	Arm Force	0.0
(17,17)	w_z	Arm Force	0.0
(18,18)	x_{LF}	Foot velocity (LF)	5000.0
(19,19)	y_{LF}	Foot velocity (LF)	5000.0
(20,20)	z_{LF}	Foot velocity (LF)	5000.0
(21,21)	x_{LH}	Foot velocity (LH)	5000.0
(22,22)	y_{LH}	Foot velocity (LH)	5000.0
(23,23)	z_{LH}	Foot velocity (LH)	5000.0
(24,24)	x_{RF}	Foot velocity (RF)	5000.0
(25,25)	y_{RF}	Foot velocity (RF)	5000.0
(26,26)	z_{RF}	Foot velocity (RF)	5000.0
(27,27)	x_{RH}	Foot velocity (RH)	5000.0
(28,28)	y_{RH}	Foot velocity (RH)	5000.0
(29,29)	z_{RH}	Foot velocity (RH)	5000.0
(30,30)	v_{j_1}	Arm Joint velocity	3000.0
(31,31)	v_{j_2}	Arm Joint velocity	3000.0
(32,32)	v_{j_3}	Arm Joint velocity	3000.0
(33,33)	v_{j_4}	Arm Joint velocity	1000.0
(34,34)	v_{j_5}	Arm Joint velocity	1000.0
(35,35)	v_{j_6}	Arm Joint velocity	1000.0

B.2. WBC Parameters

Table B.3. Control Parameters for Whole-Body Controller (WBC).

Parameter	Description	Value
$K_{P_{\text{swing}}}$	K_P of swing leg	350.0
$K_{D_{\text{swing}}}$	K_D of swing leg	37.0
$K_{P_{\text{BH}}}$	K_P of base height	400.0
$K_{D_{\text{BH}}}$	K_D of base height	140.0
$K_{P_{\text{BL}}}$	K_P of base linear	400.0
$K_{D_{\text{BL}}}$	K_D of base linear	140.0
$K_{P_{\text{BA}}}$	K_P of base angular	400.0
$K_{D_{\text{BA}}}$	K_D of base angular	140.0
$K_{P_{\text{J1}}}$	K_P of arm joint 1	4000.0
$K_{P_{\text{J2}}}$	K_P of arm joint 2	4200.0
$K_{P_{\text{J3}}}$	K_P of arm joint 3	4000.0
$K_{P_{\text{J4}}}$	K_P of arm joint 4	4000.0
$K_{P_{\text{J5}}}$	K_P of arm joint 5	4200.0
$K_{P_{\text{J6}}}$	K_P of arm joint 6	6000.0
$K_{D_{\text{J1}}}$	K_D of arm joint 1	75.0
$K_{D_{\text{J2}}}$	K_D of arm joint 2	75.0
$K_{D_{\text{J3}}}$	K_D of arm joint 3	75.0
$K_{D_{\text{J4}}}$	K_D of arm joint 4	75.0
$K_{D_{\text{J5}}}$	K_D of arm joint 5	75.0
$K_{D_{\text{J6}}}$	K_D of arm joint 6	75.0
$K_{P_{\text{EE},x}}$	K_P of arm linear X	3000.0
$K_{P_{\text{EE},y}}$	K_P of arm linear Y	3000.0
$K_{P_{\text{EE},z}}$	K_P of arm linear Z	4000.0
$K_{D_{\text{EE},x}}$	K_D of arm linear X	75.0
$K_{D_{\text{EE},y}}$	K_D of arm linear Y	75.0
$K_{D_{\text{EE},z}}$	K_D of arm linear Z	75.0
$K_{P_{\text{EE},\theta}}$	K_P of arm angular X	2000.0
$K_{P_{\text{EE},\phi}}$	K_P of arm angular Y	2000.0
$K_{P_{\text{EE},\psi}}$	K_P of arm angular Z	2000.0
$K_{D_{\text{EE},\theta}}$	K_D of arm angular X	75.0
$K_{D_{\text{EE},\phi}}$	K_D of arm angular Y	75.0
$K_{D_{\text{EE},\psi}}$	K_D of arm angular Z	75.0

Appendix C

Applications and Impact Analysis

C.1. Applications

This project has significant potential for application in various fields, particularly in areas where mobility and manipulation in complex unstructured environments are crucial. The integration of the Unitree Aliengo quadruped robot with the Z1 manipulator enables the system to perform tasks such as **SAR operations** in disaster-stricken areas, where the robot can navigate through rubble and debris to locate and assist victims. In addition, the system could be employed in **industrial maintenance** tasks, where the ability to reach and manipulate objects in hazardous or hard-to-access locations is essential, reducing the risk to human workers. The robot's capability to autonomously open doors and traverse obstacles also opens up possibilities for **military applications**, where it could be used for reconnaissance or supply missions in hostile or dangerous environments.

C.2. Impact Analysis

The **economic impact** of the project is significant, particularly in terms of the **initial investment** required for high-tech equipment like the Unitree Aliengo, Z1 manipulator, and other hardware components. Additionally, the costs associated with research, development, and testing are considerable. However, these investments are justified by the **potential to save human lives** during SAR operations, where the use of robots reduces the need for human rescuers to enter dangerous environments. In the long term, the deployment of such systems could lead to cost savings by minimizing human risk and increasing the efficiency of operations.

Socially, the project has a profound impact by enhancing the **safety of first responders** and potentially **reducing fatalities** in disaster scenarios. The robot's ability to perform tasks such as navigating through debris, opening doors, and manipulating objects can be crucial in saving lives. Additionally, the integration of such technology

in SAR operations highlights the positive role of automation in high-risk situations, contributing to the broader acceptance of robots in roles traditionally occupied by humans. However, it is also important to consider the ethical implications, particularly concerning **job displacement**, as automation could reduce the need for human personnel in certain roles.

The **legal impact** of the project is also an important consideration. The deployment of autonomous robots in SAR operations introduces a number of legal challenges. Issues surrounding **liability** in case of equipment failure, **compliance with safety standards**, and adherence to **privacy laws** when operating in sensitive environments must be addressed. For instance, if a robot malfunctions during a rescue operation, determining the responsibility could be complex, involving manufacturers, software developers, or operators. Furthermore, ensuring the robot adheres to regional and international regulations governing the use of autonomous systems in public spaces is essential. Legal frameworks must evolve alongside technological advancements to ensure that the benefits of such systems are harnessed while minimizing potential risks to individuals and society. Lastly, data security and the handling of sensitive information collected during operations must comply with the General Data Protection Regulation (GDPR), ensuring that personal information is safeguarded during rescue missions.

The **environmental impact** of the project involves the materials used in the construction of the robot, including metals and plastics, which contribute to its carbon footprint. Additionally, the electronic components and the energy required to operate the robot add to the environmental considerations. While the use of advanced robotics in SAR operations has clear societal benefits, a **life cycle assessment** would be beneficial to identify areas where the environmental footprint could be reduced, ensuring the project aligns with broader sustainability goals.

C.3. Contribution to Sustainable Development Goals

The social robot developed in this project contributes to several Sustainable Development Goals (SDGs), as highlighted below and indicated in Figure C.1.

- **SDG 3 - Good Health and Well-Being:** By enhancing SAR operations, the project directly supports efforts to save lives and reduce injuries in disaster scenarios, contributing to overall well-being.
- **SDG 8 - Decent Work and Economic Growth:** The project encourages the development of new job opportunities in the fields of robotics, programming, and maintenance. By fostering innovation and technological advancement, it supports sustainable economic growth and productive employment.

Figure C.1. SDGs to which the project contributes.

- **SDG 9 - Industry, Innovation, and Infrastructure:** The project promotes innovation in robotics technology, pushing the boundaries of what is possible in SAR operations and contributing to the development of resilient infrastructure.
- **SDG 11 - Sustainable Cities and Communities:** By improving the efficiency and effectiveness of emergency response in urban and disaster-prone areas, the project helps create safer, more resilient communities.
- **SDG 13 - Climate Action:** The project indirectly supports climate action by improving disaster response capabilities, which is increasingly important in the context of climate-related disasters such as floods, hurricanes, and wildfires.
- **SDG 17 - Partnerships for the Goals:** The project exemplifies the importance of collaboration between research institutions, technology providers, and other stakeholders. By leveraging expertise and resources from various sectors, it helps build effective partnerships that contribute to the achievement of SDGs.

Appendix D

Temporal Planning and Budget

D.1. Work Breakdown Structure (WBS)

The Work Breakdown Structure (WBS) shown in Figure D.1 provides a comprehensive overview of the project's phases, outlining the key tasks and milestones required to integrate and control the legged-manipulator platform.

D.2. Planning

This section details the project timeline corresponding to the work packages defined in the WBS using a Gantt chart. The Gantt chart is shown in Figure D.2.

D.3. Budget

This budget section outlines the financial resources required for the project. It includes a breakdown of costs associated with materials, equipment, and personnel.

D.3.1. Material Costs

Table D.1 summarizes the expenses for materials used in the project, including materials and electronics.

D.3.2. Equipment Amortization Costs

The amortization costs of equipment used during the project, such as the 3D printer and the robots, are calculated distributing their initial costs over the useful life of these tools, as summarized in Table D.2.

D.3.3. Personnel Costs

Table D.3 accounts for the labor costs, including the time and expertise of the team members involved in the project, calculated based on their respective roles and hourly rates.

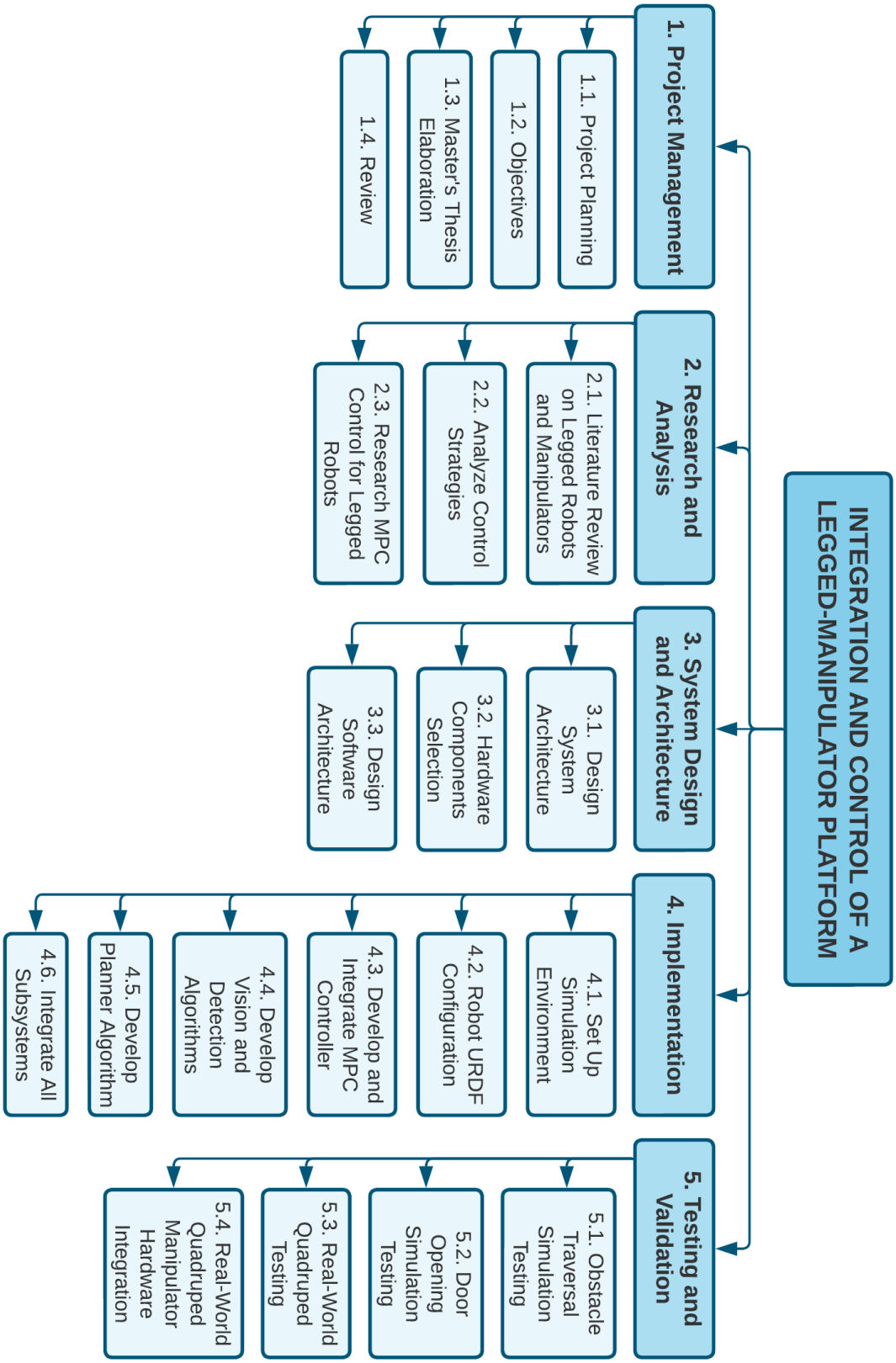
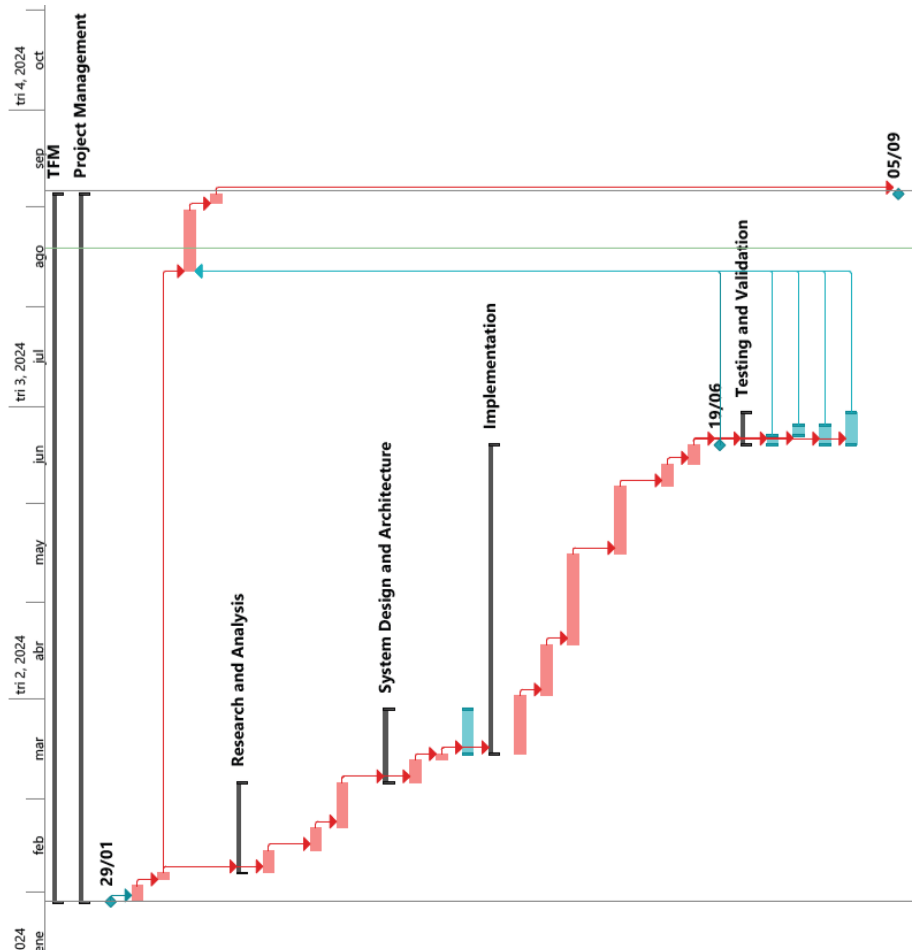


Figure D.1. Project WBS.



Nombre de tarea	Duración	Comienzo	Fin
TFM	**158 días**	**lun 29/01/24**	**jue 05/09/24**
Project Management	**158 días**	**lun 29/01/24**	**mié 04/09/24**
Initial Meeting	0 días	lun 29/01/24	lun 29/01/24
Project Planning	5 días	lun 29/01/24	vie 02/02/24
Objectives	2 días	lun 05/02/24	mar 06/02/24
Master's Thesis Elaboration	15 días	lun 12/08/24	vie 30/08/24
Review	3 días	lun 02/09/24	mié 04/09/24
Research and Analysis	**20 días**	**mié 07/02/24**	**mar 05/03/24**
Literature Review on Legged Robots and Manipulators	5 días	mié 07/02/24	mar 13/02/24
Analyze Control Strategies	5 días	mié 14/02/24	mar 20/02/24
Research MPC Control for Legged Robots	10 días	mié 21/02/24	mar 05/03/24
System Design and Architecture	**17 días**	**mié 06/03/24**	**jue 28/03/24**
Design System Architecture	5 días	mié 06/03/24	mar 12/03/24
Hardware Components Selection	2 días	mié 13/03/24	jue 14/03/24
Design Software Architecture	10 días	vie 15/03/24	jue 28/03/24
Implementation	**68 días**	**vie 15/03/24**	**mié 19/06/24**
Set Up Simulation Environment	12 días	vie 15/03/24	lun 01/04/24
Robot URDF Configuration	12 días	mar 02/04/24	mié 17/04/24
Develop and Integrate MPC Controller	20 días	jue 18/04/24	mié 15/05/24
Develop Vision and Detection Algorithms	15 días	jue 16/05/24	mié 05/06/24
Develop Planner Algorithm	5 días	jue 06/06/24	mié 12/06/24
Integrate All Subsystems	4 días	jue 13/06/24	mar 18/06/24
Finished Implementation	0 días	mié 19/06/24	mié 19/06/24
Testing and Validation	**8 días**	**mié 19/06/24**	**vie 28/06/24**
Obstacle Traversal Simulation Te	3 días	mié 19/06/24	vie 21/06/24
Door Opening Simulation Testing	2 días	sáb 22/06/24	lun 24/06/24
Real-World Quadruped Testing	4 días	mié 19/06/24	lun 24/06/24
Real-World Quadruped Manipulator Hardware	8 días	mié 19/06/24	vie 28/06/24
Project Delivery	0 días	jue 05/09/24	jue 05/09/24

Figure D.2. Project Gantt Chart.

Concept	Units	Unit Price	Total Price
PLA Filament	1	15,99 €	15,99 €
TPU Filament	1	23,99 €	23,99 €
10 mm PMMA Sheet	1	12,45 €	12,45 €
USB 3.0 Hub	1	9,70 €	9,70 €
Total Material Cost			**62,13 €**

Table D.1. Project Material Costs.

Concept	Total Cost	Years	Amortization Cost / Year	Use	Amortization Cost
Prusa MK4 3D Printer	1.199,00 €	5	239,80 €	2 days	1,31 €
Unitree Aliengo + D435 camera	50.000,00 €	7	7.142,86 €	3 weeks	410,98 €
Unitree Z1	15.999,00 €	7	2.285,57 €	2 weeks	87,67 €
Intel RealSense SR305 camera	18,99 €	3	6,33 €	3 months	1,58 €
Toolbox	65,30 €	10	6,53 €	3 weeks	0,38 €
Drill Milling Machine	1.634,00 €	15	108,93 €	1 day	0,30 €
Laptop	1.250,00 €	5	250,00 €	158 days	108,22 €
Total Amortization Cost					**610,44 €**

Table D.2. Project Amortization Costs.

Concept	Hour Cost	Hours	Cost
Project Author	7,00 €	330	2.310,00 €
Project Director	40,00 €	10	400,00 €
Postdoctoral Researcher	20,00 €	30	600,00 €
Total Personnel Cost			**3.310,00 €**

Table D.3. Project Personnel Costs.

D.3.4. Total Cost

Table D.4 shows the total project costs calculation, combining material, equipment, and personnel expenses to give a complete financial overview of the project's expenditure.

Totals	Cost
Total Material Cost	62,13 €
Total Amortization Cost	610,44 €
Total Personnel Cost	3.310,00 €
TOTAL PROJECT COST	**3.982,57 €**

Table D.4. Project Total Cost.