Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Karoline Skobba Grøtterud

Development of Inverse Kinematics for a Redundant Mobile Service Robot in Nursing Homes

Master's thesis in Cybernetics and Robotics
Supervisor: Prof. Jan Tommy Gravdahl
Co-supervisor: Prof. Trygve Thomessen
June 2024

NTNU
Norwegian University of
Science and Technology

Karoline Skobba Grøtterud

Development of Inverse Kinematics for a Redundant Mobile Service Robot in Nursing Homes

Master's thesis in Cybernetics and Robotics
Supervisor: Prof. Jan Tommy Gravdahl
Co-supervisor: Prof. Trygve Thomessen
June 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

Integrating service robots in elderly care facilities is a potential solution to the strain on healthcare systems caused by an aging population. A robot designed for this purpose is the mobile assistive robot, Lio, by F&P Robotics AG in Switzerland. Lio consists of a six-axis anthropomorphic robot arm mounted on top of a two-axis differential drive mobile platform, currently controlled as two uncoupled kinematic systems. In this thesis, an eight-axis kinematic model of Lio is developed to better utilize the advantage of the redundant degrees of freedom in various nursing home tasks.

Various redundancy techniques were explored to satisfy different secondary tasks beneficial in a nursing home environment. First, the extended task space method was implemented, augmenting the task vector with the mobile platform's configuration as a constraint. This approach exploited the redundant degrees of freedom through configuration control, ensuring predictable robot poses. Next, two optimization-based methods were developed, treating the redundancy as a nonlinear optimization problem. These methods aimed to solve automatic platform positioning, useful in task planning and autonomous operations, by finding configurations that minimize platform movement and maximize manipulability. The first optimization strategy solved the inverse kinematics by first optimizing a set of redundancy parameters to determine the platform configuration and then exploiting the robot arm's analytic inverse kinematic solution. The second optimization strategy solved the eight-axis inverse kinematics problem numerically while maximizing manipulability as a secondary task. Both optimization methods showed promising results for utilizing redundancy in autonomous operations. These control strategies demonstrated solutions for a better and more flexible motion control and task programming of the Lio service robot.

Sammendrag

Innføring av serviceroboter i eldreomsorgen er et tiltak som kan bidra til at helsevesenet kan håndtere den kommende "eldrebølgen" bedre. Lio er et eksempel på en slik servicerobot, designet for å lette arbeidsbyrden for helsepersonell på sykehjem samt bidra til økt pasientvelferd. Lio, fra F&P Robotics AG i Sveits, er en seks-akset, antropomorf robotarm som er festet på toppen av en to-akset mobil robotplattform. Fram til nå har roboten og plattformen blitt styrt som to uavhengige, dekoblede kinematiske systemer, noe som gjør Lio lite brukervennlig, og utnytter robotens arbeidsområde i begrenset grad. Denne oppgaven utvikler derfor en komplett åtte-akset kinematisk modell for Lio slik at den kan utnyttes fullt ut til sine oppgaver på sykehjemmene.

Tre strategier er presentert, implementert og testet for å demonstrere hvordan disse kan bidra til å utnytte alle Lio sine frihetsgrader i forskjellige anvendelser. Den første metoden, utvidelse av oppgaverommet (extended task space), inkluderte plattformens frihetsgrader i robotens positur. Denne strategien utnyttet de overtallige frihetsgradene gjennom konfigurasjonsstyring, som kan benyttes til å sikre forutsigbare robotpositurer. Deretter ble to optimaliseringsbaserte metoder utviklet, der robotens redundante inverskinematikk ble formulert som et ulineært optimaliseringsproblem. Disse metodene hadde som mål å løse automatisk plattformposisjonering, som er nyttig både i forbindelse med planlegging og programmering av oppgaven, samt under automatisk kjøring. Den første optimaliseringsstrategien løste inverskinematikken ved å først optimalisere et sett med redundansparametere for å bestemme plattformens posisjon, for deretter å benytte den analytiske inverskinematikk-modellen av robotarmen, til å finne robotens leddpositur. Den andre optimaliseringsstrategien løste inverskinematikken numerisk med maksimering av manipulerbarhet som sekundæroppgave. De eksperimentelle resultatene verifiserte at alle løsningene var korrekt implementert samt demonstrerte lovende resultater med tanke på å fordelaktig kunne utnytte overtallige frihetsgrader til en mer effektiv og fleksibel bruk av serviceroboten, Lio.

Preface

This master's thesis, conducted throughout the spring of 2024, concludes my Master of Science in Cybernetics and Robotics at the Norwegian University of Science and Technology. The problem description was developed by Prof. Trygve Thomessen on behalf of PPM Robotics AS and concerns the development of inverse kinematics on a redundant mobile service robot for nursing homes. The work carried out in this thesis is a continuation of my specialization project, written in the fall of 2023: *Service robot for nursing homes: Kinematic modeling and motion planning for a mobile redundant robot* (Grøtterud, 2023).

The kinematic model derived in the specialization project laid the foundation for the redundancy resolutions developed in this thesis. Therefore, some sections of this report overlap with previous work. These sections are indicated in the introduction of each chapter according to the level of modification from the previous report, as either *adapted from* the specialization project, meaning the sections are nearly identical with only minor adjustments, or *based on* the specialization project, meaning that the essential ideas or results are the same, but that the section has been significantly altered or restructured.

I would like to take the opportunity to thank my supervisor, Prof. Jan Tommy Gravdahl, and co-supervisor, Prof. Trygve Thomessen, for their support and encouragement during this project, as well as for valuable advice and feedback on drafts. An additional thanks go to Prof. Trygve Thomessen for giving me access to the PPM Nursing home lab and for personally ensuring I had the necessary equipment to complete my work when my computer failed close to the deadline. Furthermore, I would like to thank Balint Tahi from PPM Robotics for always responding quickly to any of my questions regarding the Lio robot. Lastly, I would like to thank Dr. Dirk Peter Reinhardt for offering critical support with the CasADi framework and insightful discussions regarding the developed optimization-based redundancy strategies.

Karoline Skobba Grøtterud
Trondheim, June 2024

Contents

List of Figures

List of Tables

Acronyms

DH-parameters Denavit Hartenberg parameters.

DOF Degrees of freedom.

IPOPT Interior Point OPTimizer.

NLP Non-linear optimization problem.

PoE Product of Exponentials.

SWMR Steerable wheeled mobile robot.

TCP Tool center point.

Chapter 1

Introduction

Section 1.1 and 1.2 are derived from, the specialization project (Grøtterud, 2023).

1.1 Motivation

The rapid increase in the elderly population in Norway is an undeniable reality. According to Statisticks Norway (2020), in less than ten years, the number of elderly individuals will surpass both children and adults. By the year 2060, the population over the age of 70 is projected to double. This demographic shift underscores the critical need for swift and innovative transformations within the healthcare sector. Incorporating advanced technologies, such as service robots for nursing homes, might be a crucial part of the solution. These service robots are intended to assist healthcare workers with routine tasks like walking assistance and the transfer of linen, waste, and food, thereby giving healthcare workers more time for clinical tasks and meaningful engagements with patients. The deployment of service robots aligns with the United Nations Sustainable Development Goals (SDGs) in addressing the challenges posed by the aging population (United Nations, 2012). The development of service robots contributes to the realization of SDG 3, "Good Health and Well-being," by potentially enhancing the quality and efficiency of healthcare services.

A robot designed for this purpose is Lio, a mobile assistive robot developed by F&P Robotics AG in Switzerland. Lio consists of a six-axis anthropomorphic robot arm mounted on top of a two-axis differential drive mobile platform, currently controlled as two uncoupled kinematic systems. This setup enables the robot to navigate autonomously and execute tasks such as grabbing, moving, and retrieving objects. As of today, Lio performs all its manipulation tasks through pre-programmed instructions. Consequently, Lio has difficulties with completing tasks efficiently in dynamic and unpredictable environments, such as nursing homes. For instance, imagine Lio is serving a glass of water to a patient. Lio would first drive the mobile platform to a pre-programmed position beside the table and then place the glass at a pre-determined location. If an obstacle blocks the desired

position of the platform, say the patient is sitting at an unexpected location, the entire task must be reprogrammed.

However, by connecting the platform and the arm to form a complete eight-axis kinematic model, Lio is able to relocate the platform without changing the gripper's position. This ability to perform internal joint motions without changing the end effector's pose, is a characteristic of *redundant* robots. Redundant robots have more degrees of freedom than what is required to perform a specific task. These robots are known for their flexible and high dexterity behavior because they can utilize their redundant degrees of freedom to perform secondary tasks. Achievable secondary tasks are, for example, posture control and obstacle avoidance, which are highly beneficial when operating in a nursing home environment. Hence, the objective of this thesis is to create a redundant kinematic model for the service robot Lio to achieve better and more flexible motion control and task programming.

Figure 1.1: The service robot, Lio, interacting with a nursing home resident.
Image source: (F&P Robotics AG, 2022)

1.2 About Lio

Lio is a service robot for autonomous operation in healthcare facilities and home care. A combination of visual, audio, laser, ultrasound, and mechanical sensors allows Lio to map its surroundings and safely navigate and interact with the environment (Mišeikis et al., 2020). The robot consists of an anthropomorphic robot arm mounted on a mobile platform, which enables the robot to handle and manipulate objects. Additionally, the robot has built-in AI algorithms for, amongst others, task planning, body pose estimation, and face- and object detection. Lio has been deployed in several healthcare institutions, completing tasks like delivering mail, moving blood samples, and entertaining patients.

Lio has been carefully designed to fulfill the particular requirements following the operation in a nursing home environment. Lio complies with ISO13482 - Safety requirements for personal care robots, so the robot is approved for testing and deployment in care facilities (Mišeikis et al., 2020). Processing of visual and navigation information is performed locally to ensure data privacy. Furthermore, Lio has several features that ease interaction with sick or old residents. For example, its dimensions are set so the gripper is reachable and the robot appears non-threatening for wheelchair users. Moreover, the robot is equipped with lights indicating its direction of motion, making its movements more predictable. Safe human-robot interaction is also ensured with a soft leather fabric covering the robot arm and a PE controller securing soft motion behavior.

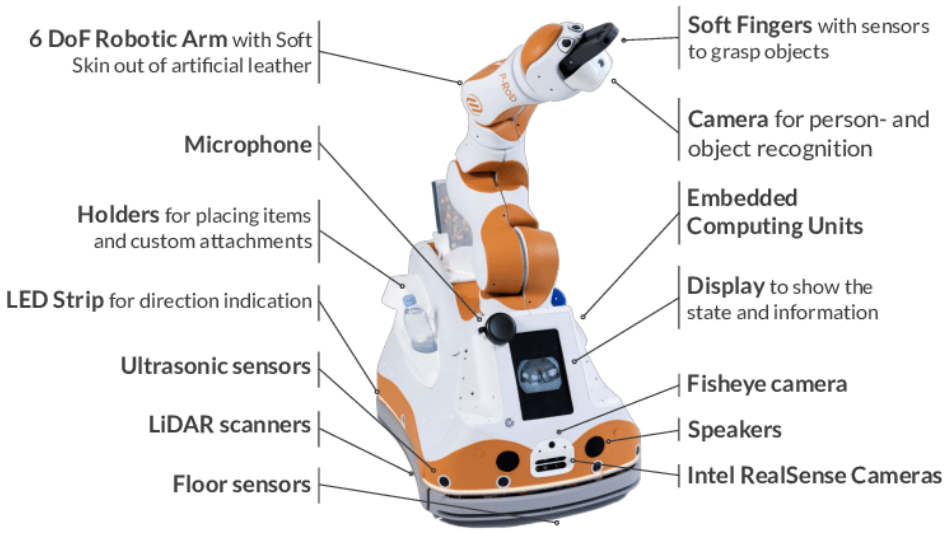


Figure 1.2: Overview of Lio hardware and sensors.
Image source: (Mišeikis et al., 2020)

1.3 Problem statement

The aim of this thesis is to develop a new eight-axis kinematic model that allows the service robot Lio to exploit its redundant degrees of freedom in various nursing home tasks. In order to achieve this objective, the work in this thesis concern the following tasks.

- Introductory literature study about service robots in healthcare and methods for redundancy resolution of mobile manipulators.
- Investigation of Lio's current control strategy.
- Development of a strategy for redundancy resolution based on an eight-axis kinematic model.
- Implementation of the new kinematic model on Lio.
- Testing and verification of the new kinematic model in the laboratory through various tasks that demonstrate its advantages.

1.4 Contributions

The main contributions of the work presented in this thesis are as follows.

- Development of an eight-axis kinematic model for Lio, completed during the specialization project (Grøtterud, 2023).
- Implementation of the new kinematic model as an object-oriented class in Python communicating with the robot control software through ROS.
- Development and implementation of a redundancy strategy using extended task space with configuration control.
- Experimental validation of the extended task space method through various tests in PPM's Nursing home lab.
- Development of two optimization-based redundancy strategies enabling automatic platform positioning.

1.5 Outline

The master thesis is structured as follows.

Chapter 1 describes the motivation, scope, and outline of the report as well as presents the basic functionality and features of the robot of interest, Lio.

Chapter 2 presents some background material concerning the implementation of robots in healthcare facilities. Furthermore, it introduces the concept of redundancy and presents redundancy resolution methods for mobile manipulators.

Chapter 3 presents a selection of theory concerning robot modeling, rigid motions and conventional redundancy resolutions.

Chapter 4 derives the forward and inverse kinematics of the eight-axis kinematic model.

Chapter 5 derives three methods for redundancy resolution on Lio: extended task space with configuration control, optimization of redundancy parameters and maximization of manipulability.

Chapter 6 describes the implementation of the kinematic model and redundancy resolutions, and the ROS communication with the Lio robot.

Chapter 7 describes the experimental test setup and results of the extended task space method and test parameters and results for the optimization-based methods.

Chapter 8 evaluates the results from Chapter 7, discusses the advantages and drawbacks of the implemented redundancy resolution strategies, and presents suggestions for future work.

Chapter 9 draws a conclusion based on the results in Chapter 7 and the following discussion in Chapter 8.

Chapter 2

Background

The background chapter provides an introduction to healthcare robots and redundancy resolution for mobile manipulators. Special considerations need to be taken when working on robots designed to operate around vulnerable groups. Therefore, it is essential to have a basic understanding of the challenges that can occur when implementing robots in healthcare. The first part of the background chapter defines different types of healthcare robots and presents an overview of some of the latest healthcare robots with manipulation capabilities. Lastly, it covers special safety requirements and stakeholder acceptance. The second part introduces the concept of redundancy and how it can be utilized to benefit robots in nursing home tasks. Finally, it presents some examples of how common redundancy resolution schemes have been adapted to work for mobile manipulators in recent research.

Section 2.1.1-2.1.3, 2.2.1 and 2.2.2 are derived from the specialization project (Grøtterud, 2023).

2.1 Healthcare robots

Healthcare robots are designed to work closely with vulnerable groups, such as residents of nursing homes. They are, therefore, subject to stricter requirements than other collaborative industrial robots. According to Servaty et al. (2020), many robotic systems that function correctly in laboratory environments have failed when implemented in the real world due to underestimating the complexity of their implementation. Successful integration of service robots in nursing homes depends on several factors, including ethical considerations, data privacy, perceived safety, and stakeholder acceptance. This section provides a brief overview of the various types of healthcare robots available on the market. Safety requirements and barriers to the implementation of robotics in healthcare are also addressed because understanding these parameters is crucial for successfully integrating robots in nursing homes.

(a) A demonstration of the lifting robot, Hug, assisting in lifting a person from the bed.
Image source: (FUJI CORPORATION, 2024)

(b) The social companion robot, Pepper, leading an exercise for patients in a Japanese day-care facility.
Image source: (Cavendish, 2018)

2.1.1 Types and features of healthcare robots

Healthcare robots are defined as "autonomous or semi-autonomous machines with a certain degree of intelligence that performs various complex tasks in various healthcare contexts" (Huang, 2022). The classification of healthcare robots varies in literature by distinguishing on either function, task, target user or medical context. This report uses the categorization proposed by Huang (2022), which divides healthcare robots into three main groups: robots as functional tools, social companions, and service assistants.

Healthcare robots as functional tools are characterized by a mechanical appearance, and their purpose is typically to perform repetitive medical tasks. Hence, it has no social capabilities or ability to learn from its environment. Examples of healthcare robots as functional tools are robotic skeletons for rehabilitation and training like Hocoma Lokomat (Hocoma, 2024) and the lifting robot, Hug (Figure 2.1a), for assisting with sit-to-stand movements.

Healthcare robots as social companions commonly have either a humanoid or animal-like appearance. Their purpose is to entertain, provide companionship or engage residents in physical or educational activities. As their primary function is interaction with voice or expression, most of the robots in this category do not have the ability to move around independently or manipulate objects despite having robot arms. Examples of robots within this classification are the pet seal, Paro (Paro, 2024), and the humanoid, Pepper (Figure 2.1b).

Healthcare robots as service assistants combine the capabilities of the healthcare robots as functional tools and social companions. These robots are typically designed to aid healthcare personnel by handling auxiliary tasks, allowing staff to focus on their primary responsibilities (Huang, 2022). Examples of these tasks include walking assistance and transferring or fetching daily items. To perform these duties effectively, assistive service robots must be capable of reacting to the environment, manipulate objects, plan tasks, and exhibit basic social capabilities.

Healthcare robots as service assistants are one of the least researched categories, possibly because of their complexity. According to Morgan et al. (2022), robots within the category "socially assistive" or "delivery transport" accounted for only 17 percent of the research done on robots in a medical environment.

2.1.2 Healthcare service robots with manipulating capabilities

Table 2.1 shows some existing healthcare service robots with manipulation capabilities. As most of these robots are commercial products, extensive details concerning robot control are not accessible. Therefore, Table 2.1 is merely a brief overview of their assumed capabilities. Moxi, Care-o-bot and TIAGo are examples of redundant mobile manipulators. However, their redundancy presumably stems from the robot arms having seven degrees of freedom, and not because their mobile platforms and robotic arms are kinematically connected.

Table 2.1: Healthcare robots with manipulation capabilities.

Robot name	**Year**	**Manufacturer**	**Intended purpose**	**Technology/features**	**Application and status**
Lio, (Mišeikis et al., 2020)	2020	F&P Robotics	Human-robot interaction and personal care assistant tasks.	Manipulator for grasping objects, sensors for autonomous navigation, algorithms for face and object detection.	Deployed in several institutions across Switzerland and Germany. Research
Moxi (Ackerman, 2018)	2018	Diligent Robotics	Assist clinical staff with non-patient-facing tasks like running patient supplies, delivering lab samples, fetching items.	Robotic arm for manipulating environment, artificial intelligence to learn tasks, sensors for autonomous navigation.	Deployed in some American institutions.
TIAGo++ (PAL Robotics, 2024)	2016	PAL Robotics	Ambient assisted living.	2 x 7-DOF robotic arm for manipulation, autonomous navigation, perception and Human-Robot interaction skills.	Research
Care-o-bot 3 (Frauenhofer IPA, 2012)	2008	Fraunhofer IPA	Support of older adults at home or support of nurses in nursing homes with fetch and carry tasks, entertainment, communication and emergency support.	Autonomous collision-free path planning, grasp and manipulate objects.	Old iteration. It can be bought for research purposes or application.
Care-o-bot 4 (Kittmann et al., 2015)	2015	Fraunhofer IPA	General purpose service assistant for, amongst others, fetching and delivering objects.	7-DOF arm, environment reconstruction, facial and gesture recognition.	For sale.
Human Support Robot, HSR (Toyota UK Magazine, 2021)	2012	Toyota	Assist elderly at home with everyday tasks.	Omnidirectional movement, robot arm with compliant control of arm joints, no autonomy.	Not for sale for the general public, open for researchers.
Jaco (Kinova Assistive, 2024)	2009	Kinova	Assist people with upper-body disabilities, letting them perform tasks like picking up objects and opening doors.	6-DOF arm, controlled by joystick.	For sale and research

2.1.3 Safety requirements and ethical considerations

During the design process, commercial robots must consider the ISO10218 - Safety requirements for industrial robots. Additionally, all robots with the intended use as mobile servants, physical assistants, or personal carriers in healthcare facilities must also comply with ISO13482 - Safety requirements for personal care robots. The latter standard underlines that the risk assessment of robots in healthcare facilities, amongst others, must pay particular concern to:

b) different levels of knowledge, experience and physical condition of users and other exposed persons (ISO 13482:2014);

c) unexpected movement of humans (ISO 13482:2014);

d) normal but unexpected movement of the personal care robot (ISO 13482:2014);

e) unintended movement of the personal care robot (ISO 13482:2014);

These concerns are of specific importance in nursing homes where residents often suffer from frailty and or impaired cognitive skills. Unexpected robot behavior can cause surprise, loss of balance, and falls, leading to severe injuries. Besides physical harm, robot behavior causing fear and discomfort significantly decreases perceived safety and trust (Akalin et al., 2022). Perceived safety and trust are crucial parameters in stakeholder acceptance, one of the most significant barriers to implementing robots in healthcare facilities (Christoforou et al., 2020).

2.1.4 Benefits of redundancy for healthcare robots

The extra demands on safety and predictable behavior in healthcare robots can be effectively addressed by using redundant robots. A redundant robot has more degrees of freedom (DOF) than required to perform a specific task (Patel and Shadpey, 2005). With redundant DOF, infinite feasible configurations exist for a given task. These additional DOF can be exploited to perform a set of user-specific tasks beneficial in a nursing home setting, such as obstacle avoidance, which simplifies maneuvering in a cluttered and dynamic environment, and configuration control, which can restrict the robot's pose to achieve more predictable behavior.

2.2 Redundancy resolution for mobile manipulators

Redundancy can be established only with respect to a given task (Siciliano, 1990). A general spatial task for a manipulator is commonly defined as moving to a desired pose or following an end effector motion trajectory, requiring six degrees of freedom (Chiaverini et al., 2016). Hence, redundancy can be introduced to a typical six-DOF manipulator by implementing additional joints or by attaching a mobile platform to the manipulator base.

The additional DOF can be exploited to avoid singularities, joint limits, and obstacles in the workspace, achieve or contain specific robot poses, or minimize the energy or velocities required to solve a task (Chiaverini et al., 2016).

Robots consisting of a manipulator mounted on top of a mobile platform are called mobile manipulators. These robots have become increasingly popular due to the combination of the manipulator's dexterity with the increased workspace of the mobile base. Redundancy resolutions were first explored on redundant manipulators, resulting in redundancy schemes such as extended task space, null space projection, and the pseudoinverse (Siciliano, 1990). These methods can also be applied to solve the inverse kinematics on redundant mobile manipulators. However, compared to conventional redundant manipulators with a fixed base, additional considerations are necessary to address the kinematics of the mobile platform. For example, the mobile base often has lower motion control accuracy due to factors like wheel slippage or lack of absolute positioning capabilities (Sorour et al., 2019). Furthermore, differential drive platforms are subject to nonholonomic constraints, whereas the constraints on the manipulator joints are holonomic. The following sections present examples of how redundancy resolutions have been adapted to suit different types of mobile manipulators.

2.2.1 Extended task space with configuration control

Seraji (1998) addresses planar motion control of a 2-link manipulator mounted on a rover. He proposes to combine the non-holonomic base constraint, the desired end-effector motion, and the user-specified redundancy resolution goals to form a set of augmented differential kinematics equations. The result is an extended Jacobian on the form

$$\begin{bmatrix} \mathbf{J}_r(\mathbf{q}) \\ \mathbf{J}_m(\mathbf{q}) \\ \mathbf{J}_c(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} = \begin{bmatrix} \mathbf{0} \\ \dot{\mathbf{X}}_d \\ \dot{\mathbf{Z}}_d \end{bmatrix}$$

where $\mathbf{J}_r(\mathbf{q})\dot{\mathbf{q}} = 0$ represents the non-holonomic rover constraint, $\mathbf{J}_m(\mathbf{q})\dot{\mathbf{q}} = \dot{\mathbf{X}}_d$ is the holonomic manipulator constraint and $\mathbf{J}_c(\mathbf{q})\dot{\mathbf{q}} = \dot{\mathbf{Z}}_d$ is the additional task variables in velocity form. Suggestions of additional task variables, $z(\mathbf{q})$, are given, such as controlling the tool orientation and manipulator elbow angle. The inverse kinematics can now be solved as

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}\dot{\mathbf{X}}_d$$

To correct for task-space trajectory drift that can occur, Seraji modifies $\dot{\mathbf{q}}$ by subtracting the actual configuration vector, $\mathbf{X}$, as

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}[\dot{\mathbf{X}}_d + \mathbf{K}(\mathbf{X}_d - \mathbf{X})]$$

2.2.2 Optimization of redundancy parameters

Ancona (2017) presents a general approach for redundancy modeling based on optimization of redundancy parameters. The paper considers a nine-DOF mobile manipulator composed

of a six-DOF manipulator mounted on a three-DOF omnidirectional mobile platform. Ancona suggests a closed inverse kinematics formulation

$$\begin{bmatrix} \mathbf{q}_a \\ \mathbf{q}_b \end{bmatrix} = \begin{bmatrix} I_a(\mathbf{P}_{ee}) \\ I(\boldsymbol{\rho}) \end{bmatrix}$$

where $\mathbf{q}_a$ is the arm joints vector, $\mathbf{P}_{ee}$ is the end effector pose and I_a is a known, numeric or analytic, inverse kinematics solution for the manipulator. $\mathbf{q}_b$ represents the position and orientation of the base and relies on the redundancy parameters $\boldsymbol{\rho} = [\rho_1, \rho_2, \rho_3]$. The three redundancy parameters describe the angular displacement between the mobile platform and the robot arm, the robotic arm extension, and the angular displacement between the tool approach direction and the arm, respectively (Ancona, 2017).

Determining fitting values for the redundancy parameters is achieved by solving the general optimization problem:

$$\begin{aligned} \max_{\boldsymbol{\rho}} \quad & g(\mathbf{P_{ee}}, \boldsymbol{\rho}) \\ \text{s.t} \quad & c(\mathbf{P_{ee}}, \boldsymbol{\rho}) \leq 0 \end{aligned}$$

where $g(\mathbf{P}_{ee}, \boldsymbol{\rho})$ characterize the desired redundant behaviour and $c(\mathbf{P}_{ee}, \boldsymbol{\rho})$ is the physical joint limits of the manipulator. Ancona suggests designing the objective function as the product of weighted metric functions, g_i, describing a desired redundant behavior.

$$\begin{aligned} g(\mathbf{P}_{ee}, \boldsymbol{\rho}) &= \prod_{i=1}^{n_g} g_i^{\gamma_i}(\mathbf{P}_{ee}, \boldsymbol{\rho}) \\ \text{s.t} \quad 0 < g_i(\mathbf{P}_{ee}) < 1 \quad \forall \quad 1 \leq I \leq n_g \\ \gamma_i > 0 \quad \forall \quad 1 \leq I \leq n_g \end{aligned}$$

He proposes the following traits as desired behaviors: 1) increased dexterity, evaluated by the manipulability index, 2) obstacle avoidance, realized with sensor inputs and the Kineostatic Danger Field potential function, 3) improved end effector stability, obtained by minimizing the mobile platform movements.

2.2.3 Pose control with three operation modes based on the weighted pseudoinverse

(Sorour et al., 2019) concerns redundancy resolution on a steerable wheeled mobile robot (SWMR). The SWMR base lacks a direct kinematic mapping between its configuration- and task space, leading to low precision in motion control of the base. To address the issue of low accuracy in the base, Sorour et al. (2019) developed a redundancy resolution algorithm that alternates responsibility between the arm and the base for completing the motion task. The task sharing is organized in three different modes depending on the robot's distance from the desired pose (Figure 2.2). The three modes are

- Gross motion: Only the base contributes to the main task while the arm is kept in a predetermined, flexible pose.

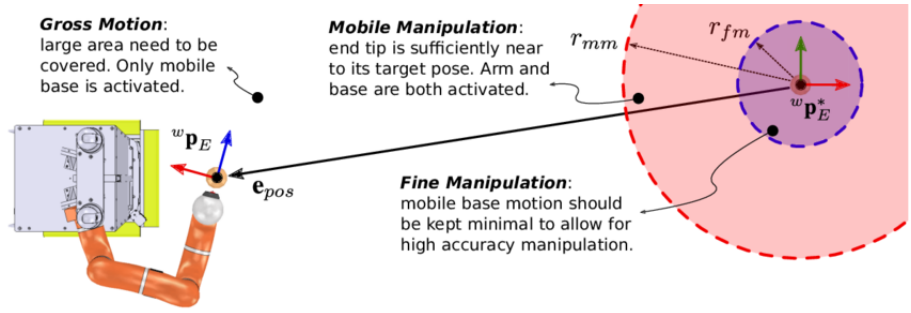


Figure 2.2: Illustration of the workspace of the three operating modes.
Image source: (Sorour et al., 2019) Copyright © 2019, IEEE

- Mobile manipulation: Both the arm and the base are responsible for the robot's motion.
- Fine manipulation: The manipulator is the main participator in the motion task. The motion of the mobile base is minimal.

The joint velocity vector of the SWMR is given as

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{q}}_b \\ \dot{\mathbf{q}}_a \end{bmatrix}$$

where $\dot{\mathbf{q}}_b \in \mathbb{R}^{n_b}$ is the joint velocity of the base and $\dot{\mathbf{q}}_a \in \mathbb{R}^{n_a}$ is the joint velocity of the robot arm. The joint space velocity $\dot{\mathbf{q}}_{ref}$ maps to the task space velocity $\dot{\mathbf{p}}_{ref}$ through the Moore-Penrose pseudoinverse of the mobile manipulator Jacobian, $\mathbf{J}_{mm}^{\dagger}$, as

$$\dot{\mathbf{q}}_{ref} = \mathbf{J}_{mm}^{\dagger}\dot{\mathbf{p}}_{ref}$$

Since this solution always results in simultaneous arm and base motion, a task-sharing redundancy resolution was proposed, formulated as

$$\dot{\mathbf{q}}_{ref} = \mathbf{J}_{mm(d)}^{\dagger W}\dot{\mathbf{p}}_{ref} + \lambda_{dex}(\mathbf{I}_{6+n_a} - \mathbf{J}_{mm}^{\dagger}\mathbf{J}_{mm})\mathbf{z}_{dex},$$

with $\mathbf{z}_{dex}$ as an arbitrary optimization term and $\mathbf{J}_{mm(d)}^{\dagger W}$ as the weighted, damped pseudoinverse of $\mathbf{J}_{mm}$ given by

$$\mathbf{J}_{mm(d)}^{\dagger W} = \mathbf{W}^{-1}\mathbf{J}_{mm}^{T}[\mathbf{J}_{mm}\mathbf{W}^{-1}\mathbf{J}_{mm}^{T} + \rho^2\mathbf{I}_{n_b}]^{-1}$$

where ρ is the damping factor and $\mathbf{W}$ is the weight matrix

$$\mathbf{W} = \begin{bmatrix} (1-\psi)\mathbf{I}_{n_b} & \mathbf{0}_{n_b \times n_a} \\ \mathbf{0}_{n_b \times n_a} & \psi\mathbf{I}_{n_a} \end{bmatrix}.$$

$\psi \in [0, 1]$ is the task sharing factor. The parameter is set to either 1 (fine manipulation), 0.5 (mobile manipulation) or 0 (gross manipulation) based on the distance from the current to the desired pose as illustrated in Figure 2.2.

The optimization term $\mathbf{z}_{dex}$ was defined as

$$\mathbf{z}_{dex} = \begin{bmatrix} \mathbf{0}_{n_b \times 1} \\ -(\mathbf{q}_a - \mathbf{q}_{a(dex)}) \end{bmatrix}$$

This choice of $\mathbf{z}_{dex}$ forces the current arm pose towards a predetermined dexterous pose, $\mathbf{q}_{a(dex)}$, as a secondary task (Sorour et al., 2019).

2.2.4 Inverse kinematics of mobile manipulators with metaheuristic algorithms

In (Lopez-Franco et al., 2018), the authors aimed to develop a redundancy strategy that avoided singularities and was generalizable for a range of mobile manipulators. The proposed solution formulated the inverse kinematics as an optimization problem and solved it with various metaheuristic algorithms where the population-based algorithm "Differential evolution" showed the best results. The cost function was formulated as

$$f'(\mathbf{q}_{\text{old}}, \hat{\mathbf{q}}) = f(\mathbf{q}_{\text{old}}, \hat{\mathbf{q}}) + \gamma \sum_{j=0}^{n} g(\hat{\mathbf{q}}_j), \tag{2.1}$$

where $f(\mathbf{q}_{\text{old}}, \hat{\mathbf{q}})$ is an error function taking the weighted sum of deviation from the desired pose, T_{error} and deviation from the current joint configuration, q_{error}

$$f(\mathbf{q}_{\text{old}}, \hat{\mathbf{q}}) = \alpha T_{\text{error}} + \beta q_{\text{error}} \tag{2.2}$$

The term $\gamma \sum_{j=0}^{n} g(\hat{\mathbf{q}}_j)$ is a penalty function ensuring the solution stays within the physical joint limits. This formulation avoided singularities by only using the forward kinematics in the error function. The mobile base was modeled as two virtual joints for differential drive platforms and three virtual joints for omnidirectional mobile platforms. Modeling the platform with virtual joints enabled the forward kinematics to be solved similarly to a fixed base manipulator with the Denavit-Hartenberg convention.

Chapter 3

Theory

This chapter presents a small selection of theory from the fields of robotics, geometry, kinematics and optimization theory needed to follow the derivation of the proposed redundancy strategies derived in Chapter 5. Additionally, three fundamental redundancy resolution methods are presented, and important concepts in redundant inverse kinematics, such as the Jacobian and singularities, are discussed.

Section 3.2.1, 3.3, 3.5 and 3.7 are derived from the specialization project (Grøtterud, 2023).

3.1 Mathematical modeling of robots

3.1.1 Configuration space and degrees of freedom

A robot manipulator is modeled as an open kinematic chain consisting of robot links connected by typically revolute and prismatic joints. The specific location of all the points of the manipulator is called a configuration. The set of all possible configurations for the robot is known as the configuration space. The links of the robot are typically assumed to be rigid bodies and the base of the manipulator is assumed fixed. Therfore, a configuration of a robot with n joints is represented only by the joint variables $\mathbf{q} = [q_1, q_2, ..., q_n]$ (Spong et al., 2019).

The minimal number needed to specify a robot's configuration is known as the robot's degrees of freedom (DOF). A manipulator's DOF is determined by its number and types of joints and equals the configuration space. A rigid body in space is defined by 3 position parameters and three rotation parameters, resulting in six DOF. Therefore, a manipulator needs at least six DOF to reach an arbitrary pose in three-dimensional space (Spong et al., 2019).

3.1.2 Task- and workspace

The workspace is the set of all the points reachable by the end effector (Spong et al., 2019). It is determined by the robot's mechanical structure and joint limits. The task space is a subset of the workspace and defines the space where the robot performs its primary tasks. The task space and workspace are usually described in cartesian coordinates and have dimensions smaller than or equal to the configuration space.

3.1.3 Representing position and rotation

The homogeneous transformation matrix

For a manipulator operating in a six-dimensional task space, a task is typically to move the end effector to a desired pose. A pose is commonly represented as a coordinate frame with a specific position and orientation in space. The homogenous transformation matrix

$$\mathbf{T}_{ab} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3), \text{ with } \mathbf{R} \in SO(3), \ \mathbf{p} \in \mathbb{R}^3 \tag{3.1}$$

describes the orientation and position of frame b relative to frame a (Egeland and Gravdahl, 2003). In addition to representing the configuration of a frame, the homogenous transformation matrix (hereby referred to only as transformation matrix) can also be used to change the reference frame in which a frame is represented (Lynch and Park, 2017):

$$\mathbf{T}_{ac} = \mathbf{T}_{ab}\mathbf{T}_{bc}. \tag{3.2}$$

Another useful property of the transformation matrix is that its inverse is the same as its transpose and results in an opposite change of reference frame

$$\mathbf{T}_{ab}^{-1} = \mathbf{T}_{ab}^{T} = \mathbf{T}_{ba} \tag{3.3}$$

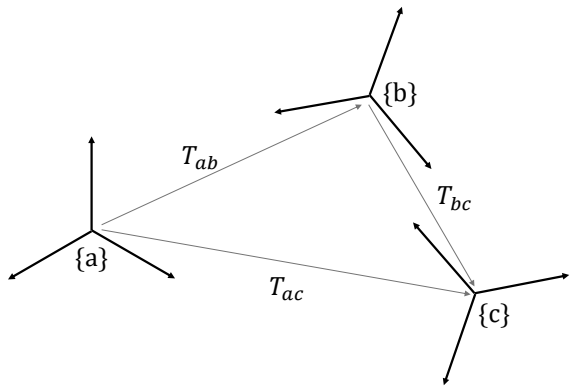


Figure 3.1: Changing reference frame of a configuration.

XYZ-fixed angles

In this thesis, the desired task is sometimes represented as the 6-dimensional task vector,

$$\mathbf{t} = [x_t, y_t, z_t, \phi_t, \theta_t, \psi_t]^T \tag{3.4}$$

where (x_t, y_t, z_t) is the position of the end effector and the angles $(\phi_t, \theta_t, \psi_t)$ represent the orientation of the end effector with the XYZ-fixed angles convention. This convention corresponds to a rotation ψ_t about the fixed x axis, θ_t about a fixed y axis and ϕ_t about the fixed z axis, resulting in the equivalent rotation matrix (Waldron and Schmiedeler, 2016)

$$\mathbf{R}(\phi, \theta, \psi) = \begin{bmatrix} \cos\phi\cos\theta & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\phi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi \\ -\sin\theta & \cos\theta\sin\psi & \cos\theta\cos\psi \end{bmatrix} \tag{3.5}$$

Quaternions

Quaternions provide a compact way of describing rotations as a normalized vector with four scalars, and are used in this thesis to compare rotations. Defining a rotation as a single rotation θ about the unit vector $\mathbf{v} = [v_x, v_y, v_z]$, the corresponding unit quaternion is

$$Q = [w, x, y, z] = [cos(\frac{\theta}{2}), v_x sin(\frac{\theta}{2}), v_y sin(\frac{\theta}{2}), v_z sin(\frac{\theta}{2})] \tag{3.6}$$

where Q satisfies $||Q|| = 1$ (Kuffner, 2004). The inner product of two unit quaternions

$$\lambda = Q_1 \cdot Q_2 = w_1 w_2 + x_1 x_2 + y_1 y_2 + z_1 z_2 \tag{3.7}$$

range from $[-1, 1]$ depending on the distance between the quaternions. In this thesis, a rotation distance component Q_{err}, suggested by Kuffner (2004),

$$Q_{err} = 1 - ||\lambda|| \tag{3.8}$$

is used as a distance metric when evaluating the similarity of two configurations. The smaller $Q_{err} \in [0, 1]$ is the smaller the distance between the two rotations.

3.2 Forward kinematics

A robot's forward kinematics refers to the process of calculating the pose of its end-effector frame given its joint coordinates $\mathbf{q}$. The forward kinematics can be described in terms of a nonlinear vector function, f, mapping from joint space to task space,

$$\mathbf{t} = f(\mathbf{q}) \tag{3.9}$$

Here, $\mathbf{q} = [\theta_1, ..., \theta_n]$ is the joint angle vector and $\mathbf{t} = [\mathbf{p}, \mathbf{r}]$ is the task vector representing the position and orientation of the end effector in the task space. The forward kinematics

can also be represented as a transformation matrix from the base of the arm, $\{a\}$, to the end effector, commonly denoted tool frame, $\{t\}$,

$$\mathbf{T}_{at}(\mathbf{q}) = \mathbf{T}_{a1}(\theta_1)\mathbf{T}_{12}(\theta_2)\mathbf{T}_{23}(\theta_3)...\mathbf{T}_{nt}(\theta_n). \tag{3.10}$$

Here, $\mathbf{T}_{at}$ is the product of all the consecutive link transformation matrices of the manipulator.

Several different schemes exist for computing the forward kinematics of open-chain robots. The most common representation is the Denavit-Hartenberg parameters (DH-parameters), which use a minimal set of four parameters to describe the transformation between link frames (Hartenberg and Denavit, 1964). Another common method for kinematic modeling is the Product of Exponentials (PoE), which builds on geometrical concepts like screws, twists and wrenches (Lynch and Park, 2017).

This thesis uses both the PoE and the DH-parameters to model Lio's forward kinematics. During the implementation of the kinematic model in the specialization project (Grøtterud, 2023), the PoE method was chosen because it is accompanied by the open-source software package, Modern Robotics (Weng and Lynch, 2018). The development of the optimization-based methods (Section 5.2) required a symbolic expression of the forward kinematics, which was derived with DH-paramters.

3.2.1 Product of Exponentials formula

The key concept of the PoE formula is to regard each joint as applying a screw motion to all the outward links of the robot (Lynch and Park, 2017). When analyzing the screw motion induced by each joint, there is no requirement to establish reference frames for all links, as is necessary with DH-parameters. The only frames that need to be defined are the space frame, $\{s\}$, chosen at a fixed point in space, and the tool frame $\{t\}$, chosen at the tool center point (TCP) of the end effector.

The transformation matrix, $\mathbf{T}_{st}(\boldsymbol{\theta})$, describes the end-effector configuration relative to the space frame. When all joints are in their zero position, the transformation is called the home configuration, $\mathbf{M}$, of the robot,

$$\mathbf{T}_{st}(\mathbf{0}) = \mathbf{M} \in SE(3) \tag{3.11}$$

Figure 3.2 shows an n-link robot where joint n is displaced to a joint value θ_n. The screw motion corresponding to rotating about joint n can be expressed in the $\{s\}$ frame as the screw axes S_n,

$$S_n = [\boldsymbol{\omega}_n \quad \boldsymbol{v}_n] \tag{3.12}$$

If joint n is a revolute joint, then $\boldsymbol{\omega}_n$ is a unit vector in the positive direction of joint axis n. $\mathbf{v}_n = -\boldsymbol{\omega}_n \times \mathbf{q}_n$ where $\mathbf{q}_n$ is a point on joint axis n written in the coordinates of the fixed frame $\{s\}$. If joint n is prismatic, then $\boldsymbol{\omega}_n = 0$ and $\mathbf{v}_n$ is a unit vector in the direction of positive translation.

Assume that the joints $\{1, ..., n\}$ are displaced to some joint angles $\{\theta_1, ..., \theta_n\}$. The end-effector frame then undergoes a displacement from the home configuration, $\mathbf{M}$, and the new pose can be described by the transform

$$\mathbf{T}_{st} = e^{[S_1]\theta_1} \ ... \ e^{[S_{n-1}]\theta_{n-1}} e^{[S_n]\theta_n} \mathbf{M} \tag{3.13}$$

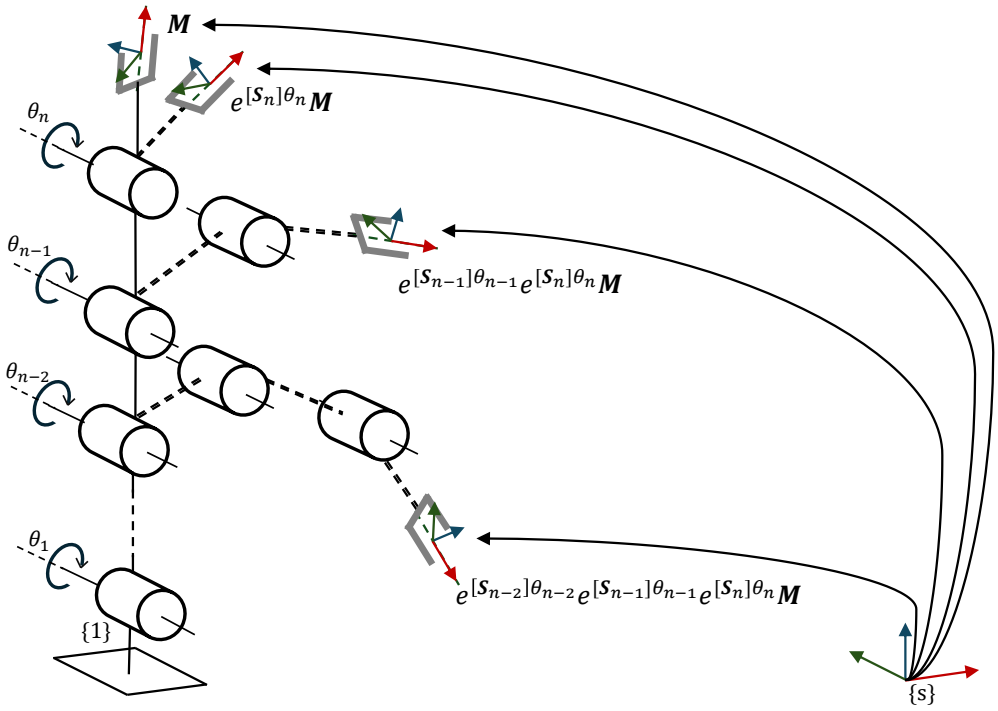


Figure 3.2: PoE formula for an n-link robot arm.

3.2.2 Denavit-Hartenberg convention

The Denavit-Hartenberg convention enables a transformation between two links to be described with a minimal set of four parameters: the link length, r_i, the joint twist, α_i, the link offset, d_i, and the joint angle θ_i. This representation is achieved through a set of rules determining the location of the link frame origins and the axis of rotation (Waldron and Schmiedeler, 2016). Various forms of the convention for locating coordinate frames exist, this thesis employs the classic (distal) DH-parameters.

When assigning coordinate frames the following rules must be applied

1. Axis Z_i must be either the axis of rotation or the direction of motion of $joint_i$, depending on if $joint_i$ is a revolute or prismatic joint, respectively.
2. Axis X_i must be perpendicular to both the Z_i- and the Z_{i-1}-axis, and intersect with axis Z_{i-1}.

3. Axis Y_i must be determined according to the right-hand rule.

Once the link frame locations are established, the four DH-parameters can be defined as

r_i: the distance from Z_{i-1} to Z_i along X_i.

α_i: the angle from Z_{i-1} to Z_i about X_i.

d_i: the distance from X_{i-1} to X_i along Z_{i-1}.

θ_i: the angle from X_{i-1} to X_i about Z_{i-1}.

With this convention, the link frame i can be determined relative to frame $i-1$ through a series of transformations: a rotation θ_i about axis Z_{i-1}, a displacement d_i along axis Z_{i-1}, a displacement r_i along axis X_i and a rotation α_i along axis X_i. Multiplying the individual transformation matrices

$$\mathbf{Rot}_{Z_{i-1}}(\theta_i)\mathbf{Trans}_{Z_{i-1}}(d_i)\mathbf{Trans}_X(r_i)\mathbf{Rot}_X(\alpha_i) \tag{3.14}$$

the homogeneous transformation matrix describing the transformation from frame $i-1$ to frame i is defined as

$$\mathbf{T}_{i-1,i} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & 0 \\ sin(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & r_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_i) & -sin(\alpha_i) & 0 \\ 0 & sin(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos(\theta_i) & -sin(\theta_i)cos(\alpha_i) & sin(\theta_i)sin(\alpha_i) & r_i cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)cos(\alpha_i) & -cos(\theta_i)sin(\alpha_i) & r_i sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

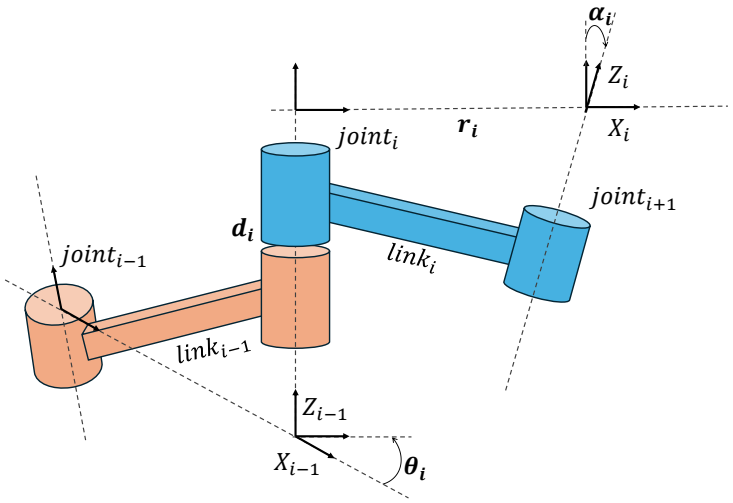


Figure 3.3: The four DH-parameters following the classic convention.

The overall transformation from the base frame to the end effector frame of a robot with n joints can then be described as

$$\mathbf{T}_{0n} = \mathbf{T}_{01}\mathbf{T}_{12} \ ... \ \mathbf{T}_{n-1,n} \tag{3.16}$$

3.3 Inverse kinematics

The objective of the inverse kinematics problem is to find the joint angles that yield a specific Cartesian position and orientation of the end effector (Nakamura, 1990). Assuming a known forward kinematics mapping, $\mathbf{T}_{st}(\boldsymbol{\theta}) \in SE(3)$, the inverse kinematics problem is to find the θs that solves

$$\mathbf{T}_{st}(\boldsymbol{\theta}) = \mathbf{X} \tag{3.17}$$

where $\mathbf{X} \in SE(3)$ is a desired end effector pose. The inverse kinematic problem can also be formulated as the inverse of 3.9 as

$$\mathbf{q} = f^{-1}(\mathbf{t}) \tag{3.18}$$

If

$$dim(\mathbf{q}) = n \quad \leq \quad dim(\mathbf{t}) = m,$$

a finite set of unique solutions exists. Pieper (1969) proved that a general 6R open-chain robot can have up to 16 solutions. Solving the inverse kinematic problem generally involves advanced nonlinear algebraic computations. However, an analytic solution exists for 6R robots with a spherical wrist (Nakamura, 1990). The analytic inverse kinematics solution to Lio's six-DOF manipulator is derived in chapter 4.2 and is therefore not included in this section. If $n > m$, the manipulator is said to be kinematically redundant with degree $r = n - m$. The different methods for solving the redundant inverse kinematic problem are called redundancy resolution and were covered in Section 3.5.

3.4 Task Jacobian and geometric Jacobian

Evaluating the first-order differential kinematics introduces an important concept in kinematic modeling, the Jacobian matrix. Differentiating equation (3.9) results in the first order forward kinematics

$$\dot{\mathbf{t}} = \mathbf{J}_t(\mathbf{q})\dot{\mathbf{q}} \tag{3.19}$$

where $\dot{\mathbf{t}}$ is the task space velocity vector and $\dot{\mathbf{q}}$ is the joint space velocity vector and

$$\mathbf{J}_t = \frac{\partial f(\mathbf{q})}{\partial \mathbf{q}} \tag{3.20}$$

is the $(m \times n)$ *task* Jacobian or *analytic* Jacobian (Chiaverini et al., 2016). Here, $\dot{\mathbf{t}}$ does not represent the velocity of the end effector, but rather the rate of change of the minimal representation of the end effector orientation.

The actual velocity of the end effector can be described with the spatial velocity vector

$$\boldsymbol{\nu} = [\mathbf{v}, \boldsymbol{\omega}]^T \tag{3.21}$$

where $\mathbf{v}$ is the translational velocity vector and $\boldsymbol{\omega}$ is the angular velocity vector of the end effector. The spatial velocity vector can be related to the task space velocities $\dot{\mathbf{t}}$ as

$$\dot{\mathbf{t}} = \mathbf{T}(\mathbf{t})\boldsymbol{\nu} \tag{3.22}$$

where $\mathbf{T}(\mathbf{t})$ is a $(M \times 6)$ transformation matrix depending on $\mathbf{t}$ (**?**).

The mapping from joint velocities to actual end-effector velocities can now be defined by the *geometric* Jacobian, $\mathbf{J}$

$$\boldsymbol{\nu} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{3.23}$$

By combining the equations (3.19), (3.22) and (3.4) the relation between the geometric Jacobiand and the analytic Jacobian can be described as

$$\mathbf{J}_t = \mathbf{T}(\mathbf{t})\mathbf{J} \tag{3.24}$$

3.4.1 Singularities

Equation 3.19 indicates that the task space velocities, $\dot{\mathbf{t}}$ are a linear combination of the columns of the task Jacobian, $\mathbf{J}_t$

$$\dot{\mathbf{t}} = \mathbf{J}_{t1}\dot{q}_1 + \mathbf{J}_{t2}\dot{q}_2 + \ ... \ + \mathbf{J}_{t6}\dot{q}_6 \tag{3.25}$$

From this expression, it is easy to see that $\mathbf{J}_t$ needs to have six linearly independent columns for the end effector to be able to generate any arbitrary velocity (Spong et al., 2019). If $\mathbf{J}_t$ has less than $n = 6$ linearly independent columns, in other words, $\mathbf{J}_t$ is rank deficient, the corresponding configuration $\mathbf{q}$ is called singular.

Looking at the relation between the task Jacobian and the geometric Jacobian (3.24) it becomes clear that a singularity $\mathbf{J}_t$ can be caused by rank-deficiency in both the transformation matrix $\mathbf{T}$ and/or the geometric Jacobian $\mathbf{J}$. Rank deficiency in the two matrices gives rise to two types of singularities: *representation singularities*, for rank deficiency of $\mathbf{T}$, and *kinematic singularities*, for rank deficiency of $\mathbf{J}$ (Chiaverini et al., 2016). The representation singularity is related to the mathematical representation of the end effector orientation, while the kinematic singularities are directly related to infeasible end effector velocities.

Avoiding singular configurations is desirable for numerous reasons. First, singularities represent configurations where it is impossible to generate end-effector velocities in certain directions, making the robot unable to solve the given task. Moreover, bounded end-effector velocities may lead to unbounded joint velocities, resulting in control issues and high mechanical stress on the robot's components.

3.4.2 Manipulability

A singularity is a binary parameter - a configuration is either singular or not. However, the effect of singularities can also be present in nonsingular configurations "close" to a singularity. The manipulability ellipsoid is a geometrical property describing the directions in which the end-effector's ability to move is diminished (Lynch and Park, 2017). A common metric used to characterize the distance from a singularity is the manipulability measure (Yoshikawa, 1985)

$$\mu = \sqrt{det(\mathbf{J}_t \mathbf{J}_t^T)} \tag{3.26}$$

where μ corresponds to the product of the lengths of the manipulability ellipsoid's principal semi-axis.

3.5 Redundancy resolution

The different methods for solving the inverse kinematics problem of redundant manipulators are called redundancy resolution. The redundancy problem can be established at the position, velocity or acceleration level. In this thesis, the inverse kinematics is solved at the position level to exploit the analytic solution of the manipulator and avoid dealing with drift that can occur from differential kinematic schemes. However, most of the fundamental redundancy resolution methods in the literature are described at the velocity level in order to include discussion about the Jacobian (Chiaverini et al., 2016). Hence, the redundancy resolution methods in this section are described in terms of the first-order kinematics.

3.5.1 Analysis of redundancy

The task Jacobian $\mathbf{J}_t$ (3.19) can be viewed as a linear transformation mapping the vector $\dot{\mathbf{q}} \in \mathbb{R}^n$ into $\dot{\mathbf{t}} \in \mathbb{R}^m$. The null space, $\aleph(\mathbf{J}_t)$, and the range space, $\Re(\mathbf{J}_t)$, shown in Figure 3.4, are fundamental subspaces associated with the linear mapping (Patel and Shadpey, 2005).

$$\aleph(\mathbf{J}_t) = \{\dot{\mathbf{q}} \in \mathbb{R}^n \quad | \quad \mathbf{J}_t(\mathbf{q})\dot{\mathbf{q}} = 0\} \tag{3.27}$$

$$\Re(\mathbf{J}_t) = \{\dot{\mathbf{q}} \in \mathbb{R}^n \quad | \quad \dot{\mathbf{q}} \in \mathbb{R}^n\} \tag{3.28}$$

The range space of $\mathbf{J}_t$ represents the subspace of task velocities that the joint velocities can generate. The null space of $\mathbf{J}_t$ represents the subspace of joint velocities that does not affect the task space velocities. If the Jacobian matrix $\mathbf{J}_t(\mathbf{q})$ has full column rank, the dimension of the null space, $\aleph(\mathbf{J}_t)$, is equal to the degree of redundancy, r. If the Jacobian has a rank of $m' < m$, the dimension of the null space increases to $n - m'$ (Fahimi, 2009). The existence of the null space is what causes the infinite solutions to the inverse kinematics problem that characterizes redundant manipulators.

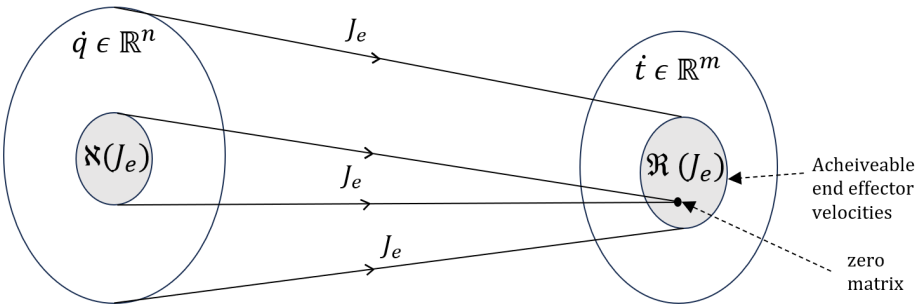


Figure 3.4: Mapping between joint velocity and end effector velocity.

3.5.2 Pseudoinverse Jacobian

On the assumption that $n \geq m$, the exact solution to the differential inverse kinematics problem relies on the pseudoinverse of $\mathbf{J}_t$, denoted, $\mathbf{J}_t^{\dagger}$.

$$\dot{\mathbf{q}} = \mathbf{J}_t^{\dagger}\dot{\mathbf{t}} \tag{3.29}$$

The pseudoinverse provides the least squares solution to 3.19 as

$$\min_{\dot{\mathbf{q}}} \quad ||\dot{\mathbf{t}} - \mathbf{J}_t\dot{\mathbf{q}}|| \tag{3.30}$$

The main advantage of the pseudoinverse Jacobian is that it provides a meaningful solution to 3.19 regardless of whether the analytic Jacobian, $\mathbf{J}_t$, is rectangular or square (Patel and Shadpey, 2005). However, the solution does not guarantee avoiding singular configurations. Singular configurations are related to rank deficiencies of $\mathbf{J}_t$ and result in cases where certain end effector velocities cannot be generated with joint velocities commands (Chiaverini et al., 2016). Another drawback with the pseudoinverse solution is that it does not allow the robot to satisfy user-specified tasks and exploit its additional DOF.

3.5.3 Null-space projection

One method for utilizing the extra DOF is to add velocities belonging to the null space of $\mathbf{J}_t$, $\dot{\mathbf{q}}_{\aleph}$.

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_p + \dot{\mathbf{q}}_{\aleph} \tag{3.31}$$

Here, the subscript p to $\dot{\mathbf{q}}_p$, indicates that it is the *primary* solution to a desired task $\dot{\mathbf{t}}_d$,

$$\dot{\mathbf{t}}_d = \mathbf{J}_t\dot{\mathbf{q}}_p \tag{3.32}$$

From the definition of the null space (3.27), it is evident that additional nullspace velocities do not affect the desired task because

$$\mathbf{J}_t(\dot{\mathbf{q}}_d + \dot{\mathbf{q}}_{\aleph}) = \mathbf{J}_t\dot{\mathbf{q}}_d + 0 = \dot{\mathbf{t}}_d \tag{3.33}$$

Instead, $\dot{\mathbf{q}}_{\aleph}$ generates internal joint motions. These internal motions can be utilized to improve dexterity and satisfy additional requirements, like obstacle avoidance, without disturbing the primary task.

The null space velocity vector, $\dot{\mathbf{q}}_{\aleph}$, can be obtained by projecting a vector $\boldsymbol{\nu}$ to the null space of the Jacobian

$$\dot{\mathbf{q}}_{\aleph} = (\mathbf{I} - \mathbf{J}_t^{\dagger}\mathbf{J}_t)\boldsymbol{\nu} \tag{3.34}$$

Using the standard Project Gradient method, ν can be selected as

$$\boldsymbol{\nu} = -\alpha\nabla\Phi(\mathbf{q}) = -\alpha[\frac{\partial\Phi}{\partial q_1} \quad ... \quad ... \quad \frac{\partial\Phi}{\partial q_n}]^T \tag{3.35}$$

where $\Phi(\mathbf{q})$ is a cost function whose optimal value ensures the desired additional tasks and $\alpha > 0$ is a scalar representing a step in the gradient direction. A drawback with the Project Gradient method is that the objective function defining the secondary tasks needs to be differentiable and it generally does not guarantee the convergence of the solution (Ancona, 2017).

3.5.4 Extended task space

Another approach to the redundant inverse kinematics problem is known as extended task space (Egeland and Balchen, 1987). With this method, the task vector is extended to include r number of secondary tasks to be fulfilled along with the original end-effector task. The new task vector is denoted the *augmented* task vector, $\mathbf{t}_a$,

$$\mathbf{t}_a = \begin{bmatrix} \mathbf{t} \\ \mathbf{t}_c \end{bmatrix} \tag{3.36}$$

with $\mathbf{t}_c$ as the vector for secondary tasks, called the *constraint-task* vector. The differential kinematics equation can now be defined as

$$\dot{\mathbf{t}}_a = \mathbf{J}_a(\mathbf{q})\dot{\mathbf{q}} \tag{3.37}$$

where the matrix

$$\mathbf{J}_a = \begin{bmatrix} \mathbf{J}_t \\ \mathbf{J}_c \end{bmatrix} \tag{3.38}$$

is known as the *augmented* Jacobian. $\mathbf{J}_c$ is the Jacobian related to the constraint task vector and is of dimension $(r \times n)$, which implies that $\mathbf{J}_a$ has dimension $(n \times n)$ and the differential inverse kinematics formulation

$$\dot{\mathbf{q}} = \mathbf{J}_a^{-1}\dot{\mathbf{t}}_a \tag{3.39}$$

is on closed form. The additional tasks in $\mathbf{t}_c$ are often represented as inequality constraints or as parts of a kinematic cost function (Ancona, 2017). The additional tasks can be chosen to specify desired kinematic characteristics such as posture control, joint limits or obstacle avoidance.

3.6 General optimization problem

A general optimization problem is formulated as

$$\begin{aligned} \min_{\mathbf{x}\in R^n} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & c_i(\mathbf{x}) = 0, \quad i \in \mathcal{E} \\ & c_i(\mathbf{x}) \geq 0, \quad i \in \mathcal{I} \end{aligned} \tag{3.40}$$

where $f(\mathbf{x})$ is the objective function, $\mathbf{x}$ is the optimization variables or decision variables and $c_i(\mathbf{x})$ is the constraint function subject to the space of equality constraints, $\mathcal{E}$, or inequality constraints, $\mathcal{I}$ (Jorge Nocedal, 2006). The goal of an optimization problem is to find the variables $\mathbf{x}$ that minimize the objective function, f, which can be selected as any quantitative measure of a system. If the objective function f and/or the constraint function c_i are non-linear, the optimization problem becomes non-linear. A non-linear optimization problem is commonly abbreviated as NLP.

3.6.1 Numerical inverse kinematics as an optimization problem

Numerical methods for solving the inverse kinematics problem are useful if a closed-form solution doesn't exist or the manipulator is redundant (Spong et al., 2019). An intuitive approach is to iterate over a range of joint values and minimize the error between the desired end effector pose and the output of the forward kinematics. This method can be formulated as a nonlinear optimization problem on the form

$$\begin{aligned} \min_{\mathbf{q}} \quad & \frac{1}{2}(f(\mathbf{q}) - \mathbf{t})^T (f(\mathbf{q}) - \mathbf{t}) \\ \text{s.t.} \quad & \mathbf{q}_{lower} \leq \mathbf{q} \leq \mathbf{q}_{upper} \end{aligned} \tag{3.41}$$

where $\mathbf{t}$ is the task vector representing the desired end effector pose and $f(\mathbf{q})$ is the forward kinematics function. The inequality constraint $\mathbf{q}_{lower} \leq \mathbf{q} \leq \mathbf{q}_{upper}$ ensures that the solution stays with the robot's physical joint limits.

3.7 Differential drive systems

Lio's platform has a differential drive system with two independently actuated wheels in the front and two caster wheels in the back. The configuration of a differential drive system can be expressed with the state $\mathbf{q}_p = [x_p \quad y_p \quad \theta_p]$ where (x_p, y_p) is the origin of the platform frame, $\{p\}$. The frame origin is defined at the midpoint of the axis between the two actuated front wheels (Figure 3.5). θ_p denotes the platform's heading direction and the platform frame's orientation. The pose of the platform relative to a fixed space frame, $\{s\}$, can be described by the transformation matrix

$$\mathbf{T}_{sp}(\mathbf{q}_p) = \begin{bmatrix} cos\theta_p & -sin\theta_p & 0 & x_p \\ sin\theta_p & cos\theta_p & 0 & y_p \\ 0 & 0 & 1 & z_p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.42}$$

where z_p is a constant indicating the platform frame's height above the ground. Denoting the radius of the wheels as r, and the translational and angular velocity of the left and right wheel as (v_L, v_R) and $(\boldsymbol{\omega}_L, \omega_R)$, the translational velocity of the platform $(\dot{x}_p, \dot{y}_p)$ can be described in terms of the wheel velocities as

$$v = \frac{v_L + v_R}{2}, \quad v_L = \omega_L r \quad v_R = \omega_R r \tag{3.43}$$

$$\dot{x}_p = v_x = vcos\theta_p = \frac{r}{2} cos\theta_p(\omega_L + \omega_R) \tag{3.44}$$

$$\dot{y}_p = v_x = vsin\theta_p = \frac{r}{2} sin\theta_p(\omega_L + \omega_R) \tag{3.45}$$

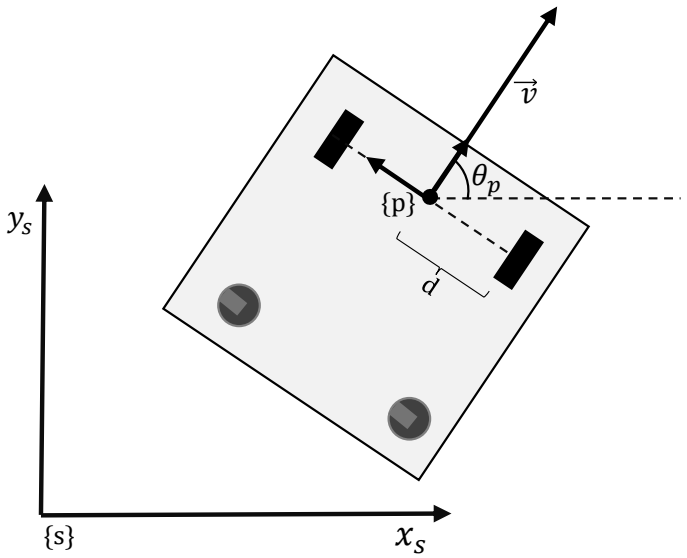


Figure 3.5: Model of differential drive system.

Denoting the distance between the wheel and the reference point as d (Figure 3.5), and defining positive θ_p against the clock, the angular velocity of the platform around the origin of frame $\{p\}$ can be expressed as

$$\dot{\theta}_p = \frac{v_R - v_L}{2d} = \frac{r}{2d}(\omega_R - \omega_L) \tag{3.46}$$

Assuming the angular velocities of each wheel are the control parameters, the differential

kinematic equations can be expressed in the canonical form

$$\dot{\mathbf{q}} = G(\mathbf{q})u = \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{\theta}_p \end{bmatrix} = \begin{bmatrix} \frac{r}{2}cos\theta_p & \frac{r}{2}cos\theta_p \\ \frac{r}{2}sin\theta_p & \frac{r}{2}sin\theta_p \\ \frac{-r}{2d} & \frac{r}{2d} \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix} \tag{3.47}$$

The canonical model (3.47) shows that there is a 2-dimensional set of velocities for a 3-DOF system. Rewriting (3.44) and (3.45) as

$$\dot{x}_p = vcos\theta_p \tag{3.48}$$

$$\dot{y}_p = vsin\theta_p \tag{3.49}$$

demonstrates that the system (3.47) is subject to the nonholonomic constraint $A(\mathbf{q})$.

$$A(\mathbf{q})\dot{\mathbf{q}} = [sin\theta_p \quad -cos\theta_p \quad 0]\dot{\mathbf{q}} = \dot{x}_p sin\theta_p - \dot{y}_p cos\theta_p = 0 \tag{3.50}$$

A nonholonomic constraint reduces the dimension of the system's achievable velocities, but does not reduce the dimension of the reachable configuration space (Lynch and Park, 2017). The constraint reflects the fact that the platform can not move sideways, but must move in the direction of its main axis of symmetry. Hence, it is important to consider this constraint when modeling the differential kinematics of a mobile manipulator.

Chapter 4

Kinematic modeling of Lio

This chapter derives Lio's forward kinematics and the inverse analytic solution of Lio's robot arm. First, the robot's forward kinematics is derived from the Product of Exponentials method. Then, the Denavit-Hartenberg parameters of the robot are presented before the analytic inverse kinematics solution of the manipulator is derived with a geometric approach.

Section 4.1 and 4.2 is derived from the specialization project (Grøtterud, 2023).

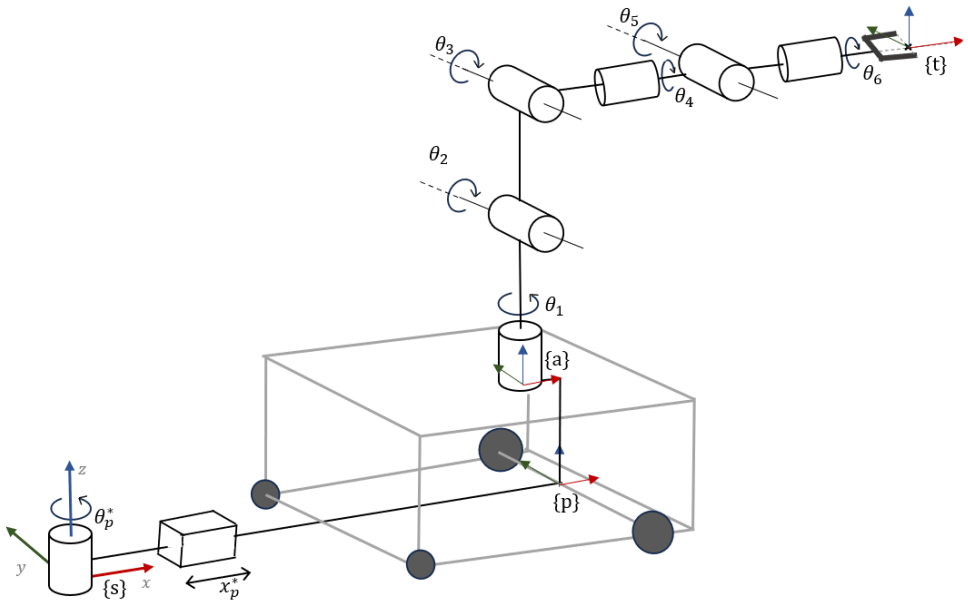


Figure 4.1: Joint figure of the mobile manipulator with marked reference frames for the platform $\{p\}$, the base of the arm $\{a\}$ and the TCP $\{t\}$.

4.1 Forward kinematics with Product of Exponentials

This section derives the forward kinematics of the mobile manipulator. First, the forward kinematics of the robot arm is derived with the PoE formula. Next, the mobile platform is modeled as a set of virtual joints, and its forward kinematics are derived. Finally, the transformation between the space frame, mobile platform and manipulator is presented to form a forward kinematics model for the mobile manipulator.

In the kinematic model of Lio, the frame at the end effector is called the tool frame, denoted as $\{t\}$. The frame of the manipulator base is referred to as the platform frame, $\{p\}$. The frame at the base of the manipulator arm is designated as $\{a\}$ (Figure 4.1). The frame at the base of the robot arm was labeled $\{a\}$ instead of the more commonly used $\{b\}$ to avoid confusion with the mobile platform, sometimes called the mobile base.

4.1.1 Forward kinematics of manipulator

To simplify the following computations, Lio's zero-configuration, $\mathbf{M}$, is defined as shown in Figure 4.2, with the end effector frame, $\{t\}$, aligned with the base frame, $\{a\}$, and a 90-degree bend in the elbow joint, θ_3. The offset from the manufacturer's zero joint state is considered in the implementation in chapter 6.3.

Using the PoE formula, the transformation from the base of the robot arm, $\{a\}$, to the TCP, $\{t\}$, can be expressed as

$$\mathbf{T}_{at} = e^{[S_1]\theta_1} e^{[S_2]\theta_2} e^{[S_3]\theta_3} e^{[S_4]\theta_4} e^{[S_5]\theta_5} e^{[S_6]\theta_6} \mathbf{M} \tag{4.1}$$

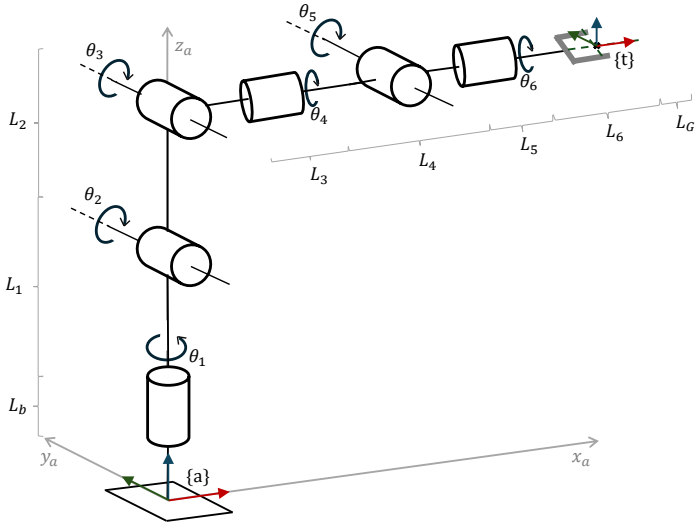


Figure 4.2: Joint representation of Lio's manipulator arm in the home configuration.

where the home configuration, $\mathbf{M}$, is

$$\mathbf{M}_a = \begin{bmatrix} 1 & 0 & 0 & L_3 + L_4 + L_5 + L_6 + L_G \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_b + L_1 + L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

and the screw axes, $S_{ai} = [\boldsymbol{\omega}_{ai} \quad \mathbf{v}_{ai}]$, is shown in Table 4.1.

Frame	$\boldsymbol{\omega}$	$\mathbf{q}$	$\mathbf{v} = -\boldsymbol{\omega} \times \mathbf{q}$
S_{a1}	$[0 \quad 0 \quad 1]$	$[0 \quad 0 \quad L_b]$	$[0 \quad 0 \quad 0]$
S_{a2}	$[0 \quad 1 \quad 0]$	$[0 \quad 0 \quad (L_b + L_1)]$	$[-(L_b + L_1) \quad 0 \quad 0]$
S_{a3}	$[0 \quad 1 \quad 0]$	$[0 \quad 0 \quad (L_b + L_1 + L_2)]$	$[-(L_b + L_1 + L_2) \quad 0 \quad 0]$
S_{a4}	$[1 \quad 0 \quad 0]$	$[L_3 \quad 0 \quad (L_b + L_1 + L_2)]$	$[0 \quad (L_b + L_1 + L_2) \quad 0]$
S_{a5}	$[0 \quad 1 \quad 0]$	$[(L_3 + L_4) \quad 0 \quad (L_b + L_1 + L_2)]$	$[-(L_b + L_1 + L_2) \quad 0 \quad (L_3 + L_4)]$
S_{a6}	$[1 \quad 0 \quad 0]$	$[(L_3 + L_4 + L_5) \quad 0 \quad (L_b + L_1 + L_2)]$	$[0 \quad (L_b + L_1 + L_2) \quad 0]$

Table 4.1: Table of components of screw axes to all robot joints

4.1.2 Modeling of the mobile platform

The movement of the mobile platform is modeled as a virtual revolute joint, θ_p^*, and a virtual prismatic joint, x_p^*, as shown in Figure 4.1. The virtual joints are denoted with a star to separate them from the actual joints of the manipulator and the pose of the platform frame represented by $[x_p, y_p, \theta_p]$. This representation takes into account the non-holonomic constraint without altering the platform's reachable workspace. The virtual joints restrict the platform to performing separate rotational and translational movements. This is a reasonable simplification as Lio typically operates with sequential rotational and translational movements.

The prismatic joint is connected to the midpoint of the front wheel axis while θ_p^* is located at the origin of the space frame. When x_p^* is zero, there is no offset between θ_p^* and x_p^*. Hence, the home configuration can be described as the identity matrix

$$\mathbf{M}_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

The Screw axes for each joint is

$$S_{pr} = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0] \tag{4.4}$$

$$S_{pp} = [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0] \tag{4.5}$$

$$\tag{4.6}$$

Using the PoE formula, the forward kinematics of the mobile platform can be described as

$$\mathbf{T}_{sp}(\theta_p^*, x_p^*) = e^{[S_{pr}]\theta_p^*} e^{[S_{pp}]x_p^*} \mathbf{M}_p \tag{4.7}$$

4.1.3 Forward kinematics of mobile manipulator

The complete forward kinematics of the mobile manipulator, $\mathbf{T}_{st}$, is simply the product of all subsequent transformation matrices from the space frame, $\{s\}$, to the tool frame, $\{t\}$.

$$\mathbf{T}_{st}(\theta_p^*, x_p^*, \theta_1, ..., \theta_6) = \mathbf{T}_{sp}(\theta_p^*, x_p^*)\mathbf{T}_{pa}\mathbf{T}_{at}(\theta_1, ..., \theta_6) \tag{4.8}$$

where $\mathbf{T}_{pa}$ is the fixed offset between the frame of the platform, $\{p\}$, and the frame at the base of the arm, $\{a\}$,

$$\mathbf{T}_{pa} = \begin{bmatrix} 1 & 0 & 0 & -113 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 265.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

with the displacement along the x and z axis given in [mm]. A joint representation of the mobile manipulator with the mentioned reference frames is depicted in Figure 4.1.

4.2 Analytic inverse kinematics of Lio arm

This chapter aims to derive the expression for the joint values of the robot arm $\theta_1, ..., \theta_6$ given a desired end effector transformation $\mathbf{X}$.The inverse kinematics of Lio's arm can be solved analytically since the last three joints have rotation axes that intersect orthogonally. In other words, θ_4, θ_5 and θ_6 form a spherical wrist with wrist center in θ_5. The position of the wrist center, $\mathbf{p}_w$, in different poses can be seen in Figure 4.3.

It can be seen from Figure 4.3 that the wrist position only depends on the first three joints while the orientation of the end effector depends on the last three joints. Therefore, the inverse kinematics problem can be decoupled into two subproblems; determining position and orientation.

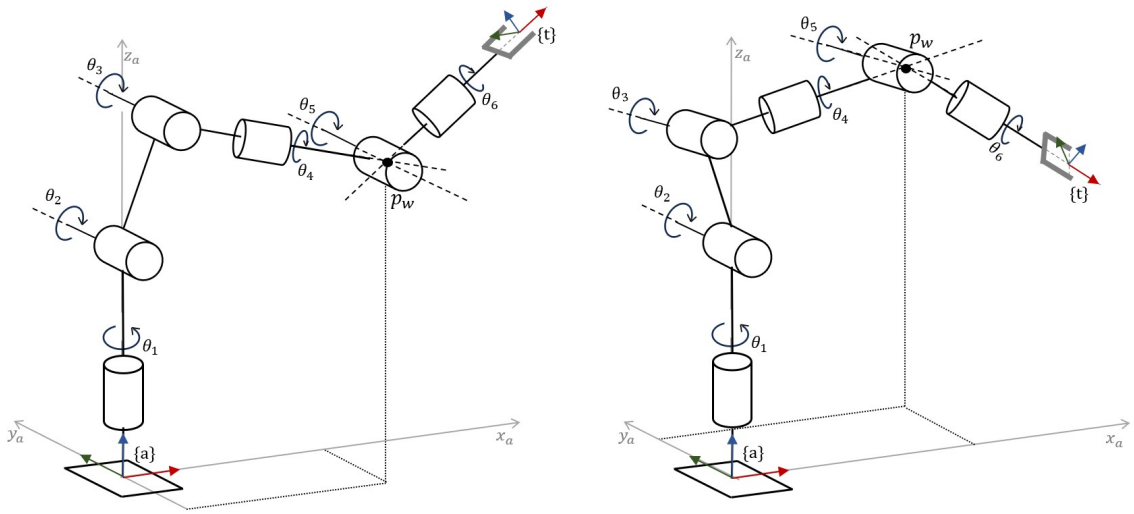


Figure 4.3: The position of the wrist center $\mathbf{p}_w$ in different poses.

4.2.1 Inverse position of wrist

Given a desired end effector pose, $\mathbf{X}$,

$$\mathbf{X} = \mathbf{T}_{at}(\theta_1, ..., \theta_6) = \begin{bmatrix} \mathbf{R}_d & \mathbf{p}_d \\ 0 & 1 \end{bmatrix} \tag{4.10}$$

the rotation and translational displacement from the base of the arm to the TCP are, $\mathbf{R}_d$, and, $\mathbf{p}_d$, respectively.

The distance from the wrist center to the tool frame in the robot's zero position is $\mathbf{l} = [L_4 + L_6 + L_g \quad 0 \quad 0]^T$. Since the orientation of the end effector only depends on the last three joints, the wrist center position $\mathbf{p}_w = [p_x \quad p_y \quad p_z]^T$ (Figure 4.3) can be found with

$$\mathbf{p}_w = \mathbf{p}_d - \mathbf{R}_d * \mathbf{l} \tag{4.11}$$

Knowing the wrist center coordinates, $\mathbf{p}_w$, θ_1 can easily be derived by evaluating Figure 4.4.

$$\theta_1 = atan2(p_y, p_x) \quad \text{or} \quad \theta_1 = atan2(p_y, p_x) + \pi \tag{4.12}$$

where the different solutions determine if it is the back or the front of the robot, respectively, that faces the end effector. The second solution, "back solution", $\theta_1 = atan2(p_y, p_x) + \pi$, is only valid when the original value for θ_2 is swapped for $\pi - \theta_2$ (Lynch and Park, 2017).

Analyzing the arm in the plane spanned by r and s, finding θ_2 and θ_3 is reduced to solving the inverse kinematics of the planar 2-link arm shown in Figure 4.5. The parameters r and s are defined as

$$r = \sqrt{p_x^2 + p_y^2}, \quad s = L_1 - p_z$$

Using the law of cosine, the relation between θ_3 and the links can be written as

$$r^2 + s^2 = L_2^2 + (L_3 + L_4)^2 - 2L_2(L_3 + L_4)cos(\beta)$$

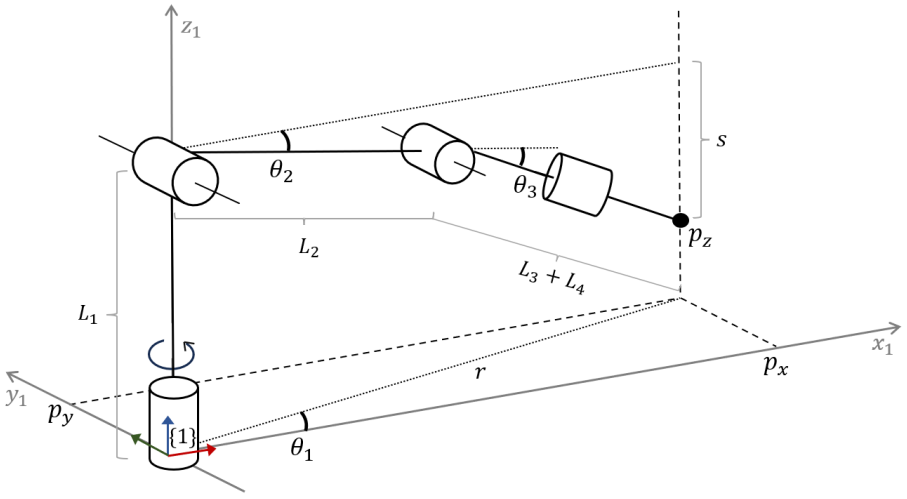


Figure 4.4: Inverse position of Lio's wrist.

Rearranging the equation, $cos(\beta)$ can be written as

$$cos(\beta) = \frac{L_2^2 + (L_3 + L_4)^2 - (r^2 + s^2)}{2L_2(L_3 + L_4)}$$

From Figure 4.5 we see that $\beta = \pi - \theta_3$ which implies $cos(\beta) = cos(\pi - \beta) = -cos(\theta_3)$. Using this relation, $cos(\theta_3)$ can be described as

$$cos(\theta_3) = \frac{(r^2 + s^2) - L_2^2 + (L_3 + L_4)^2}{2L_2(L_3 + L_4)} := c_{\theta_3} \tag{4.13}$$

Expressing the joint values with $atan2$ rather than $arcsin$ or $arccos$ is preferable because $atan2$ keeps track of the angle's quadrant (kilde). Using the trigonometric identity

$$cos(\theta) = b \implies \theta = atan2(\pm\sqrt{1 - b^2}, b)$$

θ_3 can be expressed as

$$\theta_3 = atan2(\pm\sqrt{1 - c_{\theta_3}^2}, c_{\theta_3}) \tag{4.14}$$

where c_{θ_3} is defined as in (4.13). The negative and positive roots represent the elbow-down and elbow-up solutions.

Looking at Figure 4.5, the angle $\theta_2 + \alpha$ can be expressed as

$$\theta_2 + \alpha = atan2(s, r)$$

Considering α as a part of the right triangle marked in blue on Figure 4.5, it can easily be derived with $atan2$ as

$$\alpha = atan2(sin(\theta_3)(L_3 + L_4), L_2 + cos(\theta_3)(L_3 + L_4))$$

Expressing θ_2 as $\theta_2 = \theta_2 + \alpha - \alpha$ results in the following expression

$$\theta_2 = atan2(s, r) - atan2(sin(\theta_3)(L_3 + L_4), L_2 + cos(\theta_3)(L_3 + L_4)) \tag{4.15}$$

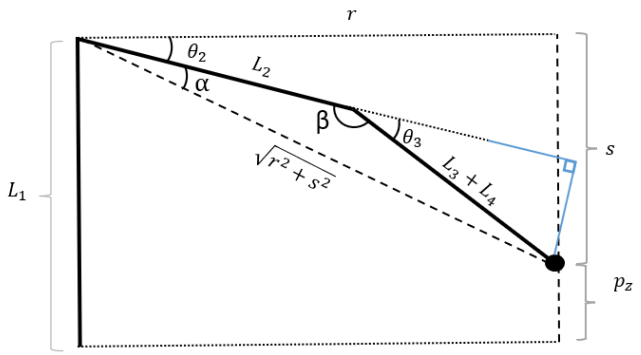


Figure 4.5: Inverse position of Lio's robot arm.

4.2.2 Inverse orientation

The three remaining joint angles θ_4, θ_5 and θ_6 define the orientation of the end effector. If we know a desired end effector frame $\mathbf{X}$ we can use the forward kinematics from Chapter 4.1.3. Denoting $\mathbf{T}(\theta) = \mathbf{X}$ we get

$$\mathbf{X} = e^{[S_1]\theta_1} e^{[S_2]\theta_2} e^{[S_3]\theta_3} e^{[S_4]\theta_4} e^{[S_5]\theta_5} e^{[S_6]\theta_6} \mathbf{M}$$

Since θ_1, θ_2 and θ_3 is known we can rearrange the equation as follows

$$e^{[S_4]\theta_4} e^{[S_5]\theta_5} e^{[S_6]\theta_6} = e^{-[S_3]\theta_3} e^{-[S_2]\theta_2} e^{-[S_1]\theta_1} \mathbf{X}\mathbf{M}^{-1}$$

The last three joints have a rotation axis around X, Y, and X, respectively. Therefore, their impact on the end effector can be described as an XYX-Euler rotation and their values can be determined as the solution to the equation

$$\mathbf{Rot}_x(\theta_4)\mathbf{Rot}_y(\theta_5)\mathbf{Rot}_x(\theta_6) = \mathbf{R} \tag{4.16}$$

where $\mathbf{R} = e^{[S_4]\theta_4} e^{[S_5]\theta_5} e^{[S_6]\theta_6}$ and the XYX Euler rotation is (Langston, 1976)

$$\mathbf{Rot}_x(\theta_4)\mathbf{Rot}_y(\theta_5)\mathbf{Rot}_x(\theta_6) = \begin{bmatrix} c_5 & s_5 s_6 & s_5 c_6 \\ s_4 s_5 & c_4 c_6 - c_5 s_4 s_6 & -c_4 s_6 - c_5 s_4 c_6 \\ -c_4 s_5 & c_6 s_4 + c_5 c_4 s_6 & c_4 c_5 c_6 - s_4 s_6 \end{bmatrix}$$

Solving 4.16 we get the following expressions for the last three joints

$$\theta_4 = atan2(\mathbf{R}_{21}, -\mathbf{R}_{31}) \tag{4.17}$$

$$\theta_5 = arccos(\mathbf{R}_{11}) = atan2(\pm\sqrt{1 - \mathbf{R}_{11}^2}, \mathbf{R}_{11}) \tag{4.18}$$

$$\theta_6 = atan2(\mathbf{R}_{12}, \mathbf{R}_{13}) \tag{4.19}$$

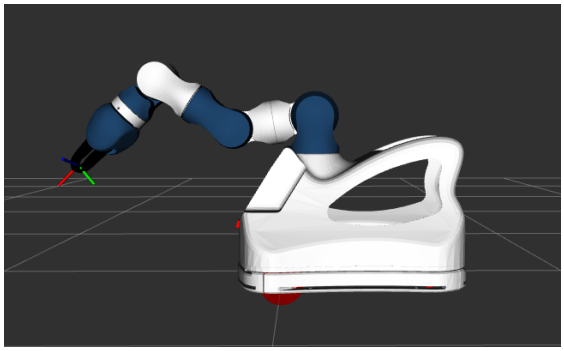
The values (4.18) and (4.19) holds for the positive root of θ_5, denoted "wrist orientation 1". For the negative root, "wrist orientation 2", the sign inside the $atan2$-functions are inverted,

$$\theta_4 = atan2(-\mathbf{R}_{21}, \mathbf{R}_{31}) \tag{4.20}$$

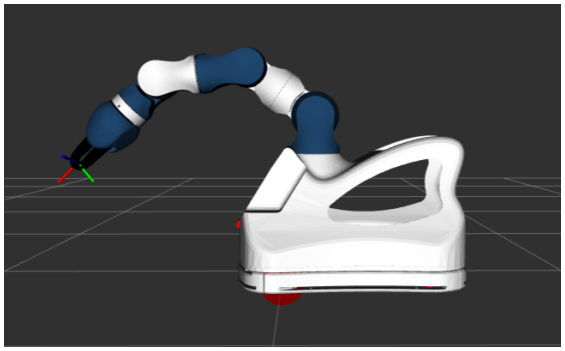
$$\theta_6 = atan2(-\mathbf{R}_{12}, -\mathbf{R}_{13}) \tag{4.21}$$

To summarize, four solutions were found for positioning the wrist; elbow up and down, and robot facing front and back. Two solutions were found for orientation; wrist orientation 1 and 2. In total $4 \times 2 = 8$ configurations yielded the same end effector pose. The eight different configurations are visualized in Figure 4.6.

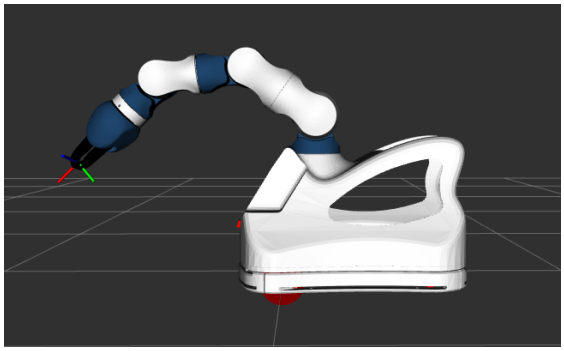
Figure 4.6: Lio's 8 configurations derived in section 4.2.



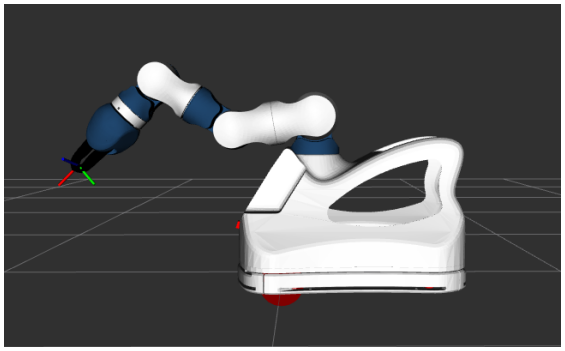
(a) `C1:` Elbow down, facing front, wrist pos 1.



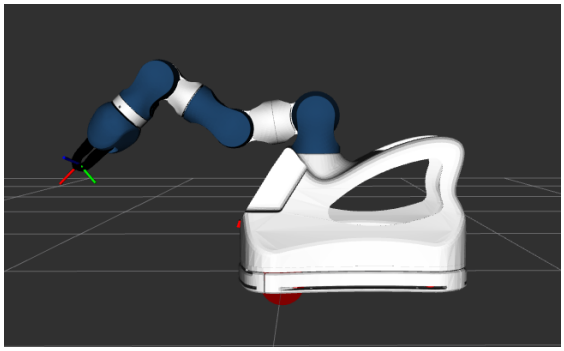
(b) `C2:` Elbow up, facing front, wrist pos 1.



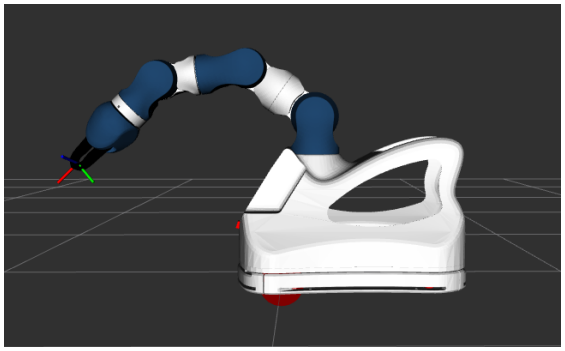
(c) `C3:` Elbow up, facing back, wrist pos 1.



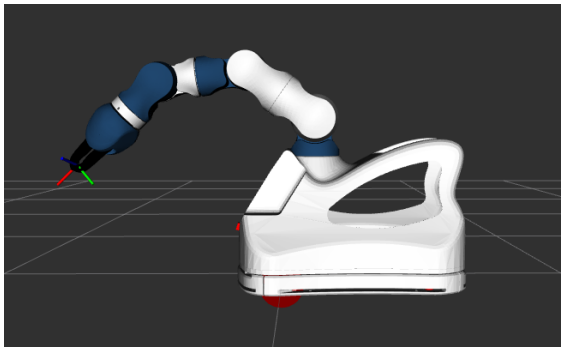
(d) `C4:` Elbow down, facing back, wrist pos 1.



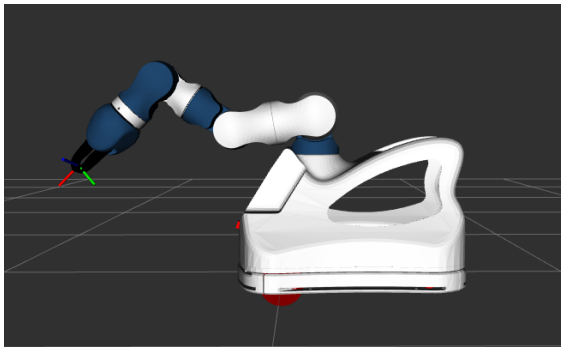
(e) `C5:` Elbow down, facing front, wrist pos 2.



(f) `C6:` Elbow up, facing front, wrist pos 2.



(g) `C7:` Elbow up, facing back, wrist pos 2.



(h) `C8:` Elbow down, facing back, wrist pos 2.

4.3 Forward kinematics with DH-parameters

Figure 4.7 shows a joint model of the Lio robot with link coordinate frames assigned according to the DH-convention. Table 4.2 displays the corresponding DH-parameters. The parameter d_{ap} is the displacement between the platform frame $\{p\}$ and the arm frame $\{a\}$ in the X_1-direction and p_{height} is the displacement in the Z_1-direction. Knowing the DH-parameters, the symbolic forward kinematic expression can be derived by multiplying the $i = 9$ transformation matrices defined in Equation 3.15. The symbolic forward kinematics can be seen in Appendix C and the actual link lengths are given in Appendix D.

Link	$\boldsymbol{\theta}$	$\boldsymbol{\alpha}$	$\boldsymbol{r}$	$\boldsymbol{d}$
1	θ_p^*	0	0	0
2	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	0	0
3	0	$\frac{\pi}{2}$	0	$x_p^* - d_{ap}$
4	$\theta_1 + \frac{\pi}{2}$	$-\frac{\pi}{2}$	0	$l_1 + p_{height}$
5	$\theta_2 - \frac{\pi}{2}$	0	l_2	0
6	θ_3	$-\frac{\pi}{2}$	0	0
7	θ_4	$\frac{\pi}{2}$	0	$l_3 + l_4$
8	θ_5	$-\frac{\pi}{2}$	0	0
9	θ_6	0	0	$l_5 + l_6$

Table 4.2: DH-parameters

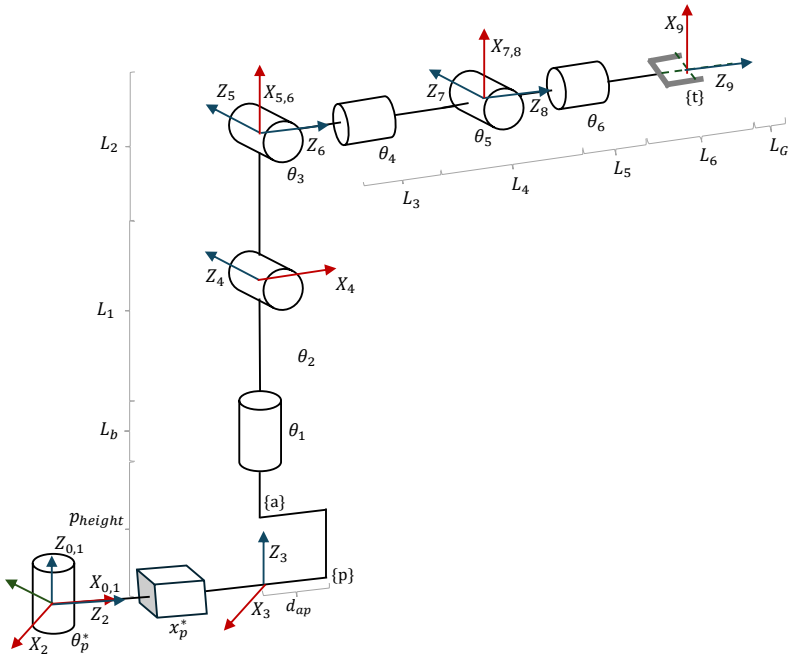


Figure 4.7: Link coordinate frames for the DH-convention.

Chapter 5

Redundancy resolution

This chapter introduces three different options for solving the redundant inverse kinematics problem. First, a solution based on extended task space with configuration control is presented. Next, two optimization-based approaches are derived with the objectives of minimizing platform movements and maximizing manipulability.

Section 5.1 is derived from the specialization project (Grøtterud, 2023).

5.1 Extended task space with configuration control

Considering the model shown in Figure 4.1, the inverse kinematics problem of the mobile manipulator can be formulated as the joint values $(\theta_p^*, x_p^*, \theta_1, ..., \theta_6)$ that solve

$$\mathbf{T}_{st}(\theta_p^*, x_p^*, \theta_1, \ ... \ , \theta_6) = \mathbf{T}_d \tag{5.1}$$

where $\mathbf{T}_{st}$ is the forward kinematics derived in chapter 4.1.3 and $\mathbf{T}_d$ is the desired end effector pose relative to the fixed space frame $\{s\}$. The mobile manipulator has a redundancy degree of $r = 8 - 6 = 2$. A simple way to exploit the two additional dof is by taking the state of the platform $\mathbf{q}_p = [\theta_p^*, x_p^*]$ as input. The frame at the base of the arm, $\{a\}$, can then be calculated as a function of $\mathbf{q}_p$. When the location of the base frame is known, the new desired transformation from the base to the end effector, $\mathbf{X}$, can be expressed as

$$\mathbf{X}(\mathbf{q}_p) = \mathbf{T}_{pa}^{-1}\mathbf{T}_{sp}^{-1}(\mathbf{q}_p)\mathbf{T}_{st} = \mathbf{T}_{at} \tag{5.2}$$

This approach equals the extended task space method with the platform configuration as secondary tasks. Rewriting (5.2) in vector form yields

$$\mathbf{t}_a = \begin{bmatrix} \mathbf{t} \\ \mathbf{t}_c \end{bmatrix} = f^{-1}\Big(\begin{bmatrix} \mathbf{q}_a \\ \mathbf{q}_p \end{bmatrix}\Big) \tag{5.3}$$

where $\mathbf{t}$ is the desired end effector pose represented by the task vector $\mathbf{t} = [x, y, z, \phi, \theta_t, \psi]^T$ and $\mathbf{t}_c = \mathbf{q}_p$ is the current platform configuration. Since the additional tasks constrain

the pose of the platform, this way of using extended task space is called posture control or configuration control.

A significant advantage of solving the redundancy resolution problem at the position level is that the analytic inverse kinematic solution, derived in Chapter 4.2, still can be used to solve the main task $\mathbf{t}$. Denoting the inverse kinematic solution as the function, I_a, the redundancy resolution with configuration control can be formulated as

$$\mathbf{t}_a = \begin{bmatrix} \mathbf{t} \\ \mathbf{t}_c \end{bmatrix} = \begin{bmatrix} I_a(\mathbf{q}_a) \\ \mathbf{q}_p \end{bmatrix} \tag{5.4}$$

Taking the platform state as input closes the inverse kinematic problem. The manipulator joints work as internal motions, compensating for changes in the platform state so that the desired end effector pose is maintained despite platform movements. In this way, the mobile platform acts as a redundant robot while still utilizing the analytic solution derived in Chapter 4.2.

5.2 Solving the inverse kinematics problem as nonlinear optimization problems

This section presents two approaches for solving the redundancy resolution by formulating a non-linear optimization problem (NLP). For increased readability, these methods are from here on referred to as NLP1 and NLP2.

5.2.1 NLP1: Minimizing platform movement by optimizing redundancy parameters

Defining redundancy parameters

The inverse kinematics of the mobile manipulator can be closed by introducing $r = 3$ redundancy parameters $\boldsymbol{\rho} = [\rho_1, \rho_2, \rho_3]$ that relates the desired pose to the platform configuration. The desired pose, $\mathbf{T}_d$, is represented by the task vector,

$$\mathbf{t} = [x_t, y_t, z_t, \phi_t, \theta_t, \psi_t]^T \tag{5.5}$$

where (x_t, y_t, z_t) is the desired location of the gripper and $(\phi_t, \theta_t, \psi_t)$ is the desired orientation represented as the XYZ-fixed angles (3.1.3). The platform configuration is represented by the state vector $\mathbf{q}_p = [x_p, y_p, \theta_p]$. Defining $\boldsymbol{\rho}$ as the "generalized mobile manipulator redundancy parameters" presented in (Ancona, 2017), the redundancy parameters describe the following geometric relations:

ρ_1 corresponds to the angular displacement between the mobile platform and the manipulator

$$\rho_1 = tan^{-1}\Big(\frac{y_t - y_a}{x_t - x_a}\Big) - \theta_p \tag{5.6}$$

where (x_t, y_t) and (x_a, y_a) are the position of the tool frame and the arm base frame, respectively, and θ_p is the platform's heading angle.

ρ_2 represents the arm extension as the distance in the xy-plane from the base of the arm to the end effector

$$\rho_2 = \sqrt{(x_t - x_a)^2 + (y_t - y_a)^2}. \tag{5.7}$$

ρ_3 describes the angular displacement between the end effector approach angle and the arm

$$\rho_3 = tan^{-1}\Big(\frac{y_t - y_a}{x_t - x_a}\Big) - \psi_t. \tag{5.8}$$

Using the redundancy parameters $\boldsymbol{\rho}$, the relation between the tool position $\{t\}$ and the arm base position $\{a\}$ can now be formulated as

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x_a \\ y_a \end{bmatrix} + \rho_2 \begin{bmatrix} cos(\rho_3 + \psi_t) \\ sin(\rho_3 + \psi_t) \end{bmatrix} \tag{5.9}$$

Exploiting the fixed relationship between the platform frame and the base of the arm

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x_a \\ y_a \end{bmatrix} + d \begin{bmatrix} cos(tan^{-1}(\frac{y_t - y_a}{x_t - x_a}) - \rho_1) \\ sin(tan^{-1}(\frac{y_t - y_a}{x_t - x_a}) - \rho_1) \end{bmatrix} \tag{5.10}$$

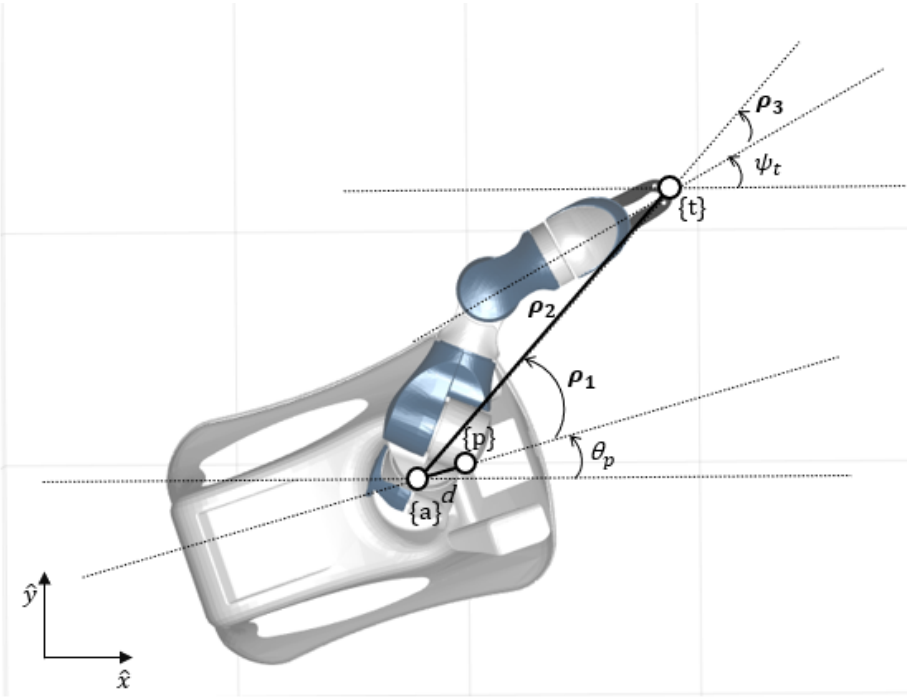


Figure 5.1: Generalized mobile manipulator redundancy parameters, defined by Ancona (2017), shown on a model of the Lio robot.

and the similarity

$$tan^{-1}(\frac{y_t - y_a}{x_t - x_a}) = \rho_1 + \theta_p = \rho_3 + \psi_t, \tag{5.11}$$

the state of the platform (x_p, y_p, θ_p) can be determined solemnly from the desired transformation as

$$\begin{bmatrix} x_p \\ y_p \\ \theta_p \end{bmatrix} = \begin{bmatrix} x_t - \rho_2 cos(\rho_3 + \psi_t) + dcos(\rho_3 + \psi_t - \rho_1) \\ y_t - \rho_2 sin(\rho_3 + \psi_t) + dsin(\rho_3 + \psi_t - \rho_1) \\ \rho_3 + \psi_t - \rho_1 \end{bmatrix} \tag{5.12}$$

Equation (5.12) defines the pose of the platform. Using the transform $\mathbf{T}_{pa}$ (4.9) the frame of the base of the arm is determined and can be used in the analytic inverse kinematics function to find the joint angles of the arm. Hence, the inverse kinematics of the mobile manipulator is fully constrained on the form

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_a \\ \mathbf{q}_p \end{bmatrix} = \begin{bmatrix} I_a(\mathbf{t}) \\ I_p(\mathbf{t}, \boldsymbol{\rho}) \end{bmatrix} \tag{5.13}$$

where I_a is the analytic inverse kinematics of the manipulator (4.2) and I_p is the relation between the platform state and the redundancy parameters (5.12).

Formulating the NLP

The objective function presented in Equation (5.14) can be designed to determine a wide range of desired redundant behaviors. In this thesis, the chosen optimization goal is to minimize platform motions. Reducing platform movements leaves the manipulator with the main responsibility for solving tasks. Since the platform lacks an accurate motion controller, minimizing platform motion can be desirable during high-precision tasks to increase the overall accuracy of the robot.

Finding suitable values for the redundancy parameters can be formulated as a nonlinear optimization problem on the form

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & J(\boldsymbol{\rho}, \mathbf{t}, \overline{\mathbf{q}}_p) \\ \text{s.t.} \quad & \boldsymbol{\rho}_{lower} \leq \boldsymbol{\rho} \leq \boldsymbol{\rho}_{upper} \\ & g(\boldsymbol{\rho}, \mathbf{t}, \overline{\mathbf{q}}_p) \leq 0 \end{aligned} \tag{5.14}$$

where $J(\boldsymbol{\rho}, \mathbf{t}, \overline{\mathbf{q}}_p)$ is a cost function that minimizes platform movement as

$$J(\boldsymbol{\rho}, \mathbf{t}, \overline{\mathbf{q}}_p) = || \overline{\mathbf{p}} - \mathbf{p} || + \alpha ||\overline{\theta}_p - \theta_p|| \tag{5.15}$$

where the first term is the Euclidean distance between the position component of the current and estimated platform configuration and the second term is the distance between the current and estimated heading direction, weighted by α. The norm of the state vector $\mathbf{q}_p$ is divided into two terms to account for the difference in scale between the position [mm] and rotation [deg] components. In the implementation of NLP1, α is set to

$$\alpha = || \overline{\mathbf{p}} - \mathbf{p} || \tag{5.16}$$

to ensure the position and rotation errors are on approximately the same scale.

The constraint function $g(\boldsymbol{\rho}, \mathbf{t}, \overline{\mathbf{q}}_p)$ uses the relation between the platform state and the redundancy parameters (5.12), together with the analytic inverse kinematics function derived in Section 3.3, to ensure that the configuration lies within the robot's workspace

$$\mathbf{q}_{lower} \leq \ g(\boldsymbol{\rho}, \mathbf{t}, \overline{\mathbf{q}}_p) = \mathbf{q}_a \ \leq \mathbf{q}_{upper} \tag{5.17}$$

The lower and upper bounds of g, $\{\mathbf{q}_{lower}, \mathbf{q}_{upper}\}$, are the physical joint limits of the manipulator, given in Table 6.1.

The bounds on the decision variable, $\boldsymbol{\rho}$, are

$$\boldsymbol{\rho}_{lower} = [q_{1lower}, 0, q_{7lower}], \ \boldsymbol{\rho}_{upper} = [q_{1upper}, 950, q_{7upper}] \tag{5.18}$$

where the bounds on ρ_1 and ρ_3 are given in [deg] and determined by the geometry in Figure 5.1 to have the same upper and lower limits as θ_1 and θ_7, respectively. The bounds on the planar arm extension, ρ_2, are given in [mm]. The upper bound was chosen slightly below the robot arm's maximum reach ($995mm$), to prevent the arm from fully extending into a singular configuration.

5.2.2 NLP2: Maximizing manipulability with numeric inverse kinematics

Using the robot state $\mathbf{q} = [\mathbf{q}_a, \ \mathbf{q}_p]$ as the decision variable in the NLP allows for easier implementation of more complex cost functions, like maximizing the manipulability index. The manipulability of a robot indicates the range of possible end effector velocities for a given configuration (Spong et al., 2019). High manipulability means that the robot has more options in which direction it can move, which is beneficial, for example, in collision avoidance (Arbo et al., 2019).

The optimization problem is formulated as

$$\begin{aligned} \min_{\mathbf{q}} \quad & F(\mathbf{q}, \mathbf{t}) - \alpha M(\mathbf{q}) \\ \text{s.t.} \quad & \mathbf{q}_{lower} \leq \mathbf{q} \leq \mathbf{q}_{upper} \\ & e(\mathbf{q}, \mathbf{t}) \leq \mathbf{e}_{tol} \end{aligned} \tag{5.19}$$

where $F(\mathbf{q}, \mathbf{t})$ describes the primary task of reaching the desired end effector pose

$$F(\mathbf{q}, \mathbf{t}) = \frac{1}{2}(f(\mathbf{q}) - \mathbf{t})^T (f(\mathbf{q}) - \mathbf{t}). \tag{5.20}$$

Here, $f(\mathbf{q})$ is the forward kinematics derived in Section 3.2 and $\mathbf{t}$ is the desired end effector pose. The secondary task $M(\mathbf{q})$, weighted by the scaling factor α, is defined as maximizing the manipulability index

$$M(\mathbf{q}) = \mu = \sqrt{det(\mathbf{J}_t \mathbf{J}_t^T)} \tag{5.21}$$

where $\mathbf{J}_t$ is the analytical Jacobian (3.20). The constraint is the error function

$$e(\mathbf{q}, \mathbf{t}) = \begin{bmatrix} ||\mathbf{p}(\mathbf{q}) - \mathbf{p}_d|| \\ Q_{err} \end{bmatrix} \tag{5.22}$$

ensuring convergence of the position and rotation. The first row takes the Euclidean distance between the position estimated by the forward kinematics function, $\mathbf{p}(\mathbf{q})$ and the desired position, $\mathbf{p}_d$. The second row is the difference in the rotations, specified by the quaternion distance metric, Q_{err}, defined in (3.8).

The bounds on the decision variable $\mathbf{q}$ are separated into bounds for the manipulator $\{\mathbf{q}_{a_{lower}}, \mathbf{q}_{a_{upper}}\}$ which equals the physical joint limits of the manipulator given in Table 6.1, and the bounds on the platform's virtual joints

$$\mathbf{q}_{p_{lower}} = [-180, -5000],\ \mathbf{q}_{p_{upper}} = [180, 5000] \tag{5.23}$$

The bounds on the virtual revolute joint θ_p^* allow the platform to rotate $360deg$ and the bounds on the virtual prismatic joint x_p^* were chosen to provide sufficient range to reach the desired poses tested in Section 7.4.

Deciding on values for the scaling factor α and the error tolerance $\mathbf{e}_{tol}$ is a tradeoff between accuracy and manipulability. A higher α prioritizes maximizing manipulability, enhancing the robot's movement options but potentially reducing precision in reaching the desired pose. Conversely, a lower α ensures accurate positioning at the cost of reduced flexibility. Similarly, a smaller $\mathbf{e}_{tol}$ ensures high precision, but may lead to longer computation times and convergence issues. On the other hand, a larger $\mathbf{e}_{tol}$ facilitates faster convergence but at the expense of precision.

Chapter 6

Implementation

This Chapter describes how the developed kinematic model is implemented on the robot as a ROS node and how it communicates with the preexisting robot modules. First, each robot module and its function is described. Then, the program flow of the redundancy resolution method derived in Section 5.1 is explained, and the limitations of the current implementation method are discussed briefly. Lastly, the implementation of the two optimization-based methods is covered.

The sections 6.1.1, 6.1.3 and 6.3 are derived from the specialization project (Grøtterud, 2023).

6.1 Robot modules

6.1.1 myP

myP is a software framework provided by F&P-robotics to control Lio's robotic arm and gripper. The framework has been designed primarily as a web application, but a selection of its functions can also be accessed through a ROS connection (AG, 2023). In this project, myP was integrated into a ROS environment to increase scalability and simplify communication with other modules. The myP-node publishes topics concerning the state of the robot arm and sensor outputs from the gripper. Some myP functions are available as ROS services. The topic `/lio_1c/joint_positions` is used to receive the current position of all arm joints in degrees. Position commands to the arm joints are given through the service `/lio_1c/core/move_joint`.

6.1.2 Platform workstation and gamepad

Lio's platform is controlled by a workstation running Ubuntu and ROS. The platform node subscribes to velocity commands sent on the ROS topic `/cmd_vel`. It publishes the

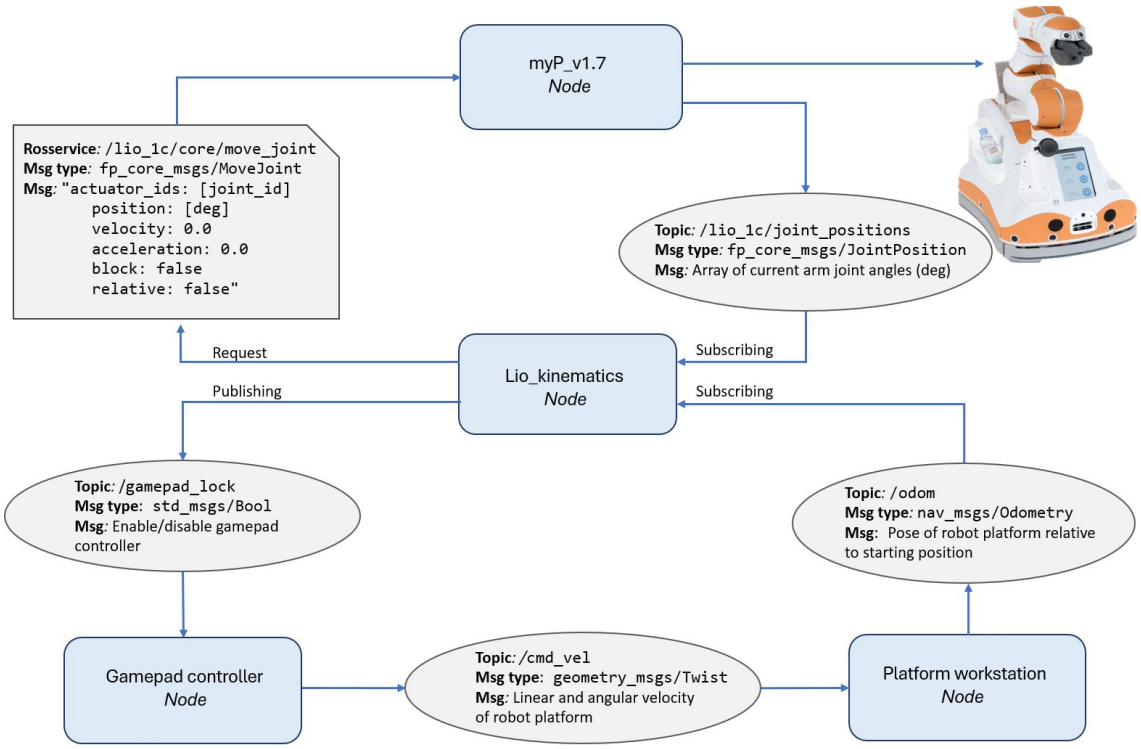


Figure 6.1: Communication with Lio through ROS modules.

position and orientation of the robot, relative to its pose at power-up, on the `/odom` topic. A gamepad has been integrated as an interface to control the movement of the platform by publishing linear- and angular velocity commands. The gamepad can be enabled and disabled during the "redundancy mode" with the `/gamepad_lock` topic.

6.1.3 Lio_kinematics

The Lio_kinematics node is the program's brain and connects all the robot modules. Its main structure is the kinematic model from Chapter 4 implemented as an object-oriented class in Python named `Lio_kinematics()`. The class attributes include all dimensions of the robot, the transformation matrices derived in Chapter 4.1 and the current state of the robot arm and the platform. The robot state is updated by the `/odom` and `/lio_1c/joint_positions` topics, published by the platform workstation and myP-node. The class methods are, amongst others, the analytic inverse kinematics derived in Chapter 4.2 and a redundant mode based on the redundancy resolution with configuration control (Section 5.1). The `Lio_kinematics()` class's main purpose is to allow for concurrent control of both the platform and the robot arm and enable the redundant mode described in the next section.

6.2 Redundant mode

The extended task space method derived in Section 5.1 is implemented as a `redundant_mode` activated with the `SOUTH_WEST` gamepad button. When the `redundant_mode` is activated, the gripper pose is locked while the platform is driveable. The `redundant_mode` enables the robot to exploit its redundant degrees of freedom as the platform can move freely within the robot's workspace while the gripper pose is approximately constant.

When the `redundant_mode` is activated, the robot's current pose is saved as the desired pose $\mathbf{T}_d$ in the form of the transformation matrix from a fixed point in space to the TCP,

$$\mathbf{T}_d = \mathbf{T}_{st}.$$

While the redundancy button is pressed, new joint reference angles, $\boldsymbol{\theta}_{ref}$, are calculated by the `inverse_kinematics(T_d)` (Algorithm 1) function every 10th millisecond. Next, $\boldsymbol{\theta}_{ref}$ are sent with a request to the rosservice `/lio_1c/core/move_joint`.

Due to the robot arm's limited joint velocity of $45deg/s$, a platform halt has been added to avoid temporary deviations from the desired pose. A zero-velocity command is sent to the platform if the reference angles exceed the current joint values by more than $30deg$. The threshold value of $30deg$ was determined experimentally as a balance between preventing the platform from stopping too frequently and avoiding excessive deviations from the desired pose. The gamepad is disabled with the `/gamepad_lock` topic, and the program is paused until the arm joints have reached their reference values and the joint motors have stopped. Figure 6.2 shows a flow diagram of the redundant mode.

Algorithm 1 Decide new joint angles with the analytic inverse kinematics

Input Desired transform $\mathbf{T}_d$, current platform state (x_p, y_p, θ_p)

$\mathbf{X}(x_p, y_p, \theta_p) \leftarrow \mathbf{T}_{pa}^{-1}\mathbf{T}_{sp}^{-1}(x_p, y_p, \theta_p)\mathbf{T}_d$

for every configuration n **do**
  $\theta_1, \theta_2, \theta_3 \leftarrow$ inv_position(X) ▷ inv_position from chapter 4.2.1
  $\theta_4, \theta_5, \theta_6 \leftarrow$ inv_orientation$(X, (\theta_1, \theta_2, \theta_3))$ ▷ inv_orientation from chapter 4.2.2
  $\theta_\mathbf{n} \leftarrow (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$
  if every θ in $\theta_\mathbf{n}$ is within joint limits **then**
    `possible_joint_configurations` $\leftarrow \theta_\mathbf{n}$
  end if
end for

for every $\theta_\mathbf{n}$ in `possible_joint_configurations` **do**
  `euclidean_distances` $\leftarrow ||\theta_\mathbf{n} -$ `current_joint_values`$||$
end for

`joint_values` $\leftarrow$ `possible_joint_configurations`$(argmin($`euclidean_distances`$))$
return : `joint_values`

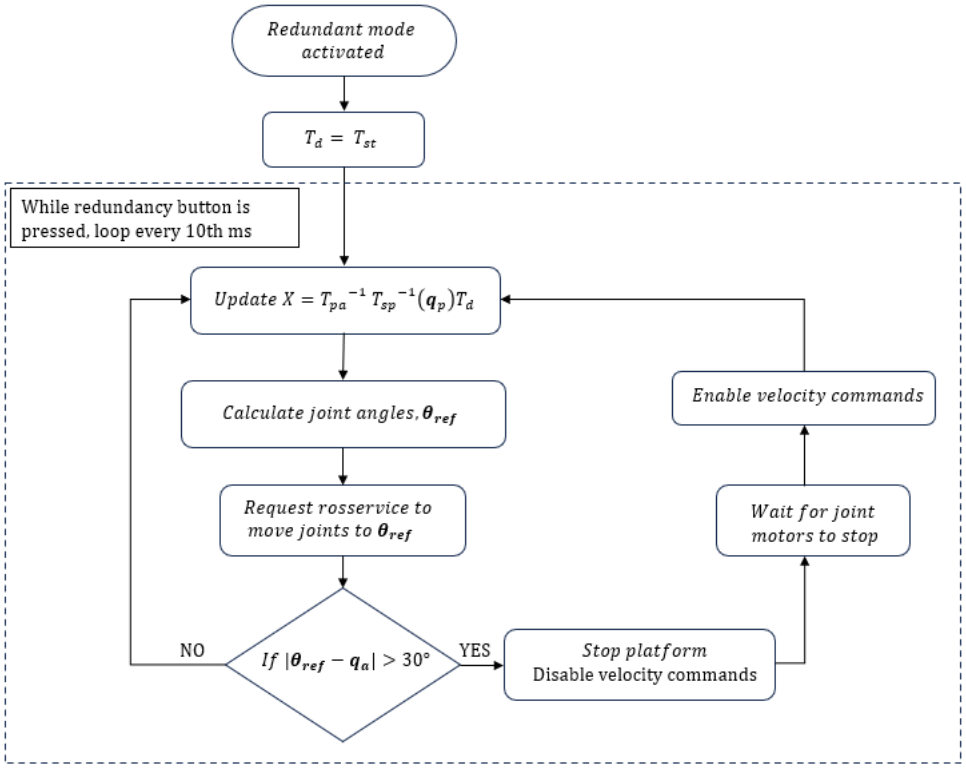


Figure 6.2: Program flow of redundancy function.

The redundant mode is made specifically for when the platform is operated manually with the gamepad. Therefore, the state of the platform is not restricted to a model of two virtual joints. Instead, the platform coordinate frame can be updated directly from the `/odom` topic with three parameters (x_p, y_p, θ_p). Because of the restricted joint limits of the robot (Table 6.1), the manipulator needs to switch configurations to maintain the gripper pose as long as possible. Algorithm 1 shows how the extended task space method from Section 5.1 has been expanded to look for solutions amongst all eight configurations. The chosen configuration is the one closest to the current joint state that does not violate the joint limits.

6.3 Offset from manufacturer's zero configuration

The zero configuration used in the kinematic modeling differs from the zero configuration determined by F&P. Figure 6.3 shows the manufacturer's zero configuration on the left, with its corresponding link reference frames, compared with the zero configuration used in chapter 4 on the right. The link reference names in the figure relate to the frame names

used in chapter 4 as

`base_footprint` : The platform frame, $\{p\}$.

`LIO_robot_base_link` : The frame at the base of the arm, $\{a\}$.

`lio_tcp_link` : The tool frame, $\{t\}$.

In the home configuration of this paper's kinematic model, the arm has a $90deg$ bend in the elbow joint, θ_3, and all the robot link frames are defined to have the same orientation as the platform frame, $\{p\}$ (4.1). In the manufacturer's zero configuration, frame $\{a\}$ is rotated $-90deg$ around the z-axis relative to frame $\{p\}$ and frame $\{t\}$ is rotated $180deg$ relative to frame $\{a\}$ (left in Figure 6.3). To compensate for the difference in zero-configuration, an offset vector,

$$\texttt{offset\_vec} = [-90 \quad 0 \quad 90 \quad 0 \quad 0]$$

must be added to joint angles outputted from the inverse kinematics function in `Lio_kinematics()` so the joint commands yield the desired outcome on the actual robot. Note that the $180deg$ rotation of $joint_6$ (shown on the right in Figure 6.3) was omitted from the `offset_vec` because it exceeds the joint limits (-170, 169). The gripper of the robot is upside down compared to the model due to the exclusion of the $180deg$ rotation. However, this has no practical significance as the gripper is symmetric.

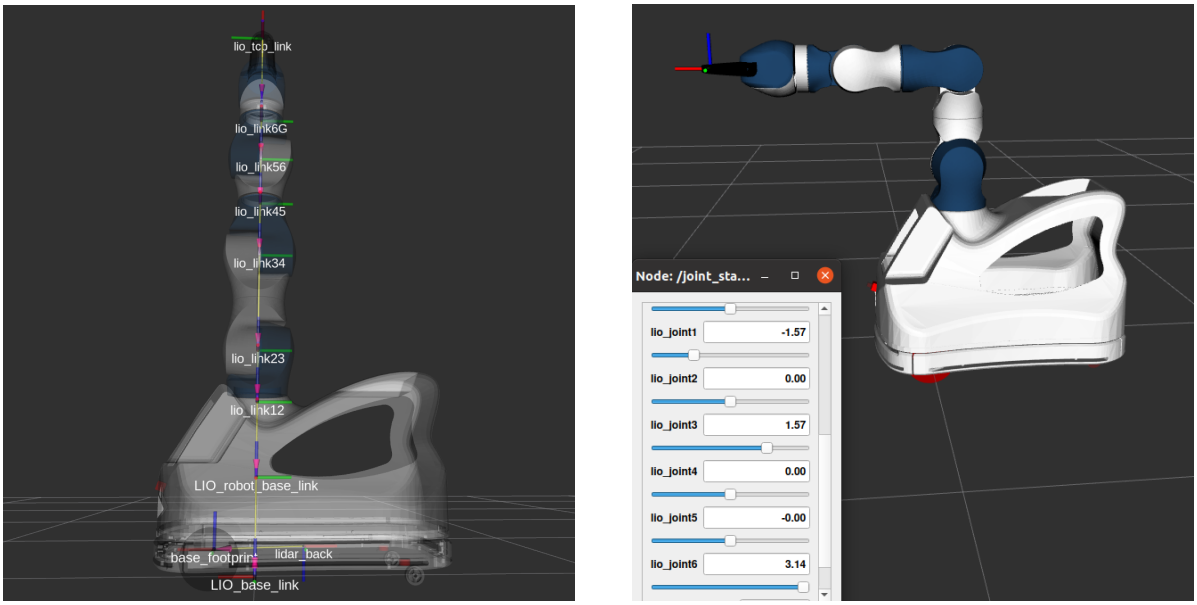


Figure 6.3: Comparison of the robot in the zero position defined by the manufacturers (left) and the zero position chosen in the kinematic modeling (right).

6.4 Limitations

The provided control software, myP, is designed for "regular users" and is not suitable for research projects due to its poor GUI, lack of version control, and limited debugging

capabilities. Therefore, the implemented code was programmed in a conventional IDE (Visual Studios Code), and information about the robot's state and integrated functions was accessed through ROS as illustrated in Figure 6.1. However, only a limited set of data and integrated functions are published and available as rosservices.

Another significant limitation was the lack of low-level motion control on the robot platform. As of now, the platform is only controllable through velocity commands on the `/cmd_vel` topic, published directly with commands from the terminal or through the gamepad. Velocity commands published to the `/cmd_vel` were found to have a significant delay of approximately $1.5s$ (determined experimentally). This limitation makes implementing redundancy resolutions that use automatic platform positioning unfeasible.

Joint angle	**Lower limit [deg]**	**Upper limit [deg]**
θ_1	-90	260
θ_2	-109	110
θ_3	-205	25
θ_4	-169	170
θ_5	-114	114
θ_6	-170	169

Table 6.1: Manipulator joint limits relative to the kinematic model's zero position (right in Figure 6.3).

6.5 Formulating the NLPs in CasADi with the IPOPT-solver

The two NLPs were programmed in Python using the CasADi framework. CasADi is an open-source symbolic framework for automatic differentiation and numerical optimization (Andersson et al., 2019). This framework was chosen for its ability to efficiently compute large symbolic expressions, such as the manipulability index, and for its integration with the IPOPT (Interior Point OPTimizer) solver. The IPOPT solver is a software package that finds local solutions to nonlinear optimization problems using an interior-point line search method (Waechter and Laird, 2005). It was chosen for its robust handling of nonlinear and non-convex objectives and constraints. IPOPT efficiently exploits sparsity in large-scale problems and uses derivative information effectively, making it suitable for optimization tasks in robotic redundancy resolution.

Due to the lack of an accurate robot platform controller, these methods were not implemented on the Lio robot. However, they were designed to be included as functions in the `Lio_kinematics` class to access the current robot state and directly give commands to the manipulator and platform controller. The main structure of the code for the optimization problems is included in the appendices A and B.

6.5.1 Parameters in NLP1

NLP1 is formulated symbolically in Section 5.2.1 as

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & J(\boldsymbol{\rho}, \mathbf{t}, \overline{\mathbf{q}}_p) = || \, \overline{\mathbf{p}} - \mathbf{p} \, || + \alpha ||\overline{\theta}_p - \theta_p|| \\ \text{s.t.} \quad & \boldsymbol{\rho}_{lower} \leq \boldsymbol{\rho} \leq \boldsymbol{\rho}_{upper} \\ & \mathbf{q}_{lower} \leq \; g(\boldsymbol{\rho}, \mathbf{t}, \overline{\mathbf{q}}_p) = \mathbf{q}_a \; \leq \mathbf{q}_{upper} \end{aligned} \tag{6.1}$$

where the scaling factor α and the upper and lower bounds on ρ are given in Equations 5.16 and 5.18, respectively, and the bounds on g, $\{\mathbf{q}_{lower}, \mathbf{q}_{upper}\}$, are the physical joint limits of the manipulator, given in Table 6.1. The desired pose, represented as the task vector, $\mathbf{t}$, and the current platform configuration, $\overline{\mathbf{q}}_p$, are the parameters in the optimization problem and are input from the `Lio_kinematics` class.

The choice of initial guess $\boldsymbol{\rho}_0$ is especially important since the IPOPT-solver only guarantees convergence to a local optimum. To avoid unnecessary platform movement, the initial guess first calculates the redundancy parameters with the given task vector $\mathbf{t}$ and current platform configuration $\overline{\mathbf{q}}_p$. If ρ_2 exceeds its upper bound, $\rho_2 > 950$, the desired end effector pose is considered out of reach for the manipulator and platform movement is required. In this case, the initial guess is set to $\boldsymbol{\rho}_0 = [0, 700, 0]$. These values for ρ_{01} and ρ_{03} were chosen because zero displacement between the platform-, manipulator- and gripper approach angle results in an intuitive robot pose. The value $\rho_{02} = 700$ was chosen because it represents a dexterous manipulator configuration where the arm is neither too extended nor too constrained. If $\rho_2 \leq 950$, the calculated values are retained as the initial guess since the desired pose might be achievable without platform movement.

6.5.2 Parameters in NLP2

NLP2 is formulated symbolically in Section 5.2.2 as

$$\begin{aligned} \min_{\mathbf{q}} \quad & \frac{1}{2}(f(\mathbf{q}) - \mathbf{t})^T (f(\mathbf{q}) - \mathbf{t}) - \alpha \sqrt{det(\mathbf{J}\mathbf{J}^T)} \\ \text{s.t.} \quad & \mathbf{q}_{lower} \leq \mathbf{q} \leq \mathbf{q}_{upper} \\ & \begin{bmatrix} ||\mathbf{p}(\mathbf{q}) - \mathbf{p}_d|| \\ Q_{err} \end{bmatrix} \leq \mathbf{e}_{tol} \end{aligned} \tag{6.2}$$

where the first term of the objective function ensures convergence to the desired pose and the second term maximizes the manipulability index. The bounds on the decision variable $\mathbf{q} = [\mathbf{q}_a, \mathbf{q}_p]$ are given in (5.23) for $\mathbf{q}_p$. The bounds on $\mathbf{q}_a$ equals the physical joint limits of the manipulator given in Table 6.1. The initial guess is set to the robot's current configuration to help find a local minimum that requires minimal movement.

The trade-off when choosing values for the scaling factor α and the error tolerance $\mathbf{e}_{tol}$ was briefly discussed in Section 5.2.2. These values were experimentally determined as

$$\alpha = 1 \times 10^{-5}, \quad \mathbf{e}_{tol} = \begin{bmatrix} 0.1 \\ 1 \times 10^{-8} \end{bmatrix} \tag{6.3}$$

to strike a balance between maximizing manipulability while maintaining the primary task's accuracy without significantly increasing runtime. The value of α needs to be small to balance the relatively large manipulability index, ranging from $10^7 - 10^9$ during testing in Section 7.4.2. The position error tolerance allows a position deviation of $0.1mm$ and a Q_{err} of 1×10^{-8}, corresponding to a rotation difference of $< 0.001rad$ around one axis in Euler angle representation. Thus, the chosen values for the error tolerance can be considered practically zero for most applications. However, further tuning to optimize these parameters is recommended for future work.

Chapter 7

Testing and results

This chapter presents the testing procedures and results for the three redundancy resolution methods derived in Chapter 5. First, the configuration control method was experimentally tested on the Lio robot at PPM Robotics' nursing home lab. The tests were conducted with both a static and dynamic base, and with two different platform velocities. The performance was evaluated based on the accuracy of reaching a desired end effector pose and the ability to maintain a fixed gripper pose during platform movement. Secondly, the two NLP methods were tested and visualized in Matlab. The evaluation criteria for these optimization algorithms included their effectiveness in fulfilling secondary tasks, such as minimizing platform motion and maximizing manipulability, as well as their runtime and number of iterations.

Figure 7.1: Lio in its test environment at PPM Robotic's nursing home lab.

7.1 Test parameters and evaluation metrics

The primary task in all the tests is to reach a desired end effector pose. The end effector pose is represented by the transformation matrix introduced in Section 3.1.3,

$$\mathbf{T}_{st} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3), \text{ with } \mathbf{R} \in SO(3), \ \mathbf{p} \in \mathbb{R}^3 \tag{7.1}$$

where $\{s\}$ represents a fixed space frame and $\{t\}$ is the tool frame of the robot. The elements of the position vector, $\mathbf{p}$, are given in [mm].

The state of the robot arm is represented by the joint vector

$$\mathbf{q}_a = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6] \tag{7.2}$$

with the joint values $\boldsymbol{\theta}$ given in degrees because that is the measurement unit for the robot's joint commands. The state of the platform is represented by the vector

$$\mathbf{q}_p = [x_p, y_p, \theta_p] \tag{7.3}$$

where (x_p, y_p) is the position and θ_p the rotation of the platform frame $\{p\}$ relative to the space frame $\{s\}$. The platform frame is located at the center of the platform's front wheel axis as illustrated in Figure 4.1.

To evaluate the success of reaching the desired end effector pose, the desired pose $\mathbf{T}_d$ is compared to the actual pose of the robot $\mathbf{T}_{st}$. The actual pose, $\mathbf{T}_{st}$, is found by inputting the current robot state $\mathbf{q}_{mm} = [\mathbf{q}_a, \mathbf{q}_p]$ to the forward kinematics derived in Section 4.1.3. Comparing the transformation matrices of the actual and desired pose is split into comparing position and orientation. The Euclidean distance is used to measure deviation in position

$$\mathbf{p}_{err} = ||\mathbf{p}_d - \mathbf{p}_{st}|| \tag{7.4}$$

To compare the orientation of the poses, the rotation matrices of the actual and desired transformation matrix, $\mathbf{R}_{st}$ and $\mathbf{R}_d$, were transformed to quaternions. The quaternion distance metric Q_{err}, described in Section 3.1.3, is used to measure deviation from the desired rotation.

$$Q_{err} = 1 - |Q_d \cdot Q_{st}| \tag{7.5}$$

The robot's active and feasible configurations are also evaluated. The eight solutions of the analytic inverse kinematics, derived in Section 4.2, are represented by an $[8 \times 7]$ configuration matrix. An example of the configuration matrix is shown below,

$$\begin{bmatrix} 1 & 43.2 & 98.6 & -144.5 & 12.8 & 61.6 & 51.5 \\ 1 & 43.2 & 42.2 & -35.5 & 53.2 & 14.1 & 5.2 \\ 0 & -136.8 & -42.2 & -144.5 & -126.8 & 14.1 & 5.2 \\ 0 & -136.8 & -98.6 & -35.5 & -167.2 & 61.6 & 51.5 \\ 1 & 43.2 & 98.6 & -144.5 & -167.2 & -61.6 & -128.5 \\ 0 & 43.2 & 42.2 & -35.5 & -126.8 & -14.1 & -174.8 \\ 0 & -136.8 & -42.2 & -144.5 & 53.2 & -14.1 & -174.8 \\ 0 & -136.8 & -98.6 & -35.5 & 12.8 & -61.6 & -128.5 \end{bmatrix} \tag{7.6}$$

where each row of the configuration matrix is on the form

$$row_n = \begin{bmatrix} 0/1 & \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 \end{bmatrix}$$

The first column is a boolean value indicating whether the configuration is within joint limits (1) or validating joint limits (0). Columns $2-7$ contains the calculated joint positions for $\theta_1 - \theta_6$ in degrees. The row number $1-8$ coincides with the configurations `C1-C8` illustrated in Figure 4.6. The example configuration matrix (7.6) represents a case with three feasible configurations, `C1, C2, C5`. The blue color highlights the configuration chosen as the solution to the inverse kinematics problem. The selected configuration is the one that minimizes the Euclidean distance to the current arm pose, as described in Algorithm 1.

For the optimization algorithms, manipulability and changes in the platform state are also evaluated for the output poses. The manipulability is evaluated with the manipulability index

$$m = \sqrt{\det\left(\frac{\partial f(\mathbf{q})}{\partial \mathbf{q}} \frac{\partial f(\mathbf{q})}{\partial \mathbf{q}}^T\right)} \tag{7.7}$$

and the change in platform state is evaluated with the Euclidean distance between the current and proposed platform state

$$||\overline{\mathbf{q}}_p - \mathbf{q}_p|| \tag{7.8}$$

The runtime and number of iterations of the solver are also considered.

7.2 Test 1: Configuration control with static base positioning

7.2.1 Experimental setup

The first experiment tested the robot's repeatability with the kinematic model from Chapter 4. The repeatability was tested by checking if Lio could return to a programmed end effector pose, $\mathbf{T}_d$. The test was performed as follows

1. The manipulator was set in `release_mode` to enable manual guiding of the arm.
2. Lio was moved to a configuration where it pointed at a bottle on the bedside table (Figure 7.2b).
3. The current pose was saved as the desired pose $\mathbf{T}_{st} = \mathbf{T}_d$.
4. The `release_mode` was deactivated.
5. Lio was moved to an arbitrary location where the bottle at the bedside table was within reach of the manipulator.

(a) Lio in its home configuration.

(b) Lio's end effector at the desired pose $\mathbf{T}_d$.

6. The `Lio_kinematics` node was initialized.
7. The manipulator was moved to its home configuration (Figure 7.2a).
8. The current platform pose and the desired end effector pose were input to the configuration control method (Algorithm 1) to find new joint angles.

Steps $4-8$ were repeated, so the configuration control was tested with two different inputs.

The state of the robot platform, $\mathbf{q}_p$ is determined by the `/odom` rostopic, which is initiated when the robot is powered up. Since Lio was turned on at the charging station, the desired pose $\mathbf{T}_d$ in Test 1,

$$\mathbf{T}_d = \begin{bmatrix} -0.95 & -0.12 & -0.30 & -699.2 \\ 0.12 & -0.99 & 0.02 & -1994.0 \\ -0.30 & -0.02 & 0.95 & 529.2 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix} \tag{7.9}$$

is given relative to the $\{s_1\}$ frame modeled in Figure 7.3

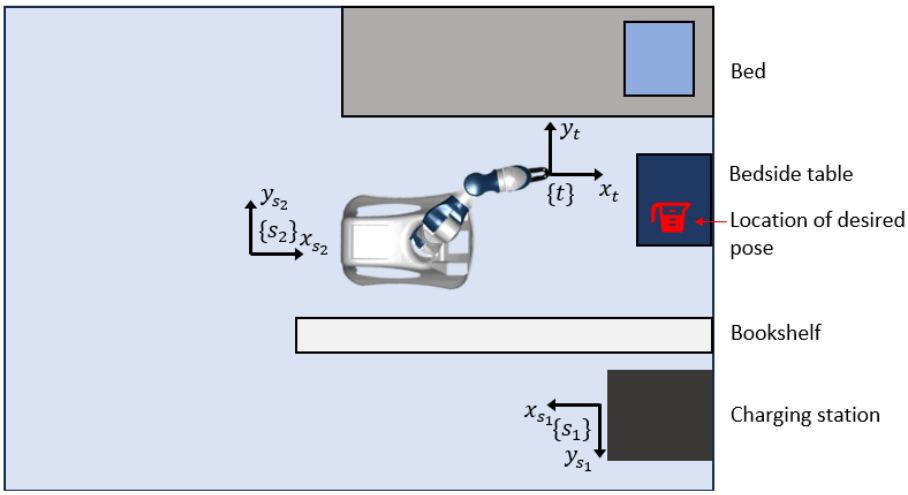


Figure 7.3: Illustration of the test environment at PPM nursing home lab with the space frames $\{s_1\}$ and $\{s_2\}$ for Test 1 and Test 2 respectively.

7.2.2 Results

Inital pose 1

In the first test, the initial platform pose was

$$\mathbf{q}_p = [112.2, -2098.8, 171.1] \tag{7.10}$$

The configuration matrix from the analytical inverse kinematics showed that configuration `C6` was the only feasible arm configuration.

$$\begin{bmatrix} 0.0 & 1.0 & 114.8 & -143.7 & 1.0 & 46.8 & -1.8 \\ 0.0 & 1.0 & 59.1 & -36.3 & 171.6 & 5.1 & -172.9 \\ 0.0 & -179.0 & -59.1 & -143.7 & -8.4 & 5.1 & -172.9 \\ 0.0 & -179.0 & -114.8 & -36.3 & -179.0 & 46.8 & -1.8 \\ 0.0 & 1.0 & 114.8 & -143.7 & -179.0 & -46.8 & 178.2 \\ 1.0 & 1.0 & 59.1 & -36.3 & -8.4 & -5.1 & 7.1 \\ 0.0 & -179.0 & -59.1 & -143.7 & 171.6 & -5.1 & 7.1 \\ 0.0 & -179.0 & -114.8 & -36.3 & 1.0 & -46.8 & 178.2 \end{bmatrix} \tag{7.11}$$

With the joint command $\mathbf{q}_a = [1.0, 59.1, -36.3, -8.1, -5.1, 6.9]$, Lio reached the end effector configuration $\mathbf{T}_{s_1t}$

$$\mathbf{T}_{s_1t} = \begin{bmatrix} -0.94 & -0.12 & -0.31 & -698.9 \\ 0.12 & -0.99 & 0.02 & -1994.0 \\ -0.31 & -0.02 & 0.95 & 528.9 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}. \tag{7.12}$$

The position and orientation error between $\mathbf{T}_{s_1t}$ and $\mathbf{T}_d$ was

$$\begin{aligned} \mathbf{p}_{err} &\approx 0.5mm \\ Q_{err} &\approx 5.57 \times 10^{-7} \end{aligned} \tag{7.13}$$

which is approximately 0 for practical applications.

Inital pose 2

In the second test, the initial platform pose was

$$\mathbf{q}_p = [-54.6, -2162.6, -163.5] \tag{7.14}$$

The output from the analytical inverse kinematics showed again that only configuration `C6` was feasible.

$$\begin{bmatrix} 0.0 & -30.1 & 139.9 & 166.3 & 6.4 & 71.6 & -5.0 \\ 0.0 & -30.1 & 31.4 & 13.7 & 167.0 & 27.9 & -171.4 \\ 0.0 & 149.9 & -31.4 & 166.3 & -13.0 & 27.9 & -171.4 \\ 0.0 & 149.9 & -139.9 & 13.7 & -173.6 & 71.6 & -5.0 \\ 0.0 & -30.1 & 139.9 & 166.3 & -173.6 & -71.6 & 175.0 \\ 1.0 & -30.1 & 31.4 & 13.7 & -13.0 & -27.9 & 8.6 \\ 0.0 & 149.9 & -31.4 & 166.3 & 167.0 & -27.9 & 8.6 \\ 0.0 & 149.9 & -139.9 & 13.7 & 6.4 & -71.6 & 175.0 \end{bmatrix} \tag{7.15}$$

With the joint command $\mathbf{q}_a = [-30.1, 31.4, 13.7, -13.0, -27.9, 8.6]$, the final end effector configuration was

$$\mathbf{T}_{s_1 t} = \begin{bmatrix} -0.95 & -0.12 & -0.30 & -699.0 \\ 0.12 & -0.99 & 0.02 & -1993.8 \\ -0.30 & -0.02 & 0.95 & 529.6 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix} \tag{7.16}$$

which is equivalent to a position and orientation error of

$$\begin{aligned} \mathbf{p}_{err} &\approx 0.5mm \\ Q_{err} &\approx 7.26 \times 10^{-8} \end{aligned} \tag{7.17}$$

7.3 Test 2: Configuration control with dynamic base positioning

The second test aimed to assess how well Lio maintained its gripper pose while the platform was moving. The implemented `redundant_mode` (6.2) allows the platform to be controlled manually with the gamepad while the gripper pose is fixed. The `redundant_mode` can be beneficial when operating in narrow spaces and provides a predictable robot behavior. To emulate a scenario in which the `redundant_mode` could be applicable, Lio was positioned with a stool as an obstacle between itself and the desired pose. Next, Lio was manually driven around the obstacle while the `redundant_mode` was active before it reached the desired pose.

7.3.1 Experimental setup

The test started with the same steps $1 - 4$ as Test 1. Then the test was structured as follows:

5. Lio was moved to a position where the desired pose was out of reach for the manipulator and a straight path to the goal was obstructed by an obstacle (Figure 7.5a).
6. The manipulator was moved to an outstretched arm configuration, $\mathbf{q}_a = [15, 40, -45, 0, 2.5, 0]$.
7. The platform was driven around the obstacle while in `redundant_mode`.
8. When the robot was sufficiently close to the desired pose, the platform was stopped and the current platform pose and the desired end effector pose were inputted to the configuration control method (Algorithm 1) to find new joint angles.

Due to Lio being powered up at a new location for Test 2, the desired pose $\mathbf{T}_d$ was given relative to the space frame $\{s_2\}$ in Figure 7.3.

$$\mathbf{T}_d = \begin{bmatrix} 0.62 & -0.76 & 0.16 & 797.0 \\ 0.74 & 0.64 & 0.19 & 943.4 \\ -0.25 & 0.00 & 0.97 & 544.4 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix} \tag{7.18}$$

To move around the obstacle, Lio's platform followed the path illustrated in Figure 7.4 with a linear speed of $v = 0.02m/s$ and an angular speed of $\dot{\theta} = 0.05rad/s$.

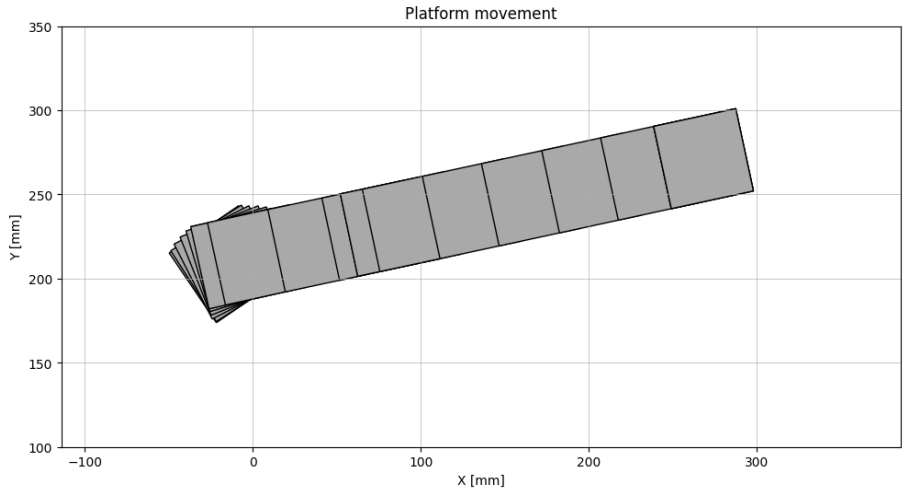


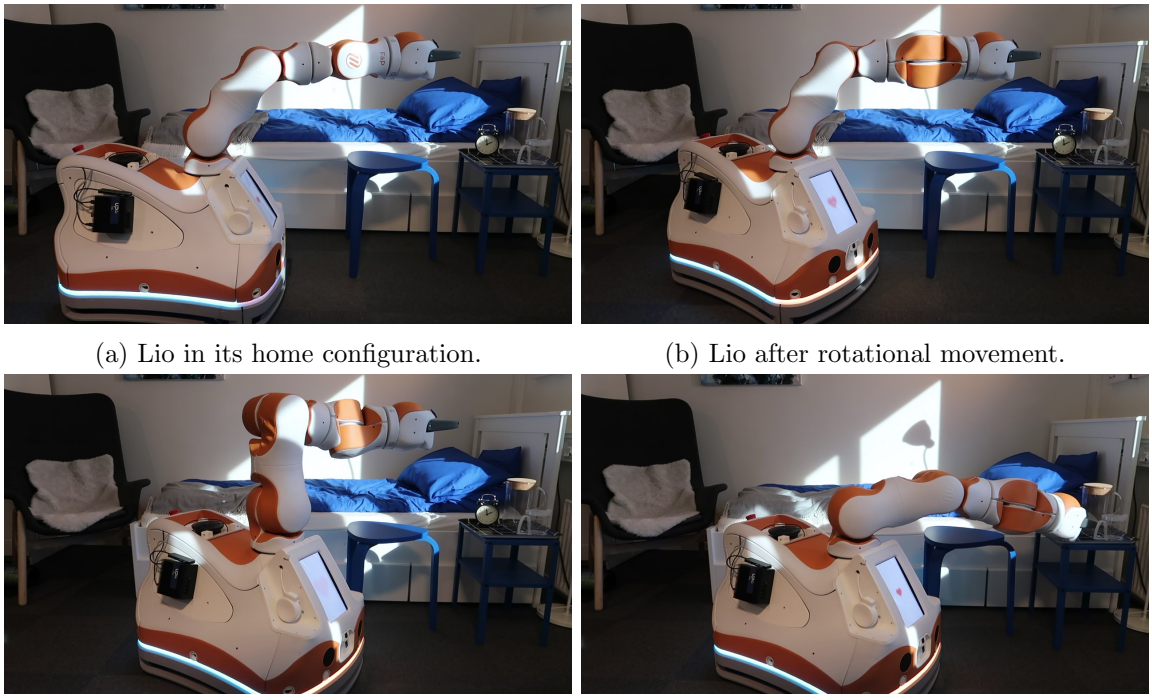
Figure 7.4: Platform movement to avoid obstacle.

7.3.2 Results

Performance in redundant mode

Figure 7.5 shows parts of Lio's path and configurations during Test 2. The upper left image shows Lio at its start configuration, the upper right image shows the pose after rotation, the bottom left image shows the pose after translational movement and the lower right image shows Lio at the desired pose, $\mathbf{T}_d$. One can see that the gripper pose is approximately maintained in the three first photos when the `redundant_mode` is activated.

The number of feasible configurations is a key factor for the performance in `redundant_mode`. Figure 7.6 shows how many configurations were available and which configurations were active during the test. From Section 4.2, eight solutions were derived from the analytic inverse kinematics problem. However, depending on the pose and due to the restricted joint limits of the robot, not all configurations are feasible at all times. The bar plot (left in Figure 7.6) shows that, at most, three configurations were feasible simultaneously. As the robot moved further away from its start pose, fewer configurations were available. The scatter plot to the right depicts which configuration was active. The two configurations



(a) Lio in its home configuration.

(b) Lio after rotational movement.

(c) Lio after translational movement

(d) Lio's end effector at the desired pose $\mathbf{T}_d$.

Figure 7.5: Lio's path of obstacle avoidance and reaching the desired pose in Test 2.

that were active during the test were `C6` and `C2`, defined in Figure 4.6. The data gap, present in the approximate time period $16 - 19s$, is caused by the configuration change. As explained in Section 6.2, the platform stops and waits for the manipulator joints to reach their reference values if a relative joint command is $\geq 30deg$. Such configuration changes may cause big deviations from the desired end effector pose, visible in Figure 7.7.

Figure 7.7 shows the deviation from the desired gripper position and rotation. Table 7.8 displays the corresponding median, mean and max value in addition to the end pose error of the gripper position in the x, y and z-direction. The "end pose error" refers to the error from the desired position when the platform has stopped moving. These deviations are of the same size as the errors in the static Test case. The max error in all four plots in Figure 7.7 occurs just after the configuration change ($16 - 19s$), as explained above. Apart from the time of the configuration change, the deviation from the desired gripper position and orientation is small ($\leq 8.4mm$). Comparing the platform path in Figure 7.4 to the error graphs, it is clear that the error is biggest in the platform's direction of motion. The error during the rotational movement is also noticeably bigger than during the translational movement. When the robot moves forward, there is a small offset in the x-direction but almost no deviation in the y-direction. The offset in the direction of motion is due to the arm joints compensating for the platform motion, resulting in a slight lag of about $8.4mm$. Since the robot is not moving vertically, there is approximately no deviation in the z-direction outside the configuration change.

Figure 7.9 shows the joint angle reference, outputted from the configuration control method

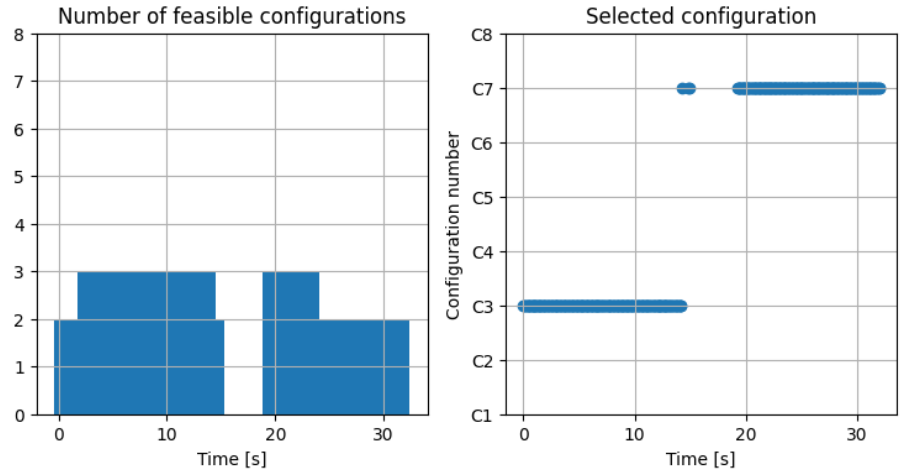


Figure 7.6: Number of feasible configurations (left) and selected configuration (right) during Test 2. The data gaps in time period $15 - 19s$ are due to the pause where the platform stops and waits for the arms joints to catch up with their reference angles.

(Algorithm 1), plotted against the actual joint positions. New angles are calculated every 10th millisecond unless the program is paused to wait for the joint motors after a $\geq 30deg$ step in the reference. The $180deg$ step in the reference for θ_4 and θ_6 is a result of the configuration change from `C6` to `C2`. The configuration change is necessary because θ_6 approaches its lower joint limit. The plots clearly demonstrate that when there is a large step in the reference signal, the robot arm is unable to follow the reference at its maximum velocity of $45deg/s$. This justifies the platform stop implemented in the `redundant_mode` to ensure that arm joints do not fall too far behind their reference. During gradual changes, the joints track the reference signal effectively.

There is an experimentally determined delay of approximately $1.5s$ from the moment the stop command is published to the `/cmd_vel` topic until the platform actually stops. The program pauses to track the platform position when the stop command is sent. Consequently, when the program resumes calculating new joint angles, the platform may have moved several cm from its previously registered position. This often results in a significant difference between the joint reference before and after a platform stop. This difference is clearly visible for θ_4 and θ_6 in Figure 7.9 as an approximate $20deg$ difference in the reference from $t = 16s$ to $t = 19s$. The arm joints will move to their last received joint command before the program resumes, which is observed as the slight overshoot in joint position at $t = 19s$.

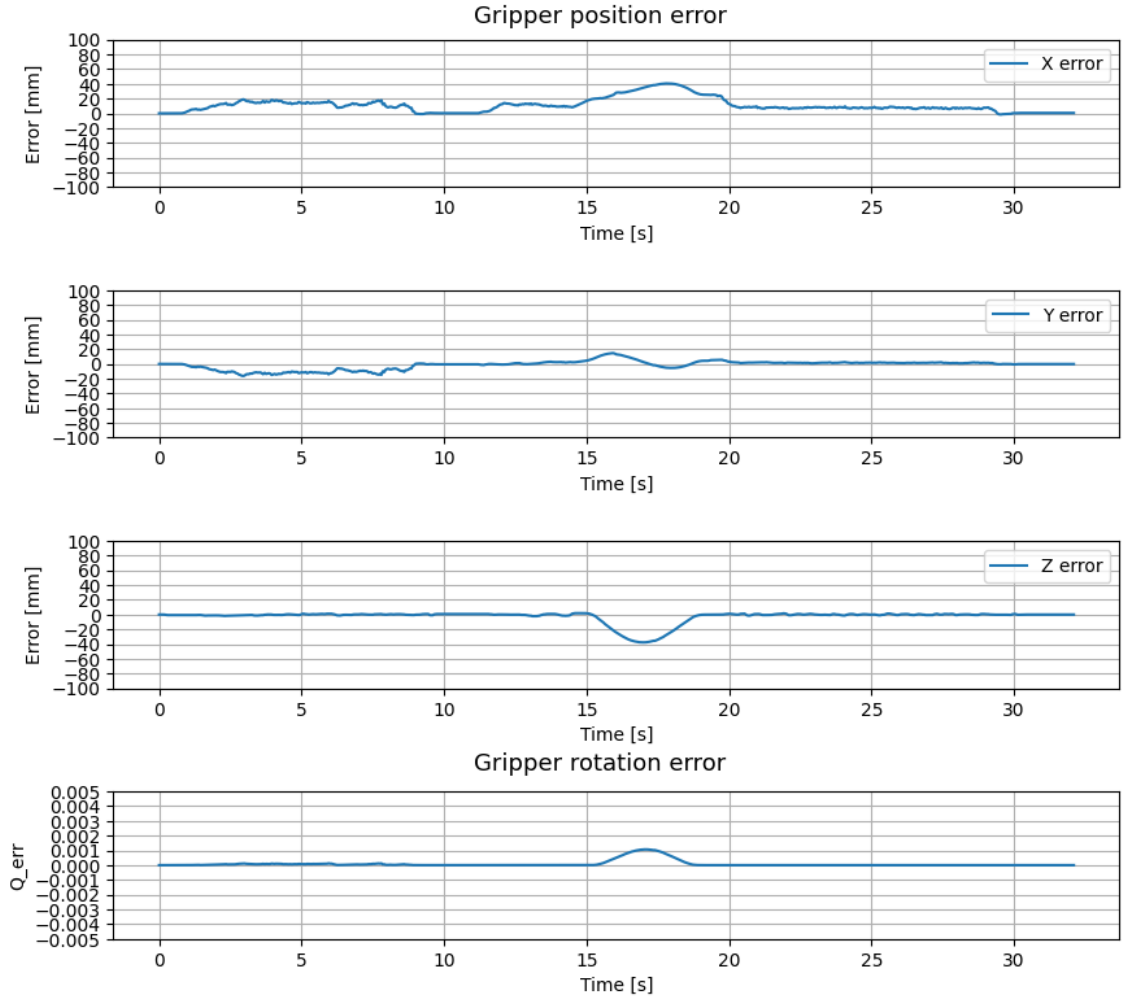


Figure 7.7: Deviation from desired gripper position (top) and desired gripper rotation (bottom).

	Median error [mm]	**Mean error [mm]**	**Max error [mm]**	**End pose error [mm]**
x	8.4	11.1	40.6	0.5
y	1.8	4.1	16.3	-0.1
z	0.6	3.12	37.8	0

Figure 7.8: Mean and median values of the gripper position error in the x-, y-, and z-direction.

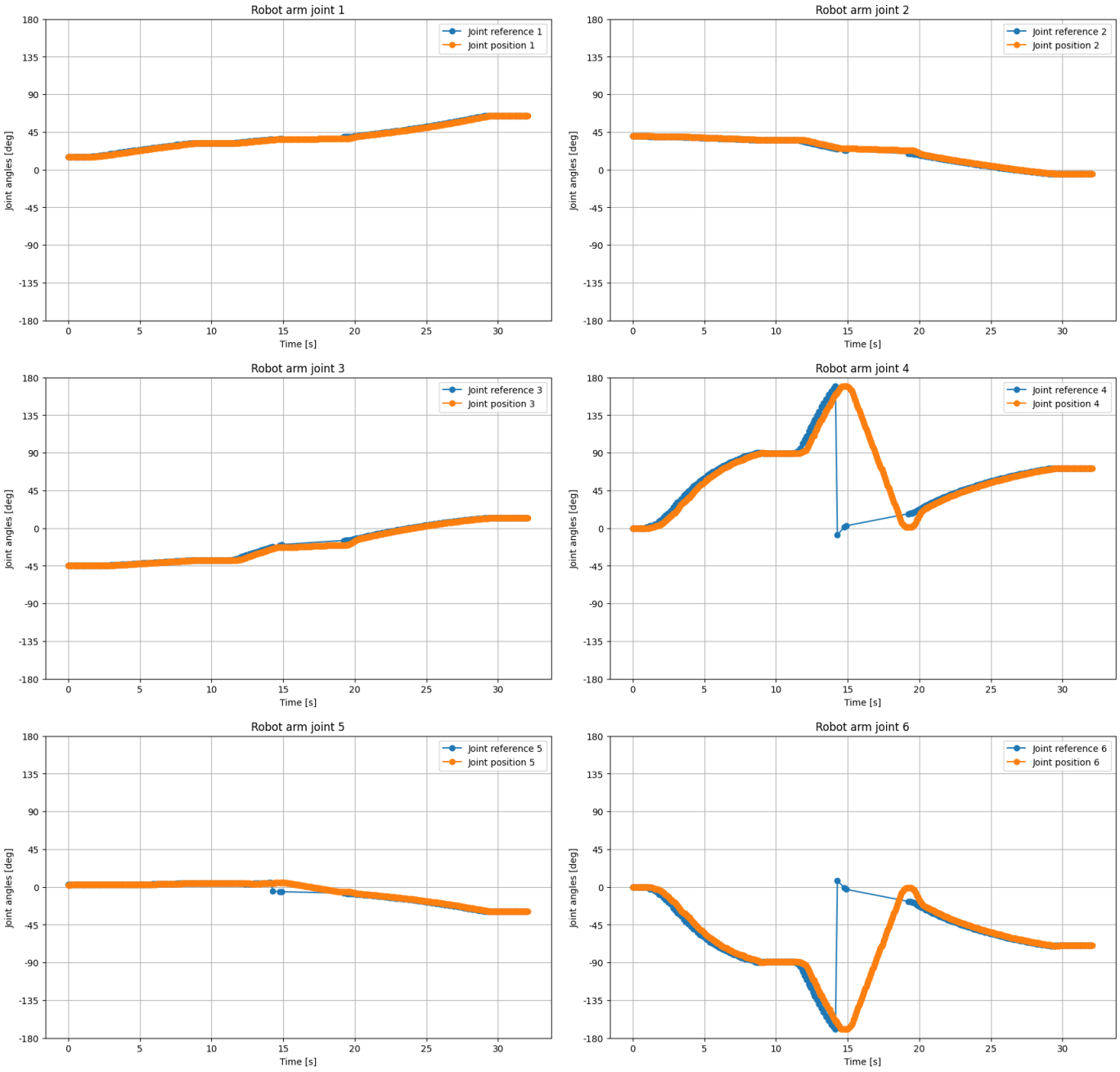


Figure 7.9: Reference joint value plotted against actual joint value for all the arm joints $\{\theta_1, \ldots, \theta_6\}$. Notice the $180deg$ jump in joint reference for θ_4 and θ_6 at the configuration change at $t \approx 15s$.

Performance in reaching the desired pose

When the `redundant_mode` was deactivated, the platform state was

$$\mathbf{q}_p = [273.6, 246.7, 12.1] \tag{7.19}$$

The analytic inverse kinematics found two feasible configurations `C1` and `C6`.

$$\begin{bmatrix} 0 & 35.6 & 110.4 & -133.5 & 3.6 & 37.4 & -2.7 \\ 1 & 35.6 & 65.4 & -46.5 & 154.6 & 5.1 & -154.6 \\ 0 & -144.4 & -65.4 & -133.5 & -25.4 & 5.1 & -154.6 \\ 0 & -144.4 & -110.4 & -46.5 & -176.4 & 37.4 & -2.7 \\ 0 & 35.6 & 110.4 & -133.5 & -176.4 & -37.4 & 177.3 \\ 1 & 35.6 & 65.4 & -46.5 & -25.4 & -5.1 & 25.4 \\ 0 & -144.4 & -65.4 & -133.5 & 154.6 & -5.1 & 25.4 \\ 0 & -144.4 & -110.4 & -46.5 & 3.6 & -37.4 & 177.3 \end{bmatrix} \tag{7.20}$$

With the joint command $\mathbf{q}_a = [35.6, 65.4, -46.5, 154.7, 5.1, -154.7]$, the final end effector configuration was

$$\mathbf{T}_{s_2t} = \begin{bmatrix} 0.62 & -0.76 & 0.17 & 797.0 \\ 0.74 & 0.65 & 0.18 & 943.9 \\ -0.25 & 0.01 & 0.97 & 545.7 \\ 0.00 & 0.00 & 0.00 & 1.0 \end{bmatrix} \tag{7.21}$$

which is equivalent to a position and orientation error of

$$\begin{aligned} \mathbf{p}_{err} &\approx 1.4mm \\ Q_{err} &\approx 1.14 \times 10^{-5} \end{aligned} \tag{7.22}$$

7.3.3 Test 3: Dynamic base positioning with double platform velocity

To assess the platform velocity's impact on the performance of the `redundant_mode`, Test 2 was conducted again with double platform velocity $v = 0.04m/s$, $\dot{\theta} = 0.1rad/s$. Apart from the increased platform velocity, all other parameters, including initial arm pose, platform path, and desired pose, remained unchanged. Only the graphs that show significant differences from Test 2 are displayed in the following section.

Results

Figure 7.12 and Table 7.13, present the gripper position and rotation error as well as error statistics for Test 3. The figures indicate that the deviations from the desired pose were significantly higher with the increased platform speed. The median and mean error in the x-direction more than doubled. Although the increase in the y and z-directions were not as big, they were still noticeably higher. The static error remains of the same size order as in Test 2 which is expected since this measurement was taken when the platform had stopped.

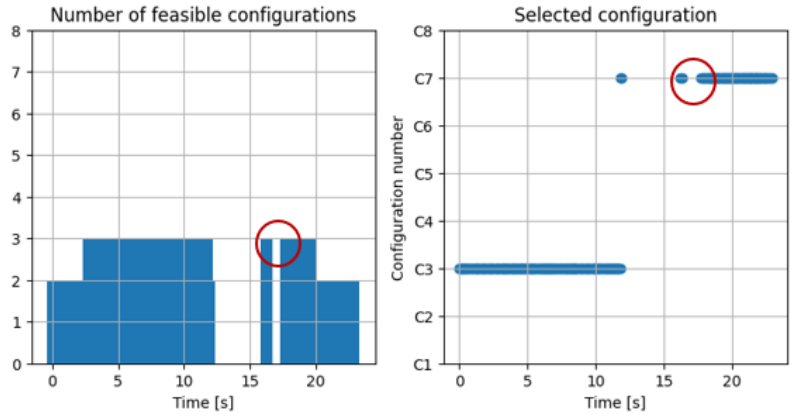


Figure 7.10: Number of feasible configurations (left) and selected configuration (right) during Test 3. Notice the second data gap not present in Test 2.

The configuration plot (Figure 7.10) shows the same number of feasible configurations and active configurations as in Test 2. However, there are now two data gaps, indicating that the platform has stopped twice. The second data gap is not due to a configuration change, but rather a step in the reference signal of $\geq 30deg$ for both θ_4 and θ_6. The big step is caused by the platform stop delay of $\approx 1.5s$. In Test 2 the platform drove slow enough to avoid a $30deg$ jump in joint reference. However, with double platform velocity in Test 3, the platform managed to move far enough offline to trigger another platform stop.

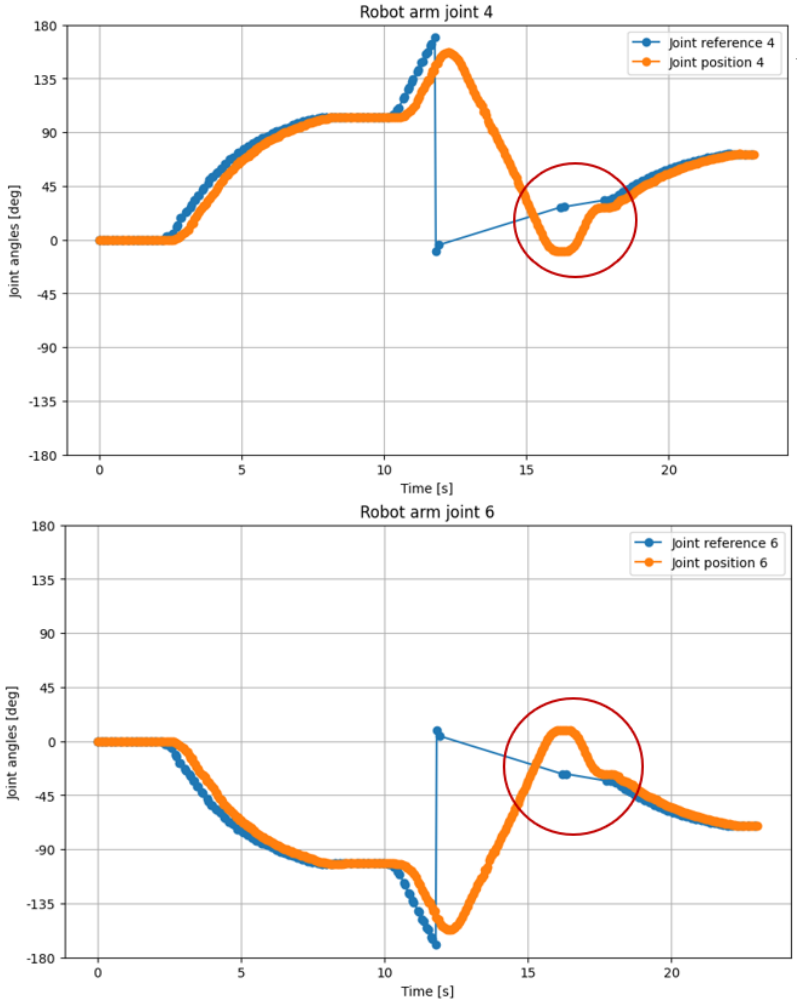


Figure 7.11: Reference joint value plotted against actual joint position for θ_4 and θ_6. Notice the second jump in the reference signal making the platform stop for the second time.

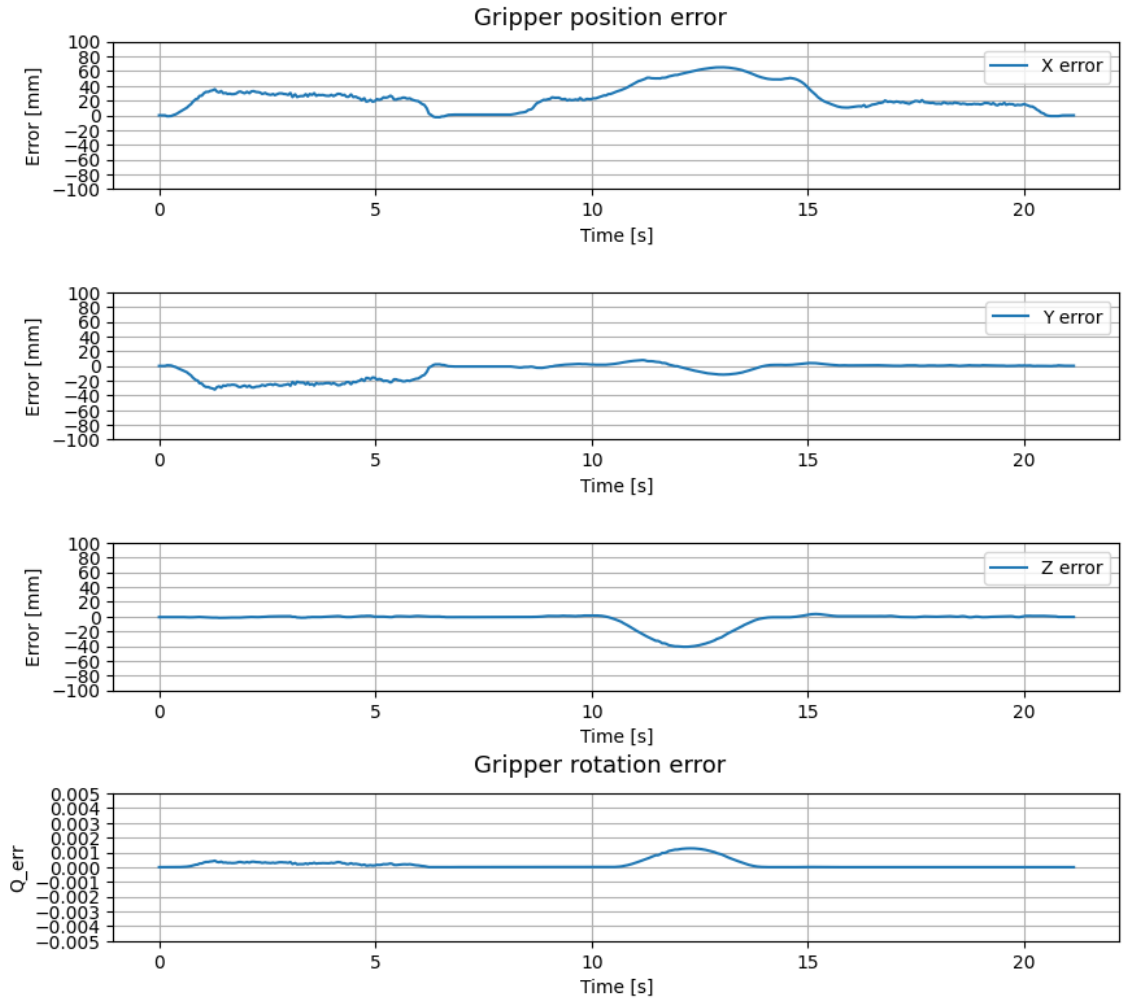


Figure 7.12: Deviation from desired gripper position (top) and desired gripper rotation (bottom) for the dynamic base Test case with double platform velocity.

	Median error [mm]	**Mean error [mm]**	**Max error [mm]**	**End pose error [mm]**
x	21.5	24.4	65.2	0.0
y	2.0	7.7	32.0	0.2
z	0.7	4.7	41.0	-0.5

Figure 7.13: Mean and median values of the gripper position error in the x-, y-, and z-direction for the Test case with double platform velocity.

7.4 Testing NLP for redundancy resolution

The objective of testing the optimization methods is to demonstrate that the platform's pose can be determined automatically, a crucial feature for future autonomous operations. Due to the current limitations in precise platform motion control accessible through ROS, these were conducted in simulations rather than experimentally on the robot. The results of the optimization methods were visualized in Matlab v2023b using a model of the Lio robot generated from a URDF file. The tests aimed to evaluate whether the optimization problems could find configurations where Lio achieved its desired pose while also fulfilling secondary tasks, such as minimizing platform movement and maximizing manipulability.

7.4.1 Test setup

Three tests were run to evaluate how the two optimization problems, referred to as NLP1 (5.2.1) and NLP2 (5.2.2), found different solutions to the inverse kinematics problem. To facilitate comparison, the tests were run with the same start configuration and a randomly chosen desired end effector pose, $\mathbf{T}_d$

$$\mathbf{T}_d = \begin{bmatrix} 0.88 & 0.00 & 0.48 & 2100.0 \\ 0.40 & 0.54 & -0.74 & 1100.0 \\ -0.26 & 0.84 & 0.47 & 700.0 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix} \tag{7.23}$$

The selected start configurations can be seen in Table 7.1 where $\mathbf{q}_{p0}$ and virtual joints $[\theta_p^*, x_p^*]$ both represent the platform's start pose and $\mathbf{q}_{a0}$ is the initial arm joint values. The start pose is represented both as virtual joint values $[\theta_p^*, x_p^*]$ and as position and orientation of the platform frame, $[x_p, y_p, \theta_p]$ because the NLP2 uses the eight-axis kinematic model considering the non-holonomic constraint, while the NLP1 method uses position and orientation of the platform frame directly. The platform's start pose was chosen such that the configuration of frame $\{p\}$ was reachable by one displacement of the virtual joints; in other words, $x_p^* = x_p$ and $\theta_p^* = \theta_p$. In Case 3 the start pose was chosen so that the desired pose was within reach of the robot without moving the platform.

Test case	$\mathbf{q}_{p0}$	$[\boldsymbol{\theta}_{p0}^*, \mathbf{x}_{p0}^*]$	$\mathbf{q}_{a0}$
1	[0.0, 1600.0, 90.0]	[90, 1600]	[-94.7, 0.0 0.0, 0.0, 0.0, 0.0]
2	[242.7, -176.3, -36.0]	[-36, 300]	[0.0, -90.0, 225.0, 90.0, 60.0, 0.0]
3	[1558.9, 900, 30]	[30, 1800]	[90.0, 60.0, -22.5, 90.0, 60.0, 0.0]

Table 7.1: The robot's start pose in the three different tests.

Test case	$\boldsymbol{\rho}_{opt}$	$\mathbf{q}_p$
1	[-72.1, 950.0, -11.1]	[1185.1, 989.9, 85.6]
2	[69.4, 950.0, 7.2]	[1382.2, 530.4, -37.6]
3	[-8.1, 688.6, -2.8]	[1558.9, 900.0, 30.0]

Table 7.2: Optimal platform configuration found with NLP1.

Test case	$[\boldsymbol{\theta}_p^*, \mathbf{x}_p^*]$	$\mathbf{q}_p$
1	[29.0, 1732.1]	[1514.5, 840.4, 29.0]
2	[11.8, 2134.0]	[2088.8, 436.9, 11.8]
3	[30, 1800]	[2088.8, 436.9, 11.8]

Table 7.3: Optimal platform configuration found with NLP2.

7.4.2 Results

The optimal redundancy parameters $\boldsymbol{\rho}_{opt} = [\rho_1, \rho_2, \rho_3]$ from NLP1 and the optimal virtual joints $[\theta_p^*, x_p^*]$ from NLP2 determines the platform configuration $\mathbf{q}_p$ and are shown in Table 7.2 and Table 7.3. Comparing the end poses of the two optimization methods to the initial pose, it is clear that NLP1 resulted in less displacement from the initial pose than NLP2. The end poses of NLP1 kept the heading angle close to the initial platform rotation, while the heading angle from NLP2 appears more arbitrary. Evaluating $\boldsymbol{\rho}_{opt}$ one can see that ρ_2, representing the extension of the manipulator in the $xy-$plane, was maximized in the two first Test cases where the robot started in a position where the desired end effector pose was out of reach.

The optimal arm configuration $\mathbf{q}_a$ found by the two optimization functions are shown in Table 7.4. NLP2 outputs the optimal arm joint angles directly with the numeric inverse kinematics, while NLP1 uses the configuration control method, inputting the state of the platform and using the analytic inverse kinematics solution. The feasible joint configurations from the analytic solution are shown in the configuration matrices (7.24a)-(7.24c). Figure 7.14 visualizes the optimal robot configurations for the three Test cases. The first column displays the output from NLP1 and the right column is the output from NLP2. The transparent robot model represents the start pose and the non-transparent robot represents the optimal end pose. The coordinate frame illustrates the desired end effector configuration, $\mathbf{T}_d$.

Test case	**Final arm joint positions, $\mathbf{q}_a$**	
	Analytic solution (NLP1)	Numeric solution (NLP2)
1	[-80.0, 60.8, -67.7, 40.8, 28.8, 18.6]	[-4.4, 11.8, 7.8, -0.8, -4.6, 61.3]
2	[74.6, 57.4, -61.5, -32.5, 22.6, 94.5]	[103.3, 72.1, -89.1, 104.3, -94.7, -45.7]
3	[-12.0, -0.3, 21.3, -47.1, -8.9, 105.6]	[103.3, 72.1, -89.1, 104.3, -94.7, -45.7]

Table 7.4: Optimal arm joint values maximizing accuracy and manipulability.

Figure 7.14 clearly shows that optimization algorithms converged to a solution where the robot succeeded in reaching its desired end effector pose. The deviation from the desired pose was

$$\begin{aligned} \mathbf{p}_{err} &\approx \quad 0.2mm \\ Q_{err} &\approx \quad 5.49 \times 10^{-8} \end{aligned}$$

with the analytic solution and

$$\begin{aligned} \mathbf{p}_{err} &\approx \quad 0.1mm \\ Q_{err} &\approx \quad 2.0 \times 10^{-8} \end{aligned}$$

with the numeric solution, which is practically zero in both cases. Table 7.5 shows the manipulability of the configurations as well as the distance from the initial and end platform pose. The manipulability found from NLP2 was always greater than the one of NLP1. For Test cases 2 and 3, it seems that NLP2 found the same local minimum since it had the same manipulability measure and platform pose. Evaluating the distance from the initial pose, it is clear that the configuration from NLP1 required less platform movement than from NLP2. For Test case 3, where the desired pose was within reach of the robot's start pose, NLP1 found a solution by only moving the manipulator while NLP2 moved the platform relatively far from the start pose.

The runtime and number of iterations needed for convergence for the two optimization algorithms are shown in Table 7.6. The analytical solution is much faster and requires far fewer iterations than the numeric solution. For Test case 3 the NLP1 doesn't need any iterations since the initial guess always checks if the current pose is within reach of the start pose.

Test case	**Manipulability**		$\|\|\mathbf{q_{b0}} - \mathbf{q}_p\|\|$	
	NLP1	NLP2	NLP1	NLP2
1	2.23×10^8	12.37×10^8	1332.9	1694.3
2	1.76×10^8	64.85×10^8	1340.8	1945.3
3	0.62×10^8	64.85×10^8	0	703.9

Table 7.5: Comparing manipulability of the configuration and platform movement for both optimization algorithms.

Test case	**Runtime [s]**		**Iterations**	
	NLP1	NLP2	NLP1	NLP2
1	0.025	0.523	9	33
2	0.030	1.837	9	107
3	0.002	0.723	0	44

Table 7.6: Comparing the runtime and number of iterations for the two optimization algorithms.

Test 1 configuration matrix:

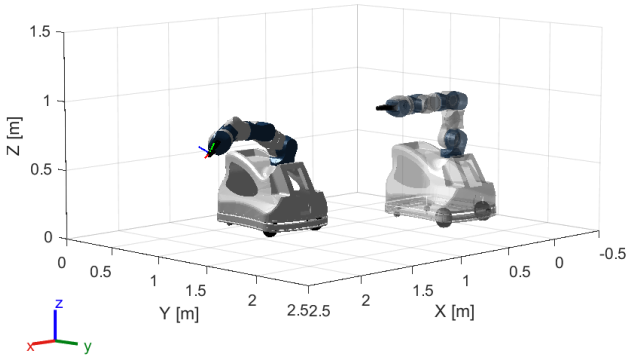
$$\begin{bmatrix} 1 & -80.0 & 83.8 & -112.3 & 25.4 & 47.1 & 37.7 \\ 1 & -80.0 & 60.8 & -67.7 & 40.8 & 28.8 & 18.6 \\ 1 & 100.0 & -60.8 & -112.3 & -139.2 & 28.8 & 18.6 \\ 1 & 100.0 & -83.8 & -67.7 & -154.6 & 47.1 & 37.7 \\ 1 & -80.0 & 83.8 & -112.3 & -154.6 & -47.1 & -142.3 \\ 1 & -80.0 & 60.8 & -67.7 & -139.2 & -28.8 & -161.4 \\ 1 & 100.0 & -60.8 & -112.3 & 40.8 & -28.8 & -161.4 \\ 1 & 100.0 & -83.8 & -67.7 & 25.4 & -47.1 & -142.3 \end{bmatrix} \tag{7.24a}$$

Test 2 configuration matrix:

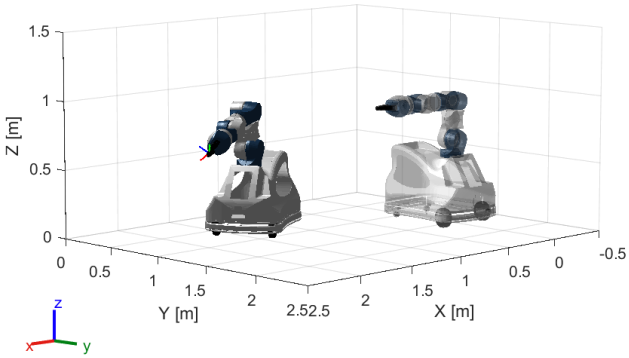
$$\begin{bmatrix} 1 & 74.6 & 86.8 & -118.5 & -16.2 & 48.0 & 75.0 \\ 1 & 74.6 & 57.4 & -61.5 & -32.5 & 22.6 & 94.5 \\ 0 & -105.4 & -57.4 & -118.5 & 147.5 & 22.6 & 94.5 \\ 0 & -105.4 & -86.8 & -61.5 & 163.8 & 48.0 & 75.0 \\ 1 & 74.6 & 86.8 & -118.5 & 163.8 & -48.0 & -105.0 \\ 1 & 74.6 & 57.4 & -61.5 & 147.5 & -22.6 & -85.5 \\ 0 & -105.4 & -57.4 & -118.5 & -32.5 & -22.6 & -85.5 \\ 0 & -105.4 & -86.8 & -61.5 & -16.2 & -48.0 & -105.0 \end{bmatrix} \tag{7.24b}$$

Test 3 configuration matrix:

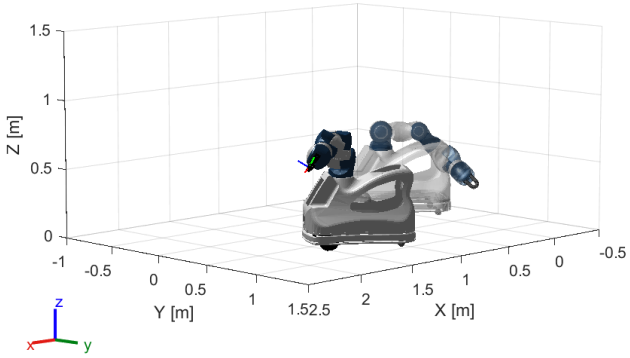
$$\begin{bmatrix} 0 & -12.0 & 116.6 & 158.7 & 6.5 & 99.7 & 60.1 \\ 1 & -12.0 & -0.3 & 21.3 & 132.9 & 8.9 & -74.4 \\ 0 & 168.0 & 0.3 & 158.7 & -47.1 & 8.9 & -74.4 \\ 0 & 168.0 & -116.6 & 21.3 & -173.5 & 99.7 & 60.1 \\ 0 & -12.0 & 116.6 & 158.7 & -173.5 & -99.7 & -119.9 \\ 1 & -12.0 & -0.3 & 21.3 & -47.1 & -8.9 & 105.6 \\ 0 & 168.0 & 0.3 & 158.7 & 132.9 & -8.9 & 105.6 \\ 0 & 168.0 & -116.6 & 21.3 & 6.5 & -99.7 & -119.9 \end{bmatrix} \tag{7.24c}$$



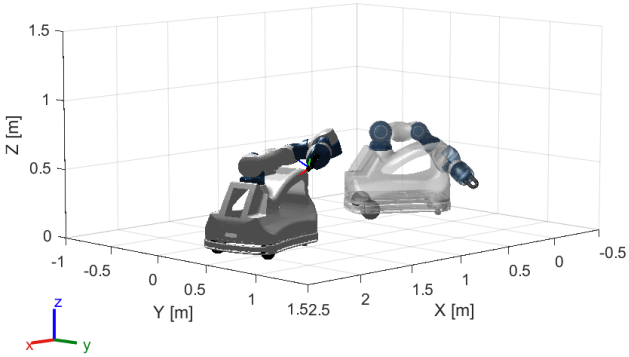
(a) Test case 1 with NLP1 minimizing platform movement.



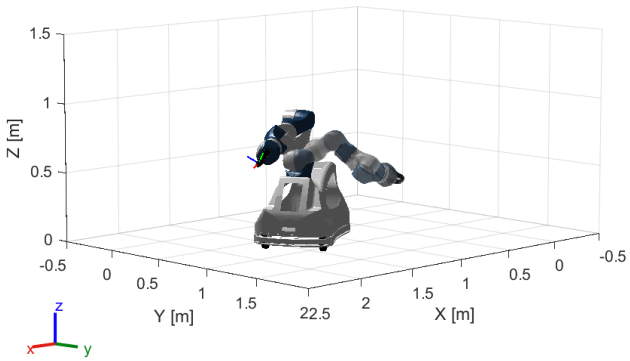
(b) Test case 2 with NLP2 maximizing manipulability.



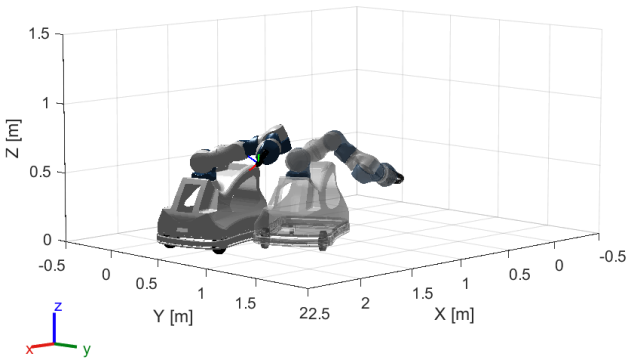
(c) Test case 2 with NLP1 minimizing platform movement.



(d) Test case 2 with NLP2 maximizing manipulability.



(e) Test case 3 with NLP1 minimizing platform movement.



(f) Test case 3 with NLP2 maximizing manipulability.

Figure 7.14: Visualization of the three Test cases with the initial configuration represented by the transparent robot and the "optimal" configuration represented by the non-transparent robot model. The coordinate frame at the robot's tooltip represents the location and orientation of the desired end effector pose.

Chapter 8

Discussion and future work

This chapter discusses the results of the experimental tests and simulations in Chapter 7. For the experimental tests, the test parameters like the platform speed and the resulting errors are seen in the context of Lio's intended operation. The optimization algorithms are evaluated based on their success in fulfilling secondary tasks and their relevance to real applications. Next, the system's limitations are described, and the chosen redundancy resolution and implementation methods are discussed. Lastly, ideas for future work are presented.

8.1 Evaluation of experimental test results

8.1.1 Test 1: Configuration control with static base positioning

The purpose of Test 1 was to assess the repeatability of the implemented kinematic robot model. According to the technical datasheet (F&P Robotics AG, 2022), the P-rob manipulator boasts a position repeatability of $\pm 0.1mm$. The kinematic model's measured position repeatability is slightly higher, $\mathbf{p}_{err} \approx 0.5mm$. The reason for the increased repeatability error is unknown, but can possibly originate from wear and tear on the hardware. Nonetheless, a repeatability error of $0.5mm$ remains more than sufficient for Lio's intended tasks. In the context of operating within a nursing home environment, navigating between rooms and handling daily items, this level of precision is more than adequate.

Evaluating the output of Q_{err} intuitively can be challenging due to its small scale. However, values on the order of $\leq 10^{-7}$ correspond to a rotation difference of about $0.001rad$ around one axis in Euler angle representation. Therefore, for practical application, the rotation error can be considered zero.

8.1.2 Test 2 and Test 3: Configuration control with dynamic base positioning

Test 2 aimed to evaluate to which degree the robot managed to keep a fixed end effector pose while the platform was moving. The characteristic of a redundant robot lies in its capability to achieve a desired end effector pose through an infinite number of joint configurations. The results obtained from the previous chapter confirmed that Lio, with the new kinematics model, operates as a redundant system. This conclusion is drawn from consistently maintaining the same end effector pose across various joint configurations. The average deviation from the desired pose was relatively low ($\leq 8.4mm$) with slow speed. However, the gripper position and rotation error plots in Figure 7.7 and 7.12 show that the end effector exhibited jerky motions. This was especially noticeable during rotational movements. Jerky motions were predicted already in the specialization project (Grøtterud, 2023) because the `reduandant_mode` relies on counteracting the platform movements. Therefore, the performance depends on factors such as the publicity rate of the joint state, the computational time of the inverse kinematics function, the joint angle velocity, and the platform velocity.

Test 3 highlighted the effect of platform velocity, showing a doubling of position error and an increase to two platform stops. Reducing the number of platform stops is desirable to achieve smoother robot motion. Increasing the threshold angle that triggers platform stops could improve performance at higher platform velocities. However, due to Lio's restricted joint range (6.1) the platform can not move far without requiring a reconfiguration of the arm. Nevertheless, Test 3 showed such high average errors that it can be concluded that the `redundant_mode` is suitable only for lower platform velocities. Fortunately, the robot is expected to operate at slow speeds while performing tasks in the nursing home.

8.2 Evaluation of NLP test results

The proposed optimization methods were intended as suggestions on how to solve automatic platform positioning. The test runs demonstrated that both algorithms successfully got Lio to the desired pose. Furthermore, in the two first Test cases, the NLP1 found solutions that maximized the redundancy parameter ρ_2, representing the robot arm extension in the xy-plane. Keeping the arm in such an extended position required a minimum of platform movements, which was the desired objective. On the other hand, NLP2 found configurations with more dexterous arm poses requiring bigger platform movements.

The IPOPT solver does not guarantee convergence to the global optimum. Therefore one could see that the NLP2 found different solutions with different initial guesses despite not including a relation to the initial arm pose in the cost function. A good initial guess can, however, increase the performance of the solvers. This was visible for Test case 3 when the desired pose was already within the reach of the manipulator. Here, the NLP1 found the global optimal solution of zero platform movement because its initial guess accounts for the scenario where the desired pose is within reach. The NLP2, on the other hand, converged to the same solution for Test case 2 and Test case 3. However, it would be preferable

to avoid unnecessary platform movements when the desired pose is already within reach. This could be accounted for by including the analytic inverse kinematic solution in the initial guess.

NLP1 had a significantly shorter runtime and required fewer iterations than NLP2. This is because the inverse kinematics and optimization solutions are separated. The optimization problem is only used to determine a platform pose which is then used to determine the robot arm's analytic inverse kinematics problem. Since the analytic inverse has a unique set of solutions, no iterations are needed to solve the joint angles of the arm. This is essentially the same approach as the configuration control method, only now the platform pose is found "automatically" and not determined by manual movement or navigation. NLP2, on the other hand, solves the inverse kinematics of the arm numerically and, therefore, has to iterate over both possible joint angle solutions and the cost of maximizing manipulability. Hence, it is no surprise that NLP2 is more computationally expensive than NLP1.

8.2.1 Advantages and drawbacks of the current problem formulations

As mentioned above, a great advantage of NLP1 is its computational efficiency following the use of the analytic inverse kinematics. However, incorporating more complex cost functions involving the arm joints $\mathbf{q}_a$ while using only $\boldsymbol{\rho}$ as a decision variable in the optimization problem, was proven difficult with the Casadi framework. This restriction limits the types of secondary tasks that can be added to the system, at least to the author's knowledge. This limitation led to the implementation of NLP2, as the manipulability expression could not be included in the cost without making $\mathbf{q}_a$ a decision variable.

Additionally, NLP1 does not consider the nonholonomic constraint that prevents the platform from moving sideways. Consequently, NLP1 may propose platform configurations that can not be reached by a single joint displacement of the virtual platform joints. This can possibly be solved by adding more constraints on the redundancy parameters $\boldsymbol{\rho}$. However, the nonholonomic constraint only reduces the dimensions of the system's achievable velocities, keeping the platform's configuration space three-dimensional. Therefore, another solution can be to develop a platform controller responsible for moving the robot to the desired platform pose generated by NLP1. Nevertheless, the problem formulation should likely be adjusted to avoid the platform having a difficult and jerky path to the desired platform pose.

A significant advantage of NLP2 is that it allows for easier implementation of secondary tasks that depend on the arm joints, like the manipulability index. Maximizing manipulability is a useful feature in, for example, collision avoidance because high manipulability means that the robot has more options as to which direction it can move to avoid obstacles (Arbo et al., 2019). The numeric inverse uses the eight-axis forward kinematic model derived in Section 4.3 to find the joint angles $\mathbf{q}_a$. Hence, it accounts for the platform's differential drive system by using the virtual joint representation to determine the platform pose. However, with the current formulation, only solutions that are within the reach of a single displacement of the virtual joints can be found. A method that allows for a

sequence of virtual joint displacements, moving the space frame $\{s\}$ to the location of the previous platform frame $\{p_{prev}\}$, would allow NLP2 to search amongst all the possible platform configurations.

Ideally, the two optimization methods could be used in real-time autonomous operations in the future. At the moment, NLP2 has too high runtime for real-time applications. Factors that could decrease the runtime include a better initial guess and increasing the acceptable error margins for the objective function. A useful initial guess could be, for example, to choose a platform location where the desired pose is within the manipulator's reach and then find the arm joints with the manipulator's analytic inverse kinematics solution. Moreover, the cost function should consider distance to the current configuration to avoid unnecessarily big movements like in Test case 3 and decrease the space of possible solutions. This can, for example, be implemented by multiplying the manipulability index with a scaling factor that punishes configurations far away from the current joint positions.

8.3 Limitations and notes on the chosen redundancy resolutions

8.3.1 Redundancy resolution at position level

The arm joints of the Lio robot were controllable only at the position level through the ROS connection, leading the inverse kinematics to be solved at the position level as well. The redundancy resolution could still be solved at the velocity level, with the joint displacements being derived from numerical integration. However, it was found more practical to exploit the analytic inverse kinematics function which directly computes joint displacements.

In (Siciliano, 1990), it was argued that the extended task space method, incorporating the analytic inverse kinematics when possible, was beneficial over other conventional redundancy resolutions such as the pseudoinverse and null space projection because it allows for the integration of user-specific secondary tasks. However, the lack of accurate motion control of the platform restricted the options for these secondary tasks. Despite the inability to control the platform precisely, the data concerning the platform's position and orientation was sufficiently accurate. Therefore, configuration control was deemed the most suitable approach for implementing a redundancy resolution method on the robot that could be tested experimentally and perform adequately. If access to the platform encoders had been available, more sophisticated redundancy resolution methods could have been explored. Moreover, in `redundant_mode`, the platform position could have been adjusted to enable the arm to reconfigure while maintaining a fixed end effector pose.

8.3.2 Challenges of limited motion control access

The lack of low-level motion control access, particularly on the platform, was the most significant limitation of this project. The provided control software for the Lio robot is

a web-based scripting environment for Python called myP. Designed for "regular users", myP is not suitable for research projects due to its poor GUI, lack of version control and limited debugging capabilities. Originally, access to a real-time low-level motion control module was anticipated, but ultimately not granted. Therefore, it was decided early on, after a discussion with a software developer from PPM Robotics, that programming in an IDE like VSCode and communicating with the robot control software through ROS was the best option for this project. However, the robot data and myP functions available through a ROS connection were very limited.

The Lio robot features advanced functionalities such as mapping the environment for autonomous navigation and bumper and floor sensors for collision avoidance, described in Section 1.2. Unfortunately, none of the sensor data or navigation functions were accessible through a ROS connection. Having access to a broader range of integrated robot functionalities could have enhanced the performance of the redundancy strategies developed in this thesis. However, there is a trade-off between having better control over the structure, debugging, and software architecture in a traditional IDE and accessing more integrated robot functions by programming in the web interface. This trade-off should be reconsidered and evaluated in future work.

8.4 Future work

This thesis has demonstrated the potential of implementing more advanced methods for utilizing a service robot with eight degrees of freedom. However, the performance of the implemented method could be improved by experimenting with different angle thresholds in the `redundant_mode` and exploring other options for secondary tasks in the extended task space method. Furthermore, the optimization methods need to be adjusted in order to be applicable in real-time control. Additionally, a robust and accurate platform controller must be developed to allow the implementation of redundancy resolution methods that use automatic platform positioning. To summarize, five topics are highlighted for future work.

- Implementing an interface to the internal position controllers of the platform, which has a higher sampling frequency related to the position references and the encoder measurements.
- The extended task space method can be further developed by testing other secondary tasks than configuration control. Potential secondary tasks can be, for example, controlling gripper orientation or the manipulator elbow position.
- The current optimization methods should be optimized for real-time control. The NLP1 should be modified to account for the nonholonomic constraint to ensure that the proposed configurations are feasible. NLP2 needs to decrease runtime which can be addressed by improving the initial guess and further tuning of the convergence criteria $\mathbf{e}_{tol}$. Moreover, the cost function can be adjusted to find more practical

robot poses by tuning the scaling factor α and accounting for the deviation from the current robot configuration.

- Obstacle avoidance is a feature that would be highly beneficial in task planning and autonomous operations in nursing homes. This can be efficiently implemented by including known obstacles as constraints in the optimization problem. Avoiding dynamic obstacles in real-time could be solved by accessing Lio's ultrasonic sensor and implementing a potential field as part of the cost functions.

- Gaining access to more of the robot's integrated sensors and functions could significantly improve the performance of the implemented redundancy resolutions. Therefore, further investigation is needed to explore options for accessing these features. Alternatively, a thorough reevaluation of the advantages and limitations of programming within the provided web interface compared to a traditional IDE should be conducted.

Chapter 9

Conclusion

In this thesis, an eight-axis kinematic model of the Lio robot has been derived, enabling the robot controller to utilize its redundant degrees of freedom. The kinematic model was validated through experimental testing, demonstrating minimal repeatability error. This accurate kinematic model has laid the groundwork for implementing redundancy resolution methods on the robot. Various approaches for solving the redundancy were explored, including extended task space with configuration control, optimization of redundancy parameters and maximizing manipulability with numerical inverse. The extended task space method with configuration control was deemed the most suitable given the current motion control limitations of the system.

To enable the robot to use more sophisticated redundancy resolution methods, an accurate platform controller is an absolute necessity. Therefore, accessing low-level control of the platform should be the first priority in future research. Once this is achieved, more options for secondary tasks in the extended task space formulation can be explored and the presented optimization methods can be implemented and tested on the robot. However, for the optimization methods to be ready for real-time applications, adjustments like accounting for the nonholonomic constraint in NLP1 and decreasing the runtime of NLP2 need to be considered.

In conclusion, with an accurate platform controller in place, the redundancy resolution methods that were modeled, implemented, and experimentally tested in this thesis:

Extended task space with configuration control,

Optimization of redundancy parameters for minimizing platform movement,

Maximizing manipulability with numerical inverse kinematics

demonstrated a significant potential for better utilizing redundancy in task programming and task execution in applications for service robotics in nursing homes.

Bibliography

Karoline Skobba Grøtterud. Service robot for nursing homes: kinematic modeling and motion planning for a mobile redundant robot. Project report, Norwegian University of Science and Technology, 2023.

Statisticks Norway. A historic shift: More elderly than children and teenagers, 2020. URL https://www.ssb.no/en/befolkning/artikler-og-publikasjoner/a-historic-shift-more-elderly-than-children-and-teenagers. (Accessed: 8 September 2023).

United Nations. The 17 goals, sustainable development, 2012. URL https://sdgs.un.org/goals. (Accessed: 10 November, 2023).

F&P Robotics AG. Lio - more time for human care, 2022. URL https://www.fp-robotics.com/en/lio/. (Accessed: 26 September 2023).

Justinas Mišeikis, Pietro Caroni, Patricia Duchamp, Alina Gasser, Rastislav Marko, Nelija Mišeikienė, Frederik Zwilling, Charles de Castelbajac, Lucas Eicher, Michael Früh, and Hansruedi Früh. Lio-a personal robot assistant for human-robot interaction and care applications. *IEEE Robotics and Automation Letters*, 5(4):5339–5346, 2020.

Ricarda Servaty, Annalena Kersten, Kirsten Brukamp, Ralph Möhler, and Martin Mueller. Implementation of robotic devices in nursing care. Barriers and facilitators: an integrative review. *BMJ Open*, 10(9), 2020.

Rong Huang. Classification of healthcare robots. In Hongxiu Li, Maehed Ghorbanian Zolbin, Robert Krimmer, Jukka Kärkkäinen, Chenglong Li, and Reima Suomi, editors, *Well-Being in the Information Society: When the Mind Breaks*, pages 115–123, Cham, 2022. Springer International Publishing.

Hocoma. Lokomat®, 2024. URL https://www.hocoma.com/us/solutions/lokomat/. (Accessed: 21 September 2023).

Paro. Paro - Advanced Therapeutic Robot, 2024. URL https://www.paroseal.co.uk/. (Accessed: 21 September 2023).

FUJI CORPORATION. Mobility support robot - Hug T1, 2024. URL https://www.fuji.co.jp/en/items/hug/hugt1. (Accessed: 21 September, 2023).

Camilla Cavendish. Robot carers for the elderly are now a reality in Japan. But do we want them here? *The Sunday Times*, 2018. URL https://www.thetimes.co.uk/article/robot-carers-for-the-elderly-are-now-a-reality-in-japan-but-do-we-want-them-here-mw8zpw0zd. (Accessed: 28 September, 2023).

Ahmed Ashraf Morgan, Jordan Abdi, Mohammed A. Q. Syed, Ghita El Kohen, Phillip Barlow, and Marcela P. Vizcaychipi. Robots in Healthcare: a Scoping Review. *Current Robotics Reports*, 3(4):271–280, 2022.

Evan Ackerman. Moxi Prototype from Diligent Robotics Starts Helping Out in Hospitals. *IEEE Spectrum*, 2018. URL https://spectrum.ieee.org/moxi-prototype-from-diligent-robotics-starts-helping-out-in-hospitals. (Accessed: 19 September, 2023).

PAL Robotics. TIAGo - Mobile Manipulator Robot, 2024. URL https://pal-robotics.com/robots/tiago/. (Accessed: 19 September, 2023).

Frauenhofer IPA. Care-O-bot 3, 2012. URL https://www.care-o-bot.de/en/care-o-bot-3.html. (Accessed: 20 September, 2023).

Ralf Kittmann, Tim Fröhlich, Johannes Schäfer, Ulrich Reiser, Florian Weisshardt, and Andreas Haug. Let me introduce myself: I am care-o-bot 4, a gentleman robot. In *Conference: Mensch und Computer 2015*, pages 223–232, Stuttgart, 2015.

Toyota UK Magazine. Toyota Human Support Robot: What is it and how can it be used?, 2021. URL https://mag.toyota.co.uk/toyota-human-support-robot/. (Accessed: 21 September, 2023).

Kinova Assistive. Robotic arm, 2024. URL https://assistive.kinovarobotics.com/product/jaco-robotic-arm. (Accessed: September 21, 2023).

ISO 13482:2014. Robots and robotic devices — safety requirements for personal care robots. Standard, International Organization for Standardization, 2014. URL https://www.iso.org/standard/53820.html.

Neziha Akalin, Annica Kristoffersson, and Amy Loutfi. Do you feel safe with your robot? factors influencing perceived safety in human-robot interaction based on subjective and objective measures. *International journal of human-computer studies*, 158, 2022.

Eftychios G. Christoforou, Sotiris Avgousti, Nacim Ramdani, Cyril Novales, and Andreas S. Panayides. The Upcoming Role for Nursing and Assistive Robotics: Opportunities and Challenges Ahead. *Frontiers in Digital Health*, 2, 2020.

RV Patel and F Shadpey. *Control of Redundant Robot Manipulators*. Springer Science and Business Media, 2005.

Bruno Siciliano. Kinematic control of redundant robot manipulators: A tutorial. *Journal of Intelligent and Robotic Systems*, 3(3):201–212, 1990.

Stefano Chiaverini, Giuseppe Oriolo, and Anthony A. Maciejewski. Redundant Robots. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 221–242. Springer International Publishing, Berlin, 2nd edition, 2016.

Mohamed Sorour, Andrea Cherubini, and Philippe Fraisse. Motion Control for Steerable Wheeled Mobile Manipulation. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–7, 2019.

Homayoun Seraji. A Unified Approach to Motion Control of Mobile Manipulators. *The International Journal of Robotics Research*, 17(2):107–118, 1998.

Roberto Ancona. Redundancy modelling and resolution for robotic mobile manipulators: a general approach. *Advanced Robotics*, 31(13):706–715, 2017.

Carlos Lopez-Franco, Jesús Hernández-Barragán, Alma Alanis, Nancy Arana-Daniel, and Michel López-Franco. Inverse kinematics of mobile manipulators based on differential evolution. *International Journal of Advanced Robotic Systems*, 15, 2018.

Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. John Wiley & Sons Inc, 2nd edition, 2019.

Olav Egeland and Jan Tommy Gravdahl. *Modeling and Simulation*. Marine Cybernetics, 2003.

Kevin M. Lynch and Frank C. Park. *Modern robotics: mechanics, planning, and control*. Cambridge university press, Cambridge, 2017.

Kenneth J. Waldron and James Schmiedeler. Kinematics. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 11–36. Springer International Publishing, Berlin, 2nd edition, 2016.

J.J. Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3993–3998, 2004.

R.S. Hartenberg and J. Denavit. *Kinematic Synthesis of Linkages*. McGraw-Hill series in mechanical engineering. McGraw-Hill, 1964.

Huan Weng and Kevin M. Lynch. Modernrobotics: Mechanics, planning and control code library, 2018. URL https://github.com/NxRLab/ModernRobotics/blob/master/doc/MRlib.pdf. (Accessed: 3 October, 2023).

Yoshihiko Nakamura. *Advanced robotics: redundancy and optimization*. Addison-Wesley Longman Publishing Co., Inc., 1990.

Donald Lee Pieper. *The kinematics of manipulators under computer control*. PhD thesis, Stanford University, Stanford, 1969.

Tsuneo Yoshikawa. Manipulability of Robotic Mechanisms. *The International Journal of Robotics Research*, 4(2):3–9, 1985. Publisher: SAGE Publications Ltd STM.

Farbod Fahimi. *Autonomous Robots: Modeling, Path Planning, and Control*. Springer US, Boston, MA, 2009.

Olav Egeland and Jens G. Balchen. Cartesian Control of a Spray-Painting Robot with Redundant Degrees of Freedom. *Modeling, Identification and Control: A Norwegian Research Bulletin*, 8(4):185–199, 1987.

Stephen J. Wright Jorge Nocedal. *Numerical Optimization*. Springer New York, NY, 2 edition, 2006.

LJ Langston. Mission planning, mission analysis and software formulation. level c requirements for the shuttle mission control center orbital guidance software. Technical report, 1976.

Mathias Hauan Arbo, Esten Ingar Grøtli, and Jan Tommy Gravdahl. CASCLIK: CasADi-Based Closed-Loop Inverse Kinematics, 2019. URL http://arxiv.org/abs/1901.06713.

F&P Robotics AG. *myP User Manual Version 1.7.0*, 2023. Unpublished internal document.

Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

Andreas Waechter and Carl Laird. Ipopt: Documentation, 2005. URL https://coin-or.github.io/Ipopt/. (Accessed: 17 April, 2024).

F&P Robotics AG. Technical specifications bas.c, 2022. Unpublished internal document.

Christopher E. Mower. Spatial casadi: A compact library for manipulating spatial transformations. 2023. URL https://github.com/cmower/spatial-casadi.

Appendix

A NLP1: minimizing platform movement with optimization of redundancy parameters

The code snippet below illustrates how the main functionality of NLP1 (5.2.1) was implemented in CasADi.

```
def init_guess(task_vec, q_p_current,):
    d = 113
    x_p, y_p, th_p = q_p_current[0], q_p_current[1], q_p_current[2]
    x_t, y_t = task_vec[0], task_vec[1]
    psi_t = task_vec[5]
    x_a = x_p-d*ca.cos(th_p)
    y_a = y_p-d*ca.sin(th_p)

    rho1 = ca.atan2((y_t-y_a),(x_t-x_a))-th_p
    rho2 = ca.sqrt((x_t-x_a)**2+(y_t-y_a)**2)
    rho3 = ca.atan2((y_t-y_a),(x_t-x_a)) - psi_t

    # Check if desired pose is within reach
    if rho2 > 950:
        rho2 = 900
        rho1 = 0
        rho3 = 0

    return ca.vertcat(rho1, rho2, rho3)

# Objective function minimizing platform movement
def obj(rho, task_vec, q_p_current):

    rho_1 = rho[0]
    rho_2 = rho[1]
    rho_3 = rho[2]

    x_t, y_t, z_t = task_vec[0], task_vec[1], task_vec[2]
    psi_t = task_vec[5]

    d = 113

    x_p = x_t - rho_2 *ca.cos(rho_3+psi_t)+d*ca.cos(rho_3+psi_t-rho_1)
    y_p = y_t - rho_2 *ca.sin(rho_3+psi_t)+d*ca.sin(rho_3+psi_t-rho_1)
    phi_p = rho_3+psi_t-rho_1

    alpha = ca.norm_2(ca.vertcat(q_p_current[0], \
                                 q_p_current[1])-ca.vertcat(x_t, y_t))

    return ca.norm_2(ca.vertcat(q_p_current[0], q_p_current[1], \
                                alpha*q_p_current[2])-ca.vertcat(x_p, y_p,
    alpha*phi_p))

# Symbolic representation of variables
rho = ca.MX.sym('rho', 3)
task_vec = ca.MX.sym('task_vec', 6)
```

```
q_p_current = ca.MX.sym("q_p_current", 3)
theta_arm = ca.MX.sym('theta_arm', 6)

p = ca.vertcat(task_vec, q_p_current)
inv_kin = ca.Function('f', [rho, task_vec], [inverse_kinematics(rho,
    task_vec)],\
                      ['rho', 'task_vec'], ['theta_arm'])
g = inv_kin(rho, task_vec)

prob = {'f': obj(rho, task_vec, q_p_current), 'x': rho, 'p': p, 'g': g}
solver = ca.nlpsol('solver', 'ipopt', prob)

p = [*task_vec, *q_p_current]
rho0 = init_guess(task_vec, q_p_current)

# Bounds on the redundancy parameters
lbx = [ca.pi/180*-80, 0, ca.pi/180*-114]
ubx = [ca.pi/180*260, 950, ca.pi/180*114]

# Joint limits
lbg = [ca.pi/180*-80, ca.pi/180*-109, ca.pi/180*-205, \
       ca.pi/180*-169, ca.pi/180*-114, ca.pi/180*-170]
ubg = [ca.pi/180*260, ca.pi/180*110, ca.pi/180*25, \
       ca.pi/180*170, ca.pi/180*114, ca.pi/180*169]

solution = solver(x0=rho0, lbx = lbx, ubx = ubx, p = p, lbg = lbg, ubg =
    ubg)
```

B NLP2: maximizing manipulability and solving the inverse kinematics numerically

The code snippet below illustrates how the main functionality of NLP2 (5.2.2) was implemented in CasADi. CasAdi does not yet have functionality for calculating the determinant of a symbolic matrix. Therefore the determinant was computed with Laplace extension with the functions `minor(matrix, i,j)` and `determinant(matrix)` The package `spatial-casadi` (Mower, 2023) was used to manipulate spatial transformation inside the `pose_err` function.

```
def minor(matrix, i, j):
    n = matrix.size1()
    assert matrix.size1() == matrix.size2()

    minor_matrix = ca.MX.zeros(n-1, n-1)

    minor_row = 0
    for row in range(n):
        if row == i:
            continue
        minor_col = 0
        for col in range(n):
            if col == j:
                continue
            minor_matrix[minor_row, minor_col] = matrix[row, col]
            minor_col += 1
        minor_row += 1
    return minor_matrix

def determinant(matrix):
    n = matrix.size1()
    assert matrix.size1() == matrix.size2()

    if n == 2:
        return matrix[0, 0]*matrix[1, 1] - matrix[0, 1]*matrix[1, 0]

    det = 0
    for col in range(n):
        cofactor = ((-1)**col) * matrix[0, col] * determinant(minor(matrix,
 0, col))
        det += cofactor
    return det

def M(th):
    J = ca.jacobian(fkin(th),th)
    man = J@ca.transpose(J)

    # Compute the determinant using the Laplace expansion
    det = determinant(man)
    return -ca.sqrt(det)

# Objective function minimizing platform movement
def obj(th,task_vec):
    fq = fkin(th)
    alpha = 0.00001
```

```
    return 1/2*ca.transpose(fq-task_vec)@(fq-task_vec) + alpha*M(th)

def pose_err(th, task_vec):
    fq = fkin(th)
    pos_err = ca.norm_2(fq[0:3]-task_vec[0:3])
    fkin_quat = R.from_euler("xyz", ca.vertcat(fq[3], fq[4], fq[5])).
    as_quat()
    task_quat = R.from_euler("xyz", ca.vertcat(task_vec[3], task_vec[4],
    task_vec[5])).as_quat()
    q_err = 1-ca.fabs(ca.dot(fkin_quat, task_quat))
    return ca.vertcat(pos_err, q_err)

# Symbolic representation of variables
th = ca.MX.sym("th", 8)
task_vec = ca.MX.sym("task_vec", 6)
q_mm_current = ca.MX.sym("q_mm_current", 8)

p = task_vec
g = pose_err(th, task_vec)

prob = {'f': obj(th, task_vec), 'x': th, 'p': p, 'g': g}
solver = ca.nlpsol('solver', 'ipopt', prob)

lbx = [-ca.pi, -5000, ca.pi/180*-80, ca.pi/180*-109, ca.pi/180*-205, \
       ca.pi/180*-169, ca.pi/180*-114, ca.pi/180*-170]
ubx = [ca.pi, 5000, ca.pi/180*260, ca.pi/180*110, ca.pi/180*25, \
       ca.pi/180*170, ca.pi/180*114, ca.pi/180*169]

lbg = [0, 0]
ubg = [0.1, 0.00000001]

p = task_vec
th0 = q_mm_current
solution = solver(x0=th0, lbx = lbx, ubx = ubx, p=p, lbg=lbg, ubg=ubg)
```

C Symbolic forward kinematics of the eight-axis kinematic model

This section shows the symbolic expression of Lio's eight-axis forward kinematics where $\{x, y, z\}$ correspond to the position of the end effector and $\{\phi, \theta, \psi\}$ corresponds to the XYZ-fixed angle representation of the end effector's orientation.

$\{\theta_1, ..., \theta_6\}$ is the manipulator joints and $\{\theta_p^*, x_p^*\}$ is the platform's virtual joints.

$$
\begin{aligned}
x = {} & (l_5 + l_6)\Bigg(\sin(\theta_5)\Bigg(\cos(\theta_p^* + \theta_1 + \frac{\pi}{2}) \sin(\theta_4) - \cos(\theta_2 + \theta_3 - \frac{\pi}{2}) \sin(\theta_p^* + \theta_1 + \frac{\pi}{2}) \cos(\theta_4) \Bigg) \\
& - \sin(\theta_p^* + \theta_1 + \frac{\pi}{2}) \sin(\theta_2 + \theta_3 - \frac{\pi}{2}) \cos(\theta_5) \Bigg) - \cos(\theta_p^*)(d_{\text{ap}} - x_p^*) \\
& + l_2 \sin(\theta_p^* + \theta_1 + \frac{\pi}{2}) \cos(\theta_2 - \frac{\pi}{2}) - \sin(\theta_p^* + \theta_1 + \frac{\pi}{2}) \sin(\theta_2 + \theta_3 - \frac{\pi}{2})(l_3 + l_4)
\end{aligned}
$$

$$
\begin{aligned}
y = {} & (l_5 + l_6)\Bigg(\sin(\theta_5)\Bigg(\sin(\theta_p^* + \theta_1 + \frac{\pi}{2}) \sin(\theta_4) + \cos(\theta_p^* + \theta_1 + \frac{\pi}{2}) \cos(\theta_2 + \theta_3 - \frac{\pi}{2}) \cos(\theta_4) \Bigg) \\
& + \cos(\theta_p^* + \theta_1 + \frac{\pi}{2}) \sin(\theta_2 + \theta_3 - \frac{\pi}{2}) \cos(\theta_5) \Bigg) - \sin(\theta_p^*)(d_{\text{ap}} - x_p^*) \\
& - l_2 \cos(\theta_p^* + \theta_1 + \frac{\pi}{2}) \cos(\theta_2 - \frac{\pi}{2}) + \cos(\theta_p^* + \theta_1 + \frac{\pi}{2}) \sin(\theta_2 + \theta_3 - \frac{\pi}{2})(l_3 + l_4)
\end{aligned}
$$

$$
\begin{aligned}
z = {} & l_1 + p_{\text{height}} - l_2 \sin(\theta_2 - \frac{\pi}{2}) - \cos(\theta_2 + \theta_3 - \frac{\pi}{2})(l_3 + l_4) \\
& - \Bigg(\cos(\theta_2 + \theta_3 - \frac{\pi}{2}) \cos(\theta_5) - \sin(\theta_2 + \theta_3 - \frac{\pi}{2}) \cos(\theta_4) \sin(\theta_5) \Bigg)(l_5 + l_6)
\end{aligned}
$$

$$
\begin{aligned}
\phi = {} & \arctan 2\Bigg(\cos(\theta_6)\Bigg(\cos(\theta_5)\Bigg(\sin(\theta_p^* + \theta_1 + \frac{\pi}{2}) \sin(\theta_4) + \cos(\theta_p^* + \theta_1 + \frac{\pi}{2}) \cos(\theta_2 + \theta_3 - \frac{\pi}{2}) \cos(\theta_4) \Bigg) \\
& - \cos(\theta_p^* + \theta_1 + \frac{\pi}{2}) \sin(\theta_2 + \theta_3 - \frac{\pi}{2}) \sin(\theta_5) \Bigg) + \sin(\theta_6)\Bigg(\sin(\theta_p^* + \theta_1 + \frac{\pi}{2}) \cos(\theta_4) \\
& - \cos(\theta_p^* + \theta_1 + \frac{\pi}{2}) \cos(\theta_2 + \theta_3 - \frac{\pi}{2}) \sin(\theta_4) \Bigg), \sin(\theta_2 + \theta_3 - \frac{\pi}{2}) \sin(\theta_4) \sin(\theta_6) \\
& - \cos(\theta_6)\Bigg(\cos(\theta_2 + \theta_3 - \frac{\pi}{2}) \sin(\theta_5) + \sin(\theta_2 + \theta_3 - \frac{\pi}{2}) \cos(\theta_4) \cos(\theta_5) \Bigg) \Bigg)
\end{aligned}
$$

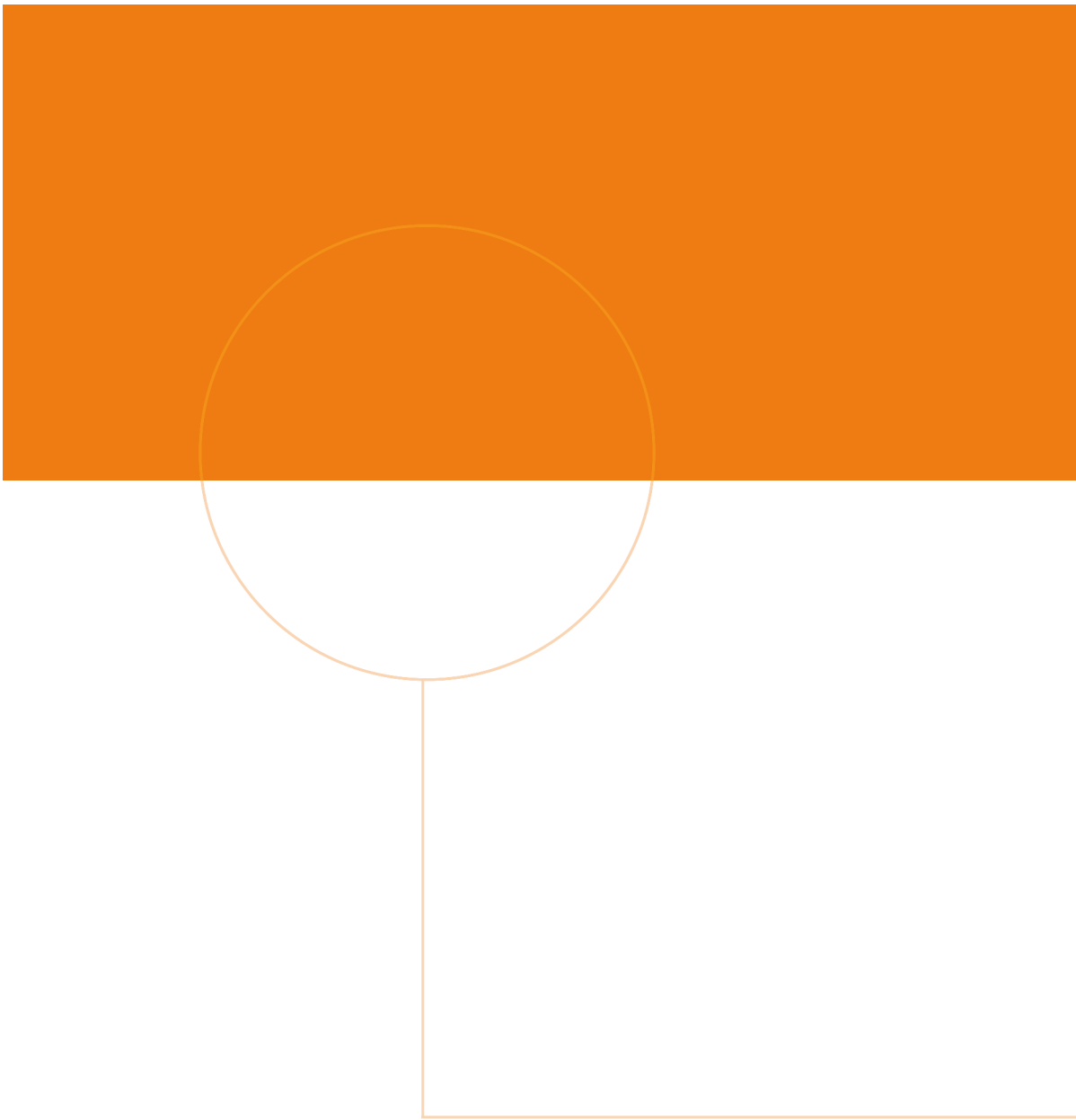
$$\theta = -\arcsin\Bigg(\cos(\theta_6)\Bigg(\cos(\theta_5)\Bigg(\cos(\theta_p^* + \theta_1 + \frac{\pi}{2})\sin(\theta_4) - \cos(\theta_2 + \theta_3 - \frac{\pi}{2})\sin(\theta_p^* + \theta_1 + \frac{\pi}{2})\cos(\theta_4)\Bigg) + \sin(\theta_p^* + \theta_1 + \frac{\pi}{2})\sin(\theta_2 + \theta_3 - \frac{\pi}{2})\sin(\theta_5)\Bigg) + \sin(\theta_6)\Bigg(\cos(\theta_p^* + \theta_1 + \frac{\pi}{2})\cos(\theta_4) + \cos(\theta_2 + \theta_3 - \frac{\pi}{2})\sin(\theta_p^* + \theta_1 + \frac{\pi}{2})\sin(\theta_4)\Bigg)\Bigg)$$

$$\psi = \arctan 2\Bigg(\sin(\theta_6)\Bigg(\cos(\theta_5)\Bigg(\cos(\theta_p^* + \theta_1 + \frac{\pi}{2})\sin(\theta_4) - \cos(\theta_2 + \theta_3 - \frac{\pi}{2})\sin(\theta_p^* + \theta_1 + \frac{\pi}{2})\cos(\theta_4)\Bigg) + \sin(\theta_p^* + \theta_1 + \frac{\pi}{2})\sin(\theta_2 + \theta_3 - \frac{\pi}{2})\sin(\theta_5)\Bigg) - \cos(\theta_6)\Bigg(\cos(\theta_p^* + \theta_1 + \frac{\pi}{2})\cos(\theta_4) + \cos(\theta_2 + \theta_3 - \frac{\pi}{2})\sin(\theta_p^* + \theta_1 + \frac{\pi}{2})\sin(\theta_4)\Bigg), \sin(\theta_5)\Bigg(\cos(\theta_p^* + \theta_1 + \frac{\pi}{2})\sin(\theta_4) - \cos(\theta_2 + \theta_3 - \frac{\pi}{2})\sin(\theta_p^* + \theta_1 + \frac{\pi}{2})\cos(\theta_4)\Bigg) - \sin(\theta_p^* + \theta_1 + \frac{\pi}{2})\sin(\theta_2 + \theta_3 - \frac{\pi}{2})\cos(\theta_5)\Bigg)$$

D Robot link dimensions

The link dimensions were extracted from the robot URDF file.

Parameter	**Length [mm]**
l_b	212
l_1	150.5
l_2	290
l_3	164.5
l_4	145.5
l_5	129
l_6	46
l_g	241
d_{ap}	113
p_{height}	265.5



NTNU

Norwegian University of
Science and Technology