Optimized design, analysis and kinematic control of highly redundant serial robotic arms

Angelica Ginnante

MEMOIRE DE DOCTORAT DE

L'ECOLE CENTRALE DE NANTES
ET L'UNIVERSITÉ DE GÊNES

ECOLE DOCTORALE N° 602
Sciences de l'Ingénierie et des Systèmes
Spécialité : Robotique

Par
Angelica GINNANTE

Optimized design, analysis and kinematic control of highly redundant serial robotic arms

Projet de recherche doctoral présenté et soutenu à Nantes, le 17/01/2024
Unité de recherche : UMR 6004, Laboratoire des Sciences du Numérique de Nantes (LS2N)

Rapporteurs avant soutenance :

Lucia PALLOTTINO	Associate Professor, Université de Pise, Italie
David DANEY	Directeur de Recherche, INRIA Bordeaux

Composition du Jury :

Président :	Philippe WENGER	Directeur de Recherche, CNRS, École Centrale Nantes
Examinateurs :	Lucia PALLOTTINO	Associate Professor, Université de Pise, Italie
	David DANEY	Directeur de Recherche, INRIA Bordeaux
	Giorgio CANNATA	Full Professor, Université de Gênes, Italie
Directeur de recherches doctorales :	Stéphane CARO	Directeur de Recherche, CNRS, École Centrale Nantes
Directeur de recherches doctorales :	Enrico SIMETTI	Associate Professor, Université de Gênes, Italie
Co-enc. de recherches doctorales :	François LEBORNE	Ingénieur de Recherche, Nimbl'Bot, Canéjan

Design, analysis and kinematic control of highly redundant serial robotic arms

Doctorate Thesis

Candidate: Angelica Ginnante

Supervisors: Stéphane Caro
Enrico Simetti
François Leborne

Jury: Lucia Pallottino
David Daney
Philippe Wenger
Giorgio Cannata

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

CAD	Computer-Aided Design
CNC	Computer Numerical Control
Opt-SNS	Optimal Saturation in the Null Space
PKM	Parallel Kinematic Mechanism
RP	Reverse Priority
RTR	Robot Transmission Ratio
SE(N)	N-dimensional Euclidean motion group
SNS	Saturation in the Null Space
TCP	Tool contact point
T&A	Tilt & Azimuth
T&T	Tilt & Torsion
TPIK	Task Priority Inverse Kinematics

Nomenclature

Robot design parameters

α	Slope of *Tube 1* and *Tube 2*
β	Angular offsets of the robotic arm
β_i	Optimized angular offset i
l	Links length of the robotic arm
l_i	Optimized link length i
r	Distance between *Platform 1* to the constant velocity joint and between the constant velocity joint to *Platform 2*
$\boldsymbol{\zeta}$	Design parameters

Robot variables

$\overline{\mathrm{CoM}}$	Mean center of mass distance
$\mathcal{F}$	Frame
L	Characteristic length
m	Dimension of the task space
n	Dimension of the joint space
n_r	Dimension of the real joint space
n_v	Dimension of the virtual joint space
$\boldsymbol{\omega}$	Angular velocity
q_1	Angular position of the first motor in NB-module
q_2	Angular position of the second motor in NB-module
$\mathbf{q}$	Joint position vector of robot
$\mathbf{q}_r$	Joint position vector of real joints
$\mathbf{q}_v$	Joint position vector of virtual joints
$\mathbf{p}$	Translation vector

$\dot{\mathbf{p}}$	Linear velocity
ϕ	Azimuth angle
$\mathbf{R}$	Rotation matrix
σ	Torsion angle
θ	Tilt angle
$\mathbf{t}$	Tip twist, $[\dot{\mathbf{p}}^\top, \boldsymbol{\omega}^\top]^\top$
$\mathbf{T}$	Homogeneous transformation matrix
$\mathbf{t}_d$	Target end-effector twist
$\vec{v}$	Displacement vector

Jacobian matrices

$\mathbf{J}$	General kinematic Jacobian matrix
Δ	Determinant of the kinematic Jacobian matrix
$\mathbf{J}_1$	Kinematic Jacobian matrix of NB-module from $\mathbf{t}$ to $[\dot{\phi}, \dot{\theta}]^\top$
$\mathbf{J}_2$	Kinematic Jacobian matrix of NB-module from $[\dot{\phi}, \dot{\theta}]^\top$ to $\dot{\mathbf{q}}$
$\mathbf{J}_3$	Reduced kinematic Jacobian matrix of NB-module for kinematic analysis
$\mathbf{J}_e$	Kinematic Jacobian matrix of robot end-effector
$\mathbf{J}_{\mathrm{NB}}$	NB-module kinematic Jacobian matrix
$\mathbf{J}_{\mathrm{task}}$	Jacobian matrix related to one task
$\mathbf{J}_k$	Jacobian matrix of k^{th} task vector
$\mathbf{J}_\eta$	Dexterity Jacobian matrix
$\mathbf{J}_\mu$	Manipulability Jacobian matrix
$\mathbf{J}_\rho$	RTR Jacobian matrix
$\mathbf{J}_w$	Weighted kinematic Jacobian matrix using the characteristic length L

Task priority algorithm variables

$\mathscr{A}$	Action
$a^i(\mathrm{x})$	Task activation functions
$a^P(\mathbf{P})$	Action activation function
$\mathbf{A}_k$	Diagonal matrix of activation functions for k^{th} task vector
γ	Positive gain
E	Equality task
I	Inequality task

λ	Positive gain related to target convergence rate in TPIK algorithm
$\mathbf{P}$	Previous and current executed actions and time elapsed in current step
S_k	Solution at the k^{th} task
S_{k-1}	Manifold of solutions of the previous tasks before the k^{th} one
$\mathbf{x}$	Task vector
$\mathrm{x}(\mathbf{q})$	Robot configuration dependent scalar variable
$(\mathrm{x}_m,\mathrm{x}_M)$	Inequality control objective thresholds
x^*	Desired goal for $\mathrm{x}(\mathbf{q})$
$\dot{\mathrm{x}}^*$	Changing rate associated to x^*
$\dot{\bar{\mathrm{x}}}$	Feedback reference rate
$\dot{\bar{\mathbf{x}}}_k$	Vector of all reference rates of scalar control tasks

Kinetostatic indices

κ	Conditioning number
η	Dexterity
μ	Manipulability
ν	Bounded manipulability
ρ	Robot transmission ratio
$\epsilon(\eta, \nu, \rho)$	Linear combination of η, ν and ρ
H	Global conditioning index

Axis names

x	Axis x
$\vec{x}$	Unit vector axis x
y	Axis y
$\vec{y}$	Unit vector axis y
z	Axis z
$\vec{z}$	Unit vector axis z

Trajectory variables

$\vec{\mathbf{f}}$	Vector of forces
$\vec{\mathbf{v}}$	Vector of velocities

Workspace surface plot

A, B, C	Node grids
d	Distance
s	Positive or negative sign
R_1, R_2	Circumference rays
(x_J, y_J, z_J)	Node coordinates of grid J

General variables

$\mathbf{0}_{1\times3}$	Vector of zeros
δ	Threshold kinetostatic index velocity
$\delta_{\text{kinematic}}$	Threshold kinematic index velocity
δ_{velocity}	Threshold end-effector velocity
k	Counter variable
m_d	Number of optimized designs
m_r	Number of repetitions
(λ_1,λ_2)	Scaling factors
p	Number of trajectories
t	Time instant

Symbols

$\gg$	Much greater operator
$\triangleq$	Define operator

Considering a general variable a.

$\overline{a}$	Mean value of a
$\hat{a}$	$0.5\ a_{\min} + 0.25\ (\overline{a} + a_{\max})$
$a_{\max}$	Max value reached by a
$a_{\min}$	Min value reached by a
$\dot{a}$	Dexterity rate

Chapter 1

Introduction

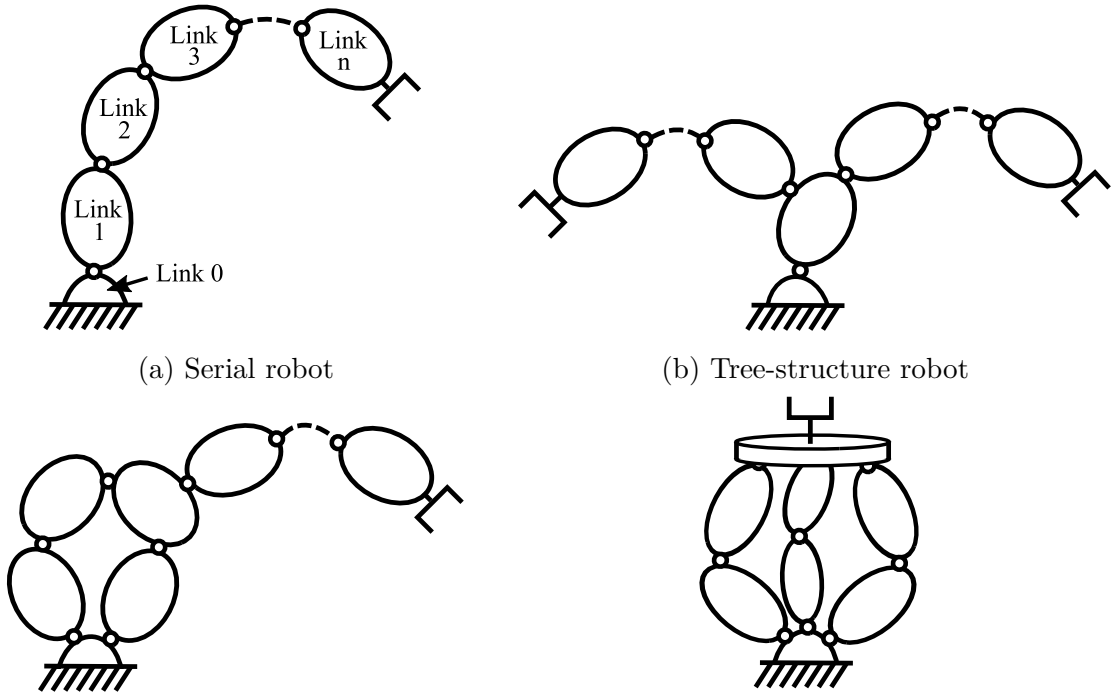
This manuscript presents the work done during the Ph.D. thesis of Angelica Ginnante in collaboration between *Ecole Centrale de Nantes*, *Università degli Studi di Genova* and the company *Nimbl'Bot*, started in November 2020. The work was supervised by Professors Stéphane Caro and Enrico Simetti, and Doctor François Leborne.

1.1 Robotic manipulators

The use of robotic manipulators in industry has grown in the last decades to improve and speed up industrial processes. Based on the desired application, different robotic solutions were developed. In [ISA15], the authors listed all the applications where manipulators are employed to improve the performance and resulting quality. Machining application is one of the main fields under study since it is a crucial task in the manufacturing industry to transform raw materials into functional parts [Che08]. Several researches were performed on robotic manipulators for machining tasks, pointing out the advantages and drawbacks [PDSC11, ISA15, KNH$^+$19]. Machining applications are mainly performed by computer numerical control (CNC) machine tools because of their high accuracy. Nevertheless, they are generally expensive and do not provide a high versatility [JW19]. Therefore, industrial manipulators started to be investigated. They can cover larger workspaces, increasing the range of achievable operations. Moreover, they reduce the scrap rates and production costs compared to CNC machines [CDGF13]. Industrial manipulators have already shown satisfactory performance in some tasks, like grinding [LUE90], polishing [TGA93] and deburring [NPRRA02]. Their main drawback is their overall lack of stiffness compared to CNC machines, leading to increased manufacturing

errors [DCGF12, CDGF13].

In [DK04], a robot manipulator is defined as composed of two main parts: (i) the end-effector to manipulate or transform objects and (ii) the articulated mechanical structure that moves the end-effector. One robot can have multiple end-effectors simultaneously in the same design. The articulated mechanical architecture aims to bring the end-effector to a desired pose or follow a specific trajectory. The robotic structure is composed of a rigid link series and is articulated by inserting joints between the links, generating a chain. The joints inserted in the chain can be of two types: revolute and prismatic. The first provides a rotational movement and the second a translational motion. The active joints actuate the robot and the passive joints that can not be directly actuated are moved by the active joints. The manipulators can be divided into four different sub-groups. The first is a simple open chain, called serial robots and shown in Fig.1.1a. There are three main classes for serial manipulators: (i) articulated robot [DP14], (ii) SCARA robot [DP14, KSP17] and (iii) Cartesian robot [DP14]. Each manipulator has its own scope and needs a different way of modeling. The authors of [DK04] present a complete guide to geometrically, kinematically and dynamically design a manipulator. The other two manipulator categories are the tree structure, shown in Fig.1.1b, and the closed



(a) Serial robot

(b) Tree-structure robot

(c) Closed-chain robot

(d) Parallel robot

Figure 1.1: Types of robotic manipulators [DK04]

chain, shown in Fig.1.1c. The presence of a closed chain always provides higher stability and strength to the robot, improving its stiffness. A specific type of closed-chain robot is the parallel one, shown in Fig.1.1d. The authors of [DK04] define a parallel robot as a mobile platform connected to a fixed base by a set of identical legs and an end-effector directly fixed to the mobile platform. However, a parallel robot can be formed of non-identical legs and include passive joints. A more detailed explanation for parallel robots is presented in [Mer06]. As described in [DK04], the geometric and kinematic modeling techniques developed for serial manipulators are inaccurate when used on parallel robots. So, specific modeling methods need to be employed. Moreover, the inverse geometric and kinematic problems are easy to solve in the case of parallel designs, but the direct geometric and kinematic problems are much more complex. The most common manipulators employed in industrial applications are serial and parallel.

In [DP14], the authors present a comparison of these two categories, listing advantages and drawbacks. Table 1.1 collects some of their main characteristics comparing these two robot types [DP14]. In general, serial and parallel manipulators have opposite advantages and disadvantages. In fact, serial robots are characterized by a larger workspace and higher flexibility, but their stiffness is commonly lower. On the contrary, parallel robots are usually stiffer and can hold higher payloads, but

Feature	**Serial**	**Parallel**
Workspace with respect to footprint	Large	Small
Stiffness	Low	High
Solving direct kinematic model	Easy	Complex
Solving inverse kinematic model	Complex	Easy
Modeling/solving dynamics	Simple	Complex
Payload/weight ratio	Low	High
Calibration	Simple	Complicated
Speed and acceleration	Low	High
Force error	Average	Accumulate
Position error	Accumulate	Average

Table 1.1: Serial and parallel robots comparison

their workspace is minimal compared to a serial design. So, one type is chosen over the other if the final application requires a wider workspace or a more robust design. This choice comports a trade-off favoring one category.

1.2 Hybrid manipulators

As described in Section 1.1, using a serial or parallel robot means choosing between a larger workspace or a stiffer architecture. When both are needed, hybrid manipulators can bring together the best of both worlds. The hybrid robots can be defined as a serial chain of non-serial mechanisms [TGK99, Tan00], a combination of closed-chain and open-chain architectures. Figure 1.2 shows a hybrid robot formed of a sequence of parallel robots serially connected. Hybrid robots can have very different designs. In [CBH08], the authors proposed a method for synthesizing new hybrid robots based on the desired application. The main drawback of hybrid robots is the complexity of their kinematics model [Tan00]. In fact, these robots can be actuated by complex closed loop or parallel mechanisms serially attached. So, the classical techniques used to analyze simpler designs, such as revolute and prismatic joints, cannot be used. In [PSVP13], a hybrid robot for surgical applications is proposed and its kinematic model is computed and analyzed. The work demonstrates

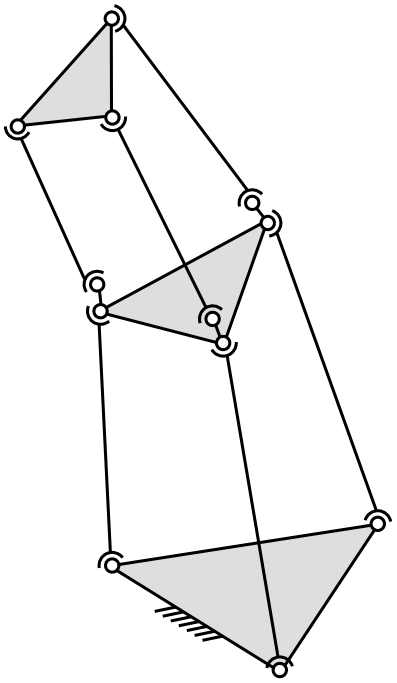


Figure 1.2: Drawing of a hybrid robot architecture [Tan00]

the complexity of defining hybrid robot kinematics.

In [KWdGF+20], the authors present a survey that examines the development of series-parallel hybrid robots across different application domains: humanoids, multi-legged robotic systems and industrial manipulators. Humanoids are bipedal robots that mimic human anatomy with complex mechatronic systems. High dynamic performance in humanoids requires a stiff architecture and good mass distribution, which can be achieved using parallel mechanisms to design serial legs. Some examples of humanoid robots with parallel mechanisms include Lola [LBUP06], NASA Valkyrie [RSH+15], TORO [EWO+14], LARMbot [CWCC16], CARL [SNM+17] and Disney Research bipedal robot [GKY18]. These robots utilize various types of parallel mechanisms in joints, such as spatial slider crank, rotational parallel mechanisms or parallel kinematic mechanism (PKM) modules. Then, there are the multi-legged robots designed for high-payload applications that incorporate closed-loop linkages and parallel mechanisms. The design of these robots incorporates different types of parallel mechanisms, like Stewart platforms, PKMs or parallelogram linkages, to enhance joint strength and stability. Several designs are proposed in the survey, for example HeritageBot [CCRC18], Menzi Muck M545 [JLKH19], MIT Cheetah [WWS+17] and the quadrupedal platform Stoch[DBG+19]. Finally, in industrial automation, series-parallel hybrid robots are used to improve the stiffness and enhance the workspace size. There exist different designs developed for

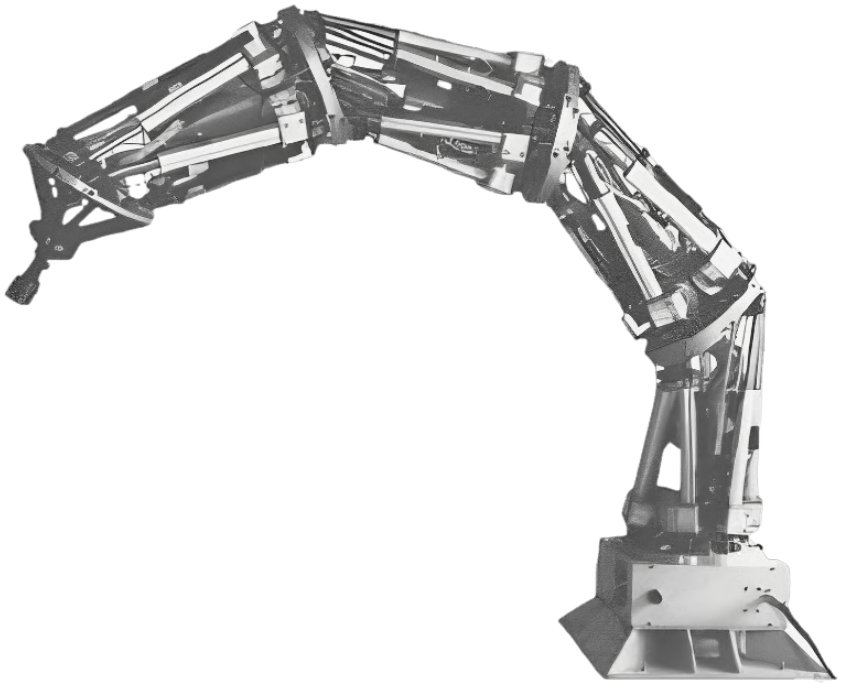


Figure 1.3: Representation of the Logabex LX4 [MD95]

real applications. One of the first examples is the Logabex LX4 robot [MD95], a serial concatenation of Stewart platforms. Figure 1.3 shows the Logabex LX4 robot. Other industrial manipulators from ABB, KUKA, Comau and the FANUC M-3iA/6A Delta robot employ parallelogram mechanisms to increase the stiffness for pick-and-place operations.

1.3 Kinematic redundant manipulators

"Redundant" means "exceeding what is necessary or normal". A robot is kinematically redundant with respect to a task when it has more degrees of freedom than the required amount necessary to perform that task [Sic90]. In fact, no manipulator is inherently redundant, but there are specific tasks with respect to which the robot becomes redundant [SKK08]. Mathematically, the kinematic redundancy appears when the dimension of its actuation vector $\mathbf{q} \in \mathbb{R}^n$ is greater than the dimension of the task vector $\mathbf{x} \in \mathbb{R}^m$, namely when $n > m$ [CB94]. So, the desired task can be achieved by multiple possible robot configurations. Both serial and parallel robots can be kinematically redundant. In the case of serial manipulators, redundancy is introduced by adding actuated joints into the serial chain. Contrarily, introducing redundancy in a parallel mechanism is less straightforward since there are several ways to do that, as described in [GS18]. This research concentrates on the kinematic redundancy of serial designs and does not address the topic of parallel mechanism redundancy.

In the manufacturing industry, the kinematic redundancy of robotic manipulators can be viewed as a possible way to improve the robotic machine abilities and performance [GST19]. One of the main motivations to introduce kinematic redundancy in a robotic manipulator is to increase the robustness to possible faults, improving the reliability [COW08]. Moreover, kinematic redundancy is also employed to increase the robot dexterity and enables new robot behaviors, like self-motion, i.e., a set of joint velocities causing no Cartesian motion to the end-effector [COW08]. The kinematic redundancy can also be used to work in cluttered environments [MRG17], such as medical robotics, and solve several tasks simultaneously while optimizing some performance criteria [SW95]. Restricting a manipulator to the minimum necessary number of degrees of freedom for accomplishing a task can lead to significant drawbacks in practical applications [COW08]. The limitations appear not only in the case of singularity issues but also in the presence of constraints like joint limits or obstacles within the workspace. A variety of seven degrees of freedom robots is

Figure 1.4: Representation of the KUKA LBR iiwa robot

largely used for different industrial applications to enhance dexterity. Some famous robotic examples are the KUKA LBR iiwa, shown in Fig. 1.4, and the ABB Yumi, shown in Fig. 1.5.

Robots are considered hyper-redundant when their number of degrees of freedom is much greater than the dimension of the task [HN91, Tan00], namely when $n \gg m$. Adding more degrees of freedom to already redundant manipulators allows solving many more simultaneous tasks [COW08]. Hyper-redundant robots can be divided into two categories: (i) rigid-link and (ii) continuum designs. Rigid-link hyper-redundant manipulators are the most straightforward evolution for redundant robots [COW08]. They are obtained by adding more links-joints to the already redundant manipulators. Generally, the link dimensions are reduced to make the robot design resemble a biological spine. This approach allows the creation of compact robotic manipulators with a high level of redundancy. The key concept behind rigid-link hyper-redundant robots is that they maintain all the geometric and kinematic conventions of classical manipulators, such as Denavit–Hartenberg based approaches and Jacobian computation methods, simplifying their use [COW08]. One example

Figure 1.5: Representation of the single-arm ABB Yumi robot

of rigid-link hyper-redundant manipulators is the 30 degrees of freedom planar manipulator developed at Caltech [CB93]. Rigid-link hyper-redundant manipulators can also be employed in dual-arm designs like the one presented in [KTV+90] and a NASA special-purpose dexterous manipulator [HW03]. Recently, hyper-redundant manipulators have been increasingly used for inspection applications. This robots are sometimes called snake robots. As biological snakes, these robots can enter in small, irregular and challenging environments [Pet17] where it would be dangerous for human operators to go. Some examples of rigid-link robotic snakes can be found in the reviews [LPSG13, SAM+17, Pet17]. Figure 1.6 shows an example of snake robots. It comprises ten identical joint modules with passive wheels created for locomotion across flat surfaces.

The second category of hyper-redundant robots is called continuum. This type of manipulator carries the concept of kinematic redundancy to the extreme where the number of joints tends to infinity and the link lengths tend to zero [COW08]. Thanks to the number of joints that tend to infinity, the size of continuum robots can be highly reduced, making them optimal in the field of surgery [BKRC15]. In this case, different and more complex ways of modeling are required. The continuum

Figure 1.6: Representation of the snake robot Wheeko

robot category is not treated in the manuscript.

1.3.1 Kinematic redundancy resolution

The primary challenge posed by redundant manipulators lies in resolving their kinematic redundancy, as multiple viable solutions exist for a given task. In hyper-redundant robots, this challenge becomes even more pronounced, with the number of potential solutions approaching infinity. The simplest way to deal with this problem is using the pseudo-inverse Jacobian matrix method [DK04]. However, this approach neither avoids singularities nor takes advantage of the robot kinematic redundancy to optimize the robot configuration. Later, different types of algorithms and optimization techniques were developed to address the redundancy problem better. As described in [SKK08], the kinematic redundancy resolution methods divide into two main groups, via optimization and via task augmentation. In the first case, the degrees of freedom excess can be used to improve the value of performance criteria while executing the main task. The improved metric can depend on both the robot configuration and the velocities and forces applied to its end-effector. One example of optimization for kinematic redundancy resolution is the minimum effort solution [GW00]. This technique exploits the least infinity norm optimization to solve the inverse kinematic problem of redundant robots. In this research, the infinity norm optimization provides better results than the pseudo-inverse since the infinity norm minimizes the maximum component magnitude, meeting all the physical limit constraints. In [CLV06], the authors presented a kinematic redundancy resolution algorithm for a serial-parallel manipulator based on local kinematic opti-

mization. The proposed algorithm effectively solved the inverse kinematic problem of a serial-parallel redundant manipulator, meeting the joint requirements for both active and passive joints. Moreover, the algorithm ensured smooth profiles for the active joints while performing the desired application. In [RMG16], the authors treated a problem of trajectory tracking for a general kinematic redundant robot as two interdependent problems, inverse kinematics and trajectory optimization, to identify the time-optimal path tracking solution. In this research, an enhancement to the differential inverse kinematics resolution method involves the addition of an optimal linear combination of null-space basis vectors from the corresponding Jacobian velocity vector. Many other researches were developed about the kinematic redundancy resolution via optimization in the literature.

The second way to solve the inverse kinematic problem exploiting the redundancy of a manipulator is via task augmentation [SKK08]. For example, considering a robot with seven degrees of freedom and a tracking trajectory task that requires only five degrees of freedom, two degrees of freedom remain available. The task vector can be augmented from five to six or seven degrees of freedom by adding other objectives. This resolution methodology is particularly effective in the case of hyper-redundant robots. In fact, these robots have many available degrees of freedom that are not employed by the main task and many additional tasks can be considered. The task augmentation technique is employed in this manuscript to kinematically control the employed manipulators, exploiting their kinematic redundancy. In fact, adding new objectives to the main task allows respecting different constraints, optimizing metrics or performing simultaneous applications. The literature review about task augmentation methods is proposed later in Section 3.1.

1.4 Nimbl'Bot robot overview

This section presents an overview of the first robot prototype developed by the company Nimbl'Bot and its purpose. A complex topic when talking about machining applications is the high precision milling, grooving and trimming of small metallic components. The elements to produce can have complex shapes and their required accuracy can be difficult to reach. CNC machines or existing serial machining robots can be too big and not have the required flexibility to properly shape these components. So, Nimbl'Bot started investigating new robotic solutions to solve this problem. The key concept behind the performed studies was identifying a new robotic design that ensures stiffness, positioning accuracy, compactness, modularity

and flexibility. As pointed out in the previous sections, the typical actuation joints, revolute and prismatic, can not be used to build stiff, flexible and compact serial manipulators. So, Nimbl'Bot developed and patented a new actuation mechanism whose goal is to ensure stiffness and avoid backlash. Moreover, its compact design avoids the generation of bulky manipulators. This actuation mechanism is composed of two kinematic chains that together form a closed design. These two chains help distributing the applied forces on the entire design improving the stiffness and positioning accuracy. This actuation mechanism is called NB-module in this manuscript and its geometric and kinematic models are fully analyzed in Chapter 2. The NB-module is actuated by two motors that generate two rotational actuation. So, one mechanism provides two degrees of freedom.

The first prototype of the Nimbl'Bot robot is composed of a serial arrangement of ten NB-modules. The NB-module design requirement to ensure stiffness is to have a maximum solid reachable angle of $\pm\pi/6$ rad. So, many NB-modules need to be arranged together to ensure a sufficiently large orientation workspace. The prototype is shown in Fig. 1.7 and named NB-R1 in this manuscript. It can be divided into three regions, i.e., the shoulder, formed of three NB-modules, the elbow,

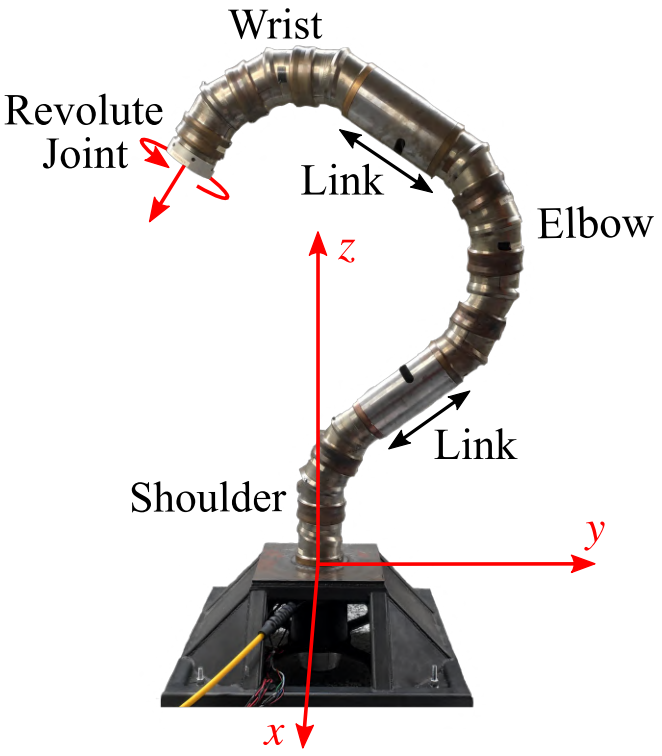


Figure 1.7: NB-R1 robot actuated by ten NB-modules mounted in series and a final revolute joint. Shoulder and wrist made of three NB-modules, covered solid angle of $\pm\pi/2$ rad each. Elbow made of four NB-modules, solid angle of $\pm 2\pi/3$ rad.

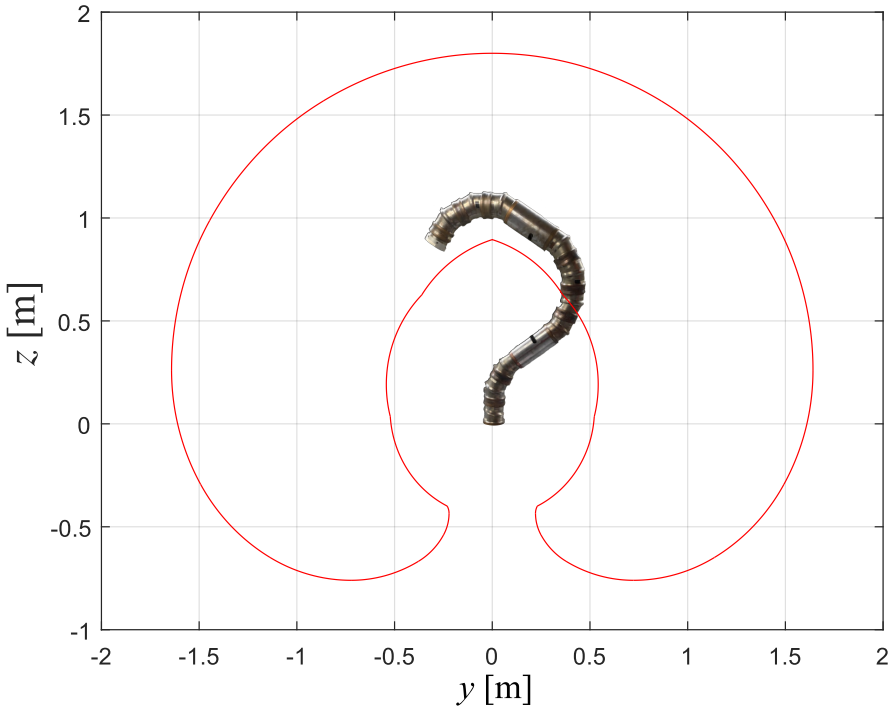
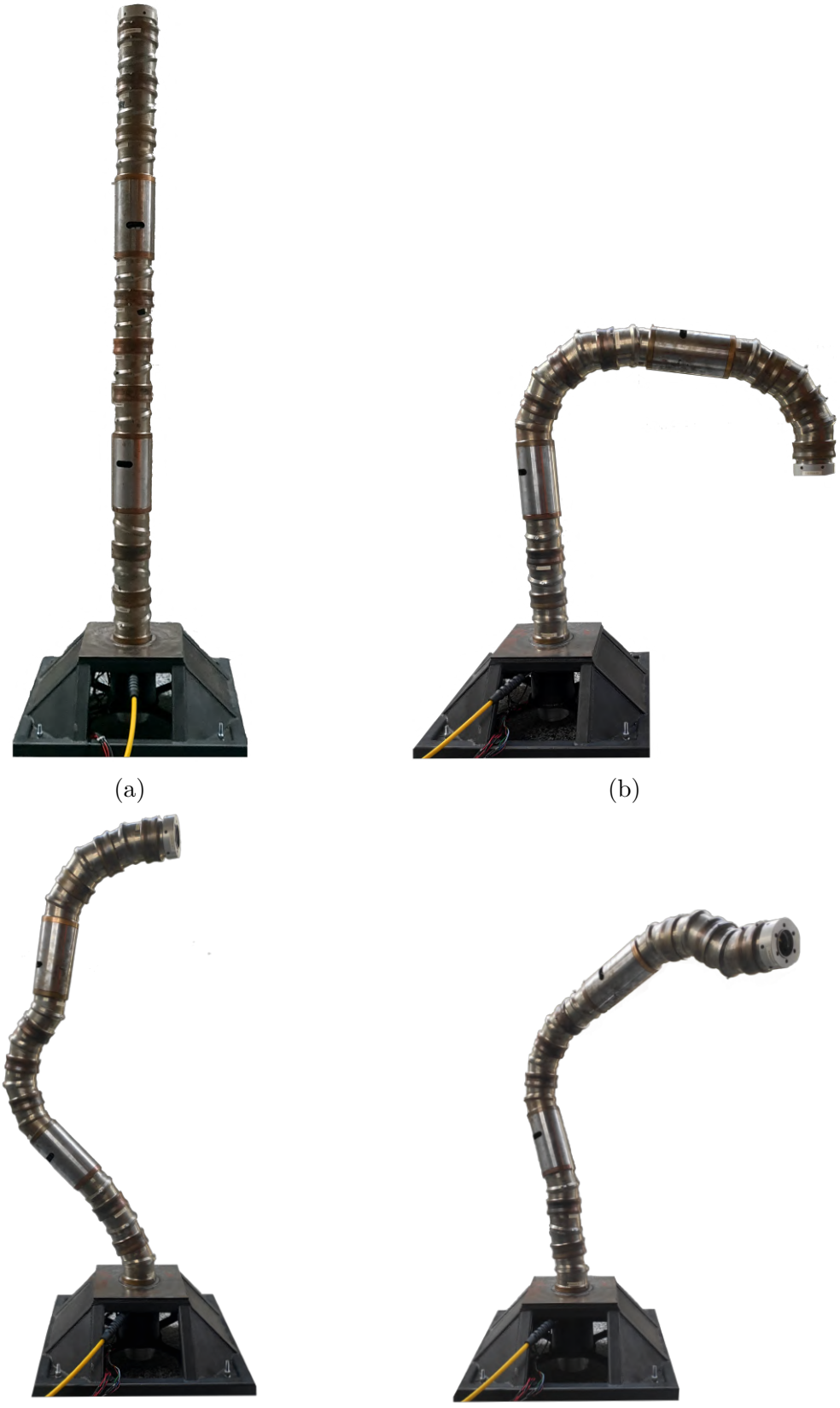


Figure 1.8: Vertical section of the NB-R1 workspace boundaries

four NB-modules, and the wrist, three NB-modules. Two links connect these three regions to increase even more the workspace size, allowing the end-effector to reach further poses. A vertical section of the NB-R1 workspace is shown in Fig. 1.8. This workspace is obtained through a new proposed method explained later in Chapter 5. The complete workspace is obtained by rotating this section around the axis z.

Since each NB-module provides two degrees of freedom plus a final revolute joint added at the end to allow adjusting the tool orientation, the NB-R1 has 21 degrees of freedom. This makes it a kinematic hyper-redundant robot considering that machining operations usually requires five degrees of freedom. So, the NB-R1 could possibly perform the same machining task with almost infinite possible configuration. This high redundancy provides versatility and the ability of working in cluttered environments. Moreover, this robot can be defined as hybrid, mixing serial and closed chains, and modular, an attachment of modules. The serial and closed chains mix gives strength to final design and the modularity improve the robot robustness and fault tolerance. A deeper discussion about modular robots is provided in Section 2.1. Finally, a cable passes inside the NB-R1 structure, constraining the design and canceling the remaining mechanical backlash. The two links are hollow to allow the routing of internal cables and reduce the final weight. Figure 1.9 shows four postures of the NB-R1 prototype.



(a) (b)

(c) (d)

Figure 1.9: Four postures of the NB-R1 prototype

1.5 Thesis contribution and outline

The research proposed in this manuscript revolves around some open issues related to kinematically redundant spatial robots. As mention in the previous sections, these type of robots are increasingly explored because of the opportunities they can provide in the manufacturing industry. However, there are still several problems that needs to be addressed. Here, three main issues are analyzed and studied to propose some possible solutions. The first issue concerns the kinematic redundancy resolution problem. To address this topic, a task priority kinematic resolution algorithm is used for the kinematic control of redundant manipulators. This type of algorithm exploits the kinematic redundancy of the robot to solve multiple simultaneous tasks. The second issue is related to the design optimization of kinematic redundant robots as a function of their main application. A new design optimization method based on the task priority kinematic resolution algorithm is proposed. This new process gives as output some guidelines to build performant robots with respect to the desired application and working area. The third issue is related to the workspace determination of kinematic redundant robots, which is a complex and important topic. A new workspace determination process again base on the task priority kinematic resolution algorithm is proposed. This new method can be defined as ray-based and accurately detect the workspace boundaries of highly redundant designs in a small period of time. The proposed solutions to the three addressed issues are all tested on some Nimbl'Bot robot designs.

Here, the chapters organization is presented. Chapter 2 describes the mechanism developed by Nimbl'Bot to build kinematic redundant robots, called NB-module. The geometric and kinematic models of this mechanism are presented and explained. Then, the NB-module design parameters are investigated as a function of its geometric and kinematic performance. Part of the work presented in this chapter was published in [GLC+21, GCSL23a]. Chapter 3 presents the task priority kinematic resolution algorithm and the tasks developed to perform a kinetostatic optimization of the robot configuration while performing other tasks. The algorithm is tested making the NB-R1 robot following some trajectories while improving its kinetostatic performance. Part of the work presented in this chapter was published in [GCSL23a]. Chapter 4 describes the new design optimization process for kinematic redundant manipulators. This method is tested to optimize the NB-R1 robot design with respect to a set of trajectories. The optimization is performed to obtain high kinetostatic performance while following the selected trajectories. In the end,

some guidelines to build kinetostatic performant designs are obtained. Part of the work presented in this chapter was published in [GSCL23]. Chapter 5 presents a workspace determination algorithm developed for kinematic redundant robots. This process is tested on three different designs of the Nimbl'Bot robot to demonstrate its versatility. Then, it is compared with other two methods demonstrating its pre-eminence. Part of the work presented in this chapter was published in [GCSL23b]. Chapter 6 presents the conclusion and future works. Figure 1.10 shows a flowchart with the manuscript organization.

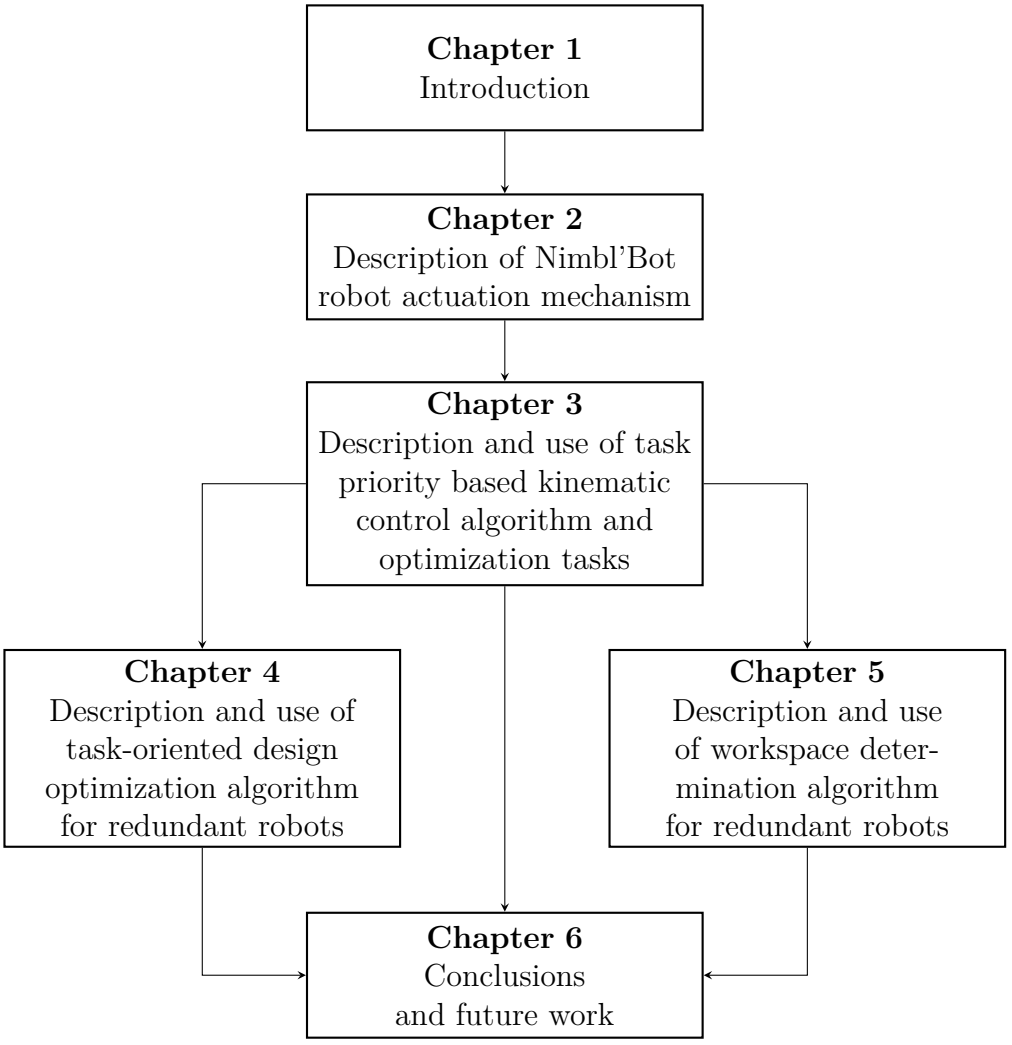


Figure 1.10: Flowchart of the thesis manuscript outline

Chapter 2

Geometric and Kinematic Analysis of Nimbl'Bot NB-Module

This chapter describes the two degrees of freedom mechanism patented by the company Nimbl'Bot [Duf21] and called NB-module. Here, the models are recalled and further analysis are performed. The chapter is organized as follows. Section 2.1 introduces the possible ways to build modular redundant manipulators and the existing actuation mechanisms that can be used. Section 2.2 describes the NB-module design and its actuation. Section 2.3 presents the geometric model computation for the NB-module and shows its workspace and joint space for specific design parameter values. Section 2.4 computes the NB-module kinematic model and analyzes its kinematic performance based on the design parameter values. The conclusion are presented in Section 2.5.

2.1 State of the art on mechanism and modular robots

As introduced in Section 1.3, kinematic redundant robots can be designed in different ways and provide several advantages. Kinematic redundant robots are generally classified into three categories: discrete, continuous and modular robots [CB92, CB95]. Modular robots were firstly introduced in [FN88, FK90]. They can be defined as an assembly of several actuation mechanisms, or modules [Bra16]. Each module is functionally and structurally independent [Bra16] and disposes of few degrees of freedom [AM15]. Usually, modular robots are composed of many modules and have a high number of degrees of freedom, leading to kinematic redundancy [Bra16]. Modular robots were developed as a solution to the low flexibility and adaptabil-

ity of fixed-body monolithic conventional robots [AM15]. This type of robots has three main advantages: versatility, robustness and lower costs [APS19, BRS+17]. The modular system versatility refers to the ability of transforming into numerous shapes [APS19]. This feature allows completing various tasks in different and cluttered environments. The versatility can be evaluated by the number of isomorphic configurations [DDN19] that the robot can reach and the number of degrees of freedom it possesses. This characteristic increases together with the number of modules that compose the robot. Moreover, the robot design can be quickly adapted according to a given task by changing the number of modules [Bra16]. Redundancy and self-repair due to the use of many identical modules provide robustness to the system, enabling any broken module to be replaced easily [APS19]. Thanks to the large amount of degrees of freedom, if one module stops working properly it can be disabled and the others can still complete the desired task. 3D printing prototyping and later batch production are cost-effective methods for developing repeated modules while maintaining low costs [APS19]. One drawback of modular robots is the raising complexity of computing the geometric and kinematic models [Bra16]. In fact, the modules used to built the modular manipulator can have complex non-conventional designs. So, identifying their geometric and kinematic models can become more difficult.

A first approach to the kinematic modeling for modular robots was presented in [BZL89]. The research describes a methodology to derive the individual kinematic models of all the modular units and a global kinematic model for any robot configured using these modular units. In [CY96], the authors presented a newly developed modular robot aimed for assembly task. The authors employed the so called dyad kinematics along with a graph traversing algorithm to derive the forward kinematics. Other types of modular robots developed for industrial application, like assembly task, were proposed later in [ABR08, SY11, LXGC17]. Thanks to the system versatility, modular robots can be employed in cluttered environments. A mechanism for modular redundant snake robots was introduced in [WBC+03]. The authors demonstrated how the obtained robot was able to inspect unreachable areas dangerous for users. In [WJP+07, JWT+11, WBB+12], different types of modular snake robots are described demonstrating their abilities in inspection applications. In [SWBC03], the authors presented a two degrees of freedom mechanism for modular redundant robots. The investigated concept uses a complex design optimized for compactness, strength and range of motion. Then, a similar improved design was proposed in [SWC06]. In [RSBT18], a flexible universal spatial robotic tail actuated by a cable-driven segment is introduced. Another possible application for

modular redundant robot is the surgical environment. In [OHAH+20], the authors proposed an actuation mechanism for surgical snake robots and tested its abilities. However, none of these mechanisms were ever used to build machining robots. The mechanism called NB-module and proposed by Nimbl'Bot is specifically developed to build compact machining robots.

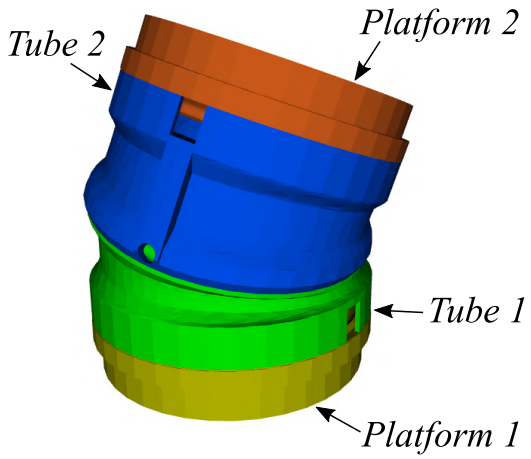
2.2 Nimbl'Bot mechanism description

The NB-module is the actuation mechanism developed by the company Nimbl'Bot to build compact stiffer modular robots. This innovative mechanism is designed to donate strength and stiffness to manipulators, improving the end-effector pose precision and reducing the backlash. This new design consists of a closed kinematic chain mechanism composed of two chains, one internal and the other external. As Chapter 1 points out, the closed chain architectures provide higher stiffness and stability to the entire manipulator. Moreover, the NB-module design is compact, avoiding the development of bulky designs. It is actuated by two motors, providing two degrees of freedom. This section describes the NB-module external and internal design. Then, its actuation is presented.

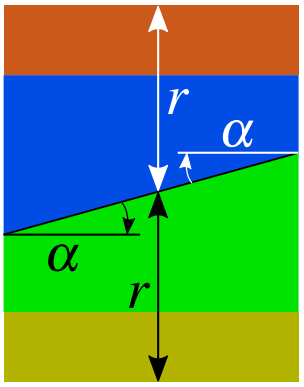
2.2.1 Description of the NB-module external kinematic chain

The external kinematic chain has seven different components. Four of them are shown in Fig. 2.1a. The fixed base, in yellow, is named *Platform 1* and is considered centered on the origin frame for which the NB-module transformation matrix is calculated. Above *Platform 1*, there is a rotating cylinder, in green, named *Tube 1*. *Tube 1* is a hollow cylinder cut by an oblique plane with height r and slope α. The design parameters r and α are shown in Fig. 2.1b. The first motor actuates the component *Tube 1*. The motor is attached directly to the inner side of *Tube 1*. In this way, *Tube 1* can rotate around the vertical axis that passes through the center of *Platform 1*. *Tube 2*, in blue, is placed over *Tube 1*. In this case, they have the same shape, but, in principle, their height r and slope α could be different. The second motor actuates *Tube 2* and is attached to its inner side. *Tube 2* can rotate around the axis perpendicular to and centered in *Platform 2*. The external kinematic chain is closed by *Platform 2*, the moving platform in orange, which is the end-effector of the NB-module.

Cutting obliquely the cylindrical tubes results in an elliptical shape. However,



(a) 3D view with component names



(b) Front view scheme with design parameters

Figure 2.1: External view of the NB-module

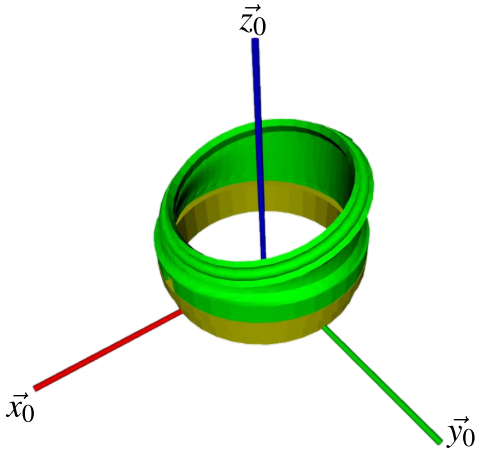


Figure 2.2: View of *Tube 1* oblique side

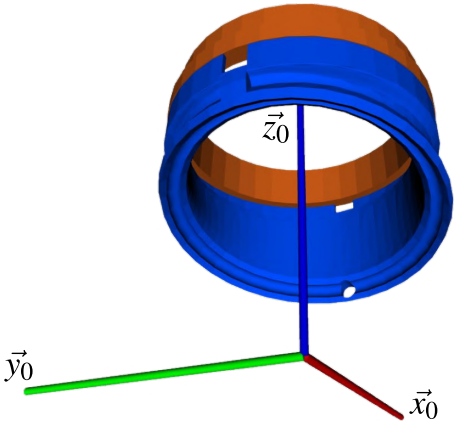
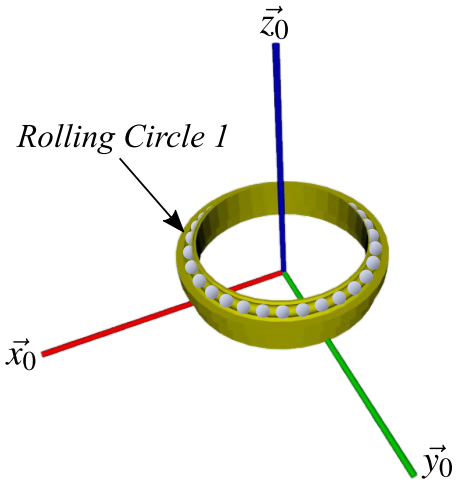
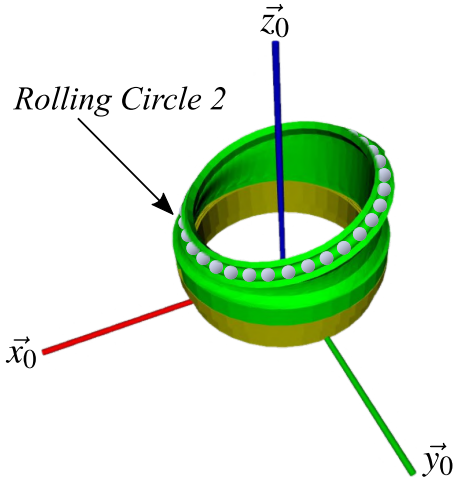


Figure 2.3: View of *Tube 2* oblique side

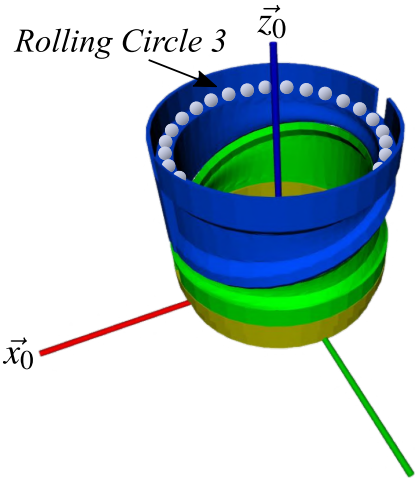
the tube oblique sides are reshaped in a circular way to allow a continuous rotation between the oblique planes. So, as it can be seen in Figs. 2.2 and 2.3, a circular groove is designed above the inclined sides of the two tubes. Three rolling circles formed of a series of small balls are inserted between the platforms and the tubes to allow a fluid movement. The balls are inserted in the grooves machined in the platforms and tubes, as shown in Fig. 2.4. These rolling circles are called *Rolling Circle 1*, *Rolling Circle 2* and *Rolling Circle 3* and represent the three last elements of the external kinematic chain. Consequently, *Tube 1* and *Tube 2* can independently rotate with a continuous movement.



(a) Location of *Rolling Circle 1* over *Platform 1*



(b) Location of *Rolling Circle 2* over *Tube 1*



(c) Location of *Rolling Circle 3* over *Tube 2*

Figure 2.4: Location of rolling circles formed of a series of balls in NB-module

2.2.2 Description of the NB-module internal kinematic chain

The internal kinematic chain has four components, as shown in Fig. 2.5. Two of those also belong to the external chain, i.e. *Platform 1* and *Platform 2*, generating the closed kinematic chain mechanism. *Platform 1* is linked to the component *Ball Nut* in purple through a prismatic joint, which prevents internal breaks while the NB-module is actuated. These could occur due to dimensional inaccuracies in the mechanical parts. Following that, there is the element *Ball Joint Axis* in cyan that forms a constant velocity joint with the element *Ball Nut*. Finally, *Ball Joint Axis* is linked to *Platform 2* through another prismatic joint, again to avoid internal breaks. The variable r equal to the tube heights represents also the distance between *Platform 1* and the constant velocity joint and between the constant velocity joint

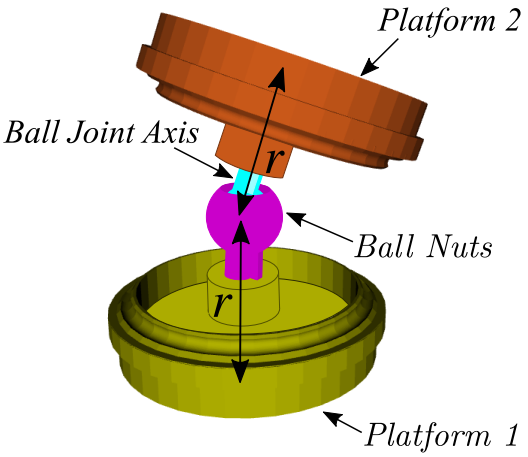


Figure 2.5: Internal view of the NB-module with component names

and *Platform 2*.

One NB-module interesting feature is the presence of a constant velocity joint, which works like a universal joint, but allows its two ends to rotate at the same velocity [Car09]. Therefore, fixing *Platform 1* means forcing no rotation to the whole internal kinematic chain. So, thanks to the constant velocity joint and the rolling circles that decouple the rotation of each component, the tube rotations lead to an inclination of *Platform 2* with no rotation about its normal axis. The NB-module amounts to a zero-torsion mechanism.

2.2.3 NB-module actuation

The lower half NB-module, yellow *Platform 1* and green *Tube 1*, is actuated in the same way as the upper half NB-module, blue *Tube 2* and orange *Platform 2*. The lower half NB-module actuation generates a rotation angle between *Platform 1* Plane and *Tube 1* Plane, called q_1. The upper half NB-module actuation works equally, generating a rotation angle called q_2 between *Platform 2* Plane and *Tube 2* Plane. Figures 2.6 and 2.7 show the actuation joint variables q_1 and q_2 with respect to the component planes. A peculiarity of the NB-module is that both the motors have endless courses and can infinitely rotate, never reaching a limit. Figure 2.8 shows the zero position of the NB-module. In this configuration, q_1 and q_2 are both equal to 0. The shortest side of *Tube 1* is along the positive side of axis $\vec{x_0}$ and the shortest side of *Tube 2* is along the negative side of axis $\vec{x_0}$.

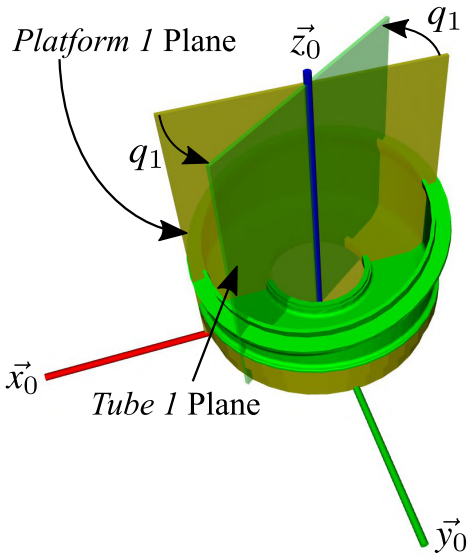


Figure 2.6: Actuation variable q_1 of the NB-module

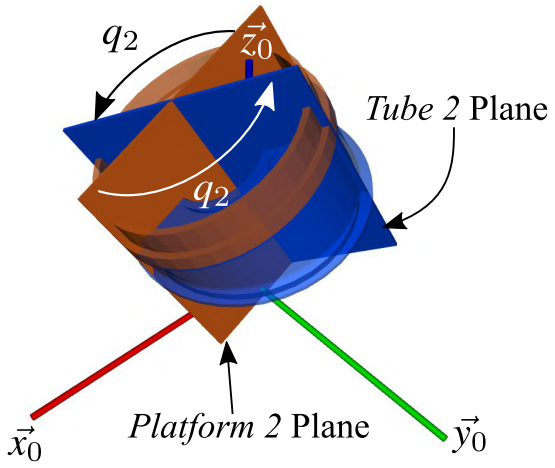


Figure 2.7: Actuation variable q_2 of the NB-module

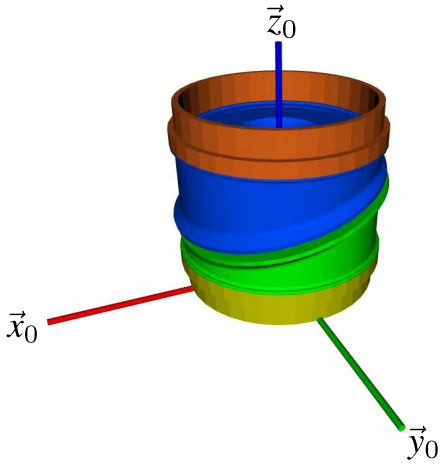


Figure 2.8: Home pose of the NB-module

2.3 Geometric model of NB-module

This section presents the geometric model of the NB-module. Firstly, the zero-torsion characteristic of the NB-module is explained. Then, the complete NB-module geometric model is presented with its transformation matrices. Finally, its workspace is shown and analyzed, setting the design parameters to specific values.

2.3.1 Zero-torsion mechanism

The ending platform of a zero-torsion mechanism never rotates about its normal axis [NCW18], but it can tilt around the axis parallel to the platform. The zero-torsion rotation matrix can be derived from the Tilt & Torsion (T&T) angles notation defined in [BZG02]. Figure 2.9 depicts the T&T angles convention. It involves only two rotation angles, the tilt angle θ rotating around the axis a and the torsion angle σ rotating around the axis $\vec{z^*}$. The azimuth angle ϕ, rotating around the axis $\vec{z}$, defines the vertical plane orientation perpendicular to the axis a. The tilt θ, torsion σ and azimuth ϕ angles are shown in Fig. 2.9. These angles take values as follows: $\theta \in [0, \pi)$ rad, $\sigma \in (-\pi, \pi]$ rad and $\phi \in (-\pi, \pi]$ rad. The T&T convention rotation matrix is:

$$\mathbf{R}(\phi,\theta,\sigma) = \begin{bmatrix} \cos\phi\cos\theta\cos(\sigma-\phi) - \sin\phi\sin(\sigma-\phi) & -\cos\phi\cos\theta\sin(\sigma-\phi) - \sin\phi\cos(\sigma-\phi) & \cos\phi\sin\theta \\ \sin\phi\cos\theta\cos(\sigma-\phi) + \cos\phi\sin(\sigma-\phi) & -\sin\phi\cos\theta\sin(\sigma-\phi) + \cos\phi\cos(\sigma-\phi) & \sin\phi\sin\theta \\ -\sin\theta\cos(\sigma-\phi) & \sin\theta\sin(\sigma-\phi) & \cos\theta \end{bmatrix}. \quad (2.1)$$

The value of σ can be set to zero to obtain the rotation matrix of a zero-torsion mechanism,

$$\mathbf{R}(\phi,\theta) = \begin{bmatrix} \cos^2\phi\cos\theta + \sin^2\phi & \cos\phi\sin\phi(\cos\theta - 1) & \cos\phi\sin\theta \\ \sin\phi\cos\phi(\cos\theta - 1) & \sin^2\phi\cos\theta + \cos^2\phi & \sin\phi\sin\theta \\ -\sin\theta\cos\phi & -\sin\theta\sin\phi & \cos\theta \end{bmatrix}. \quad (2.2)$$

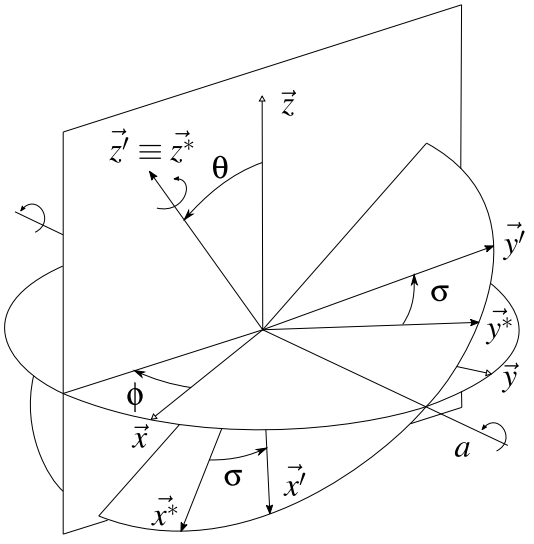
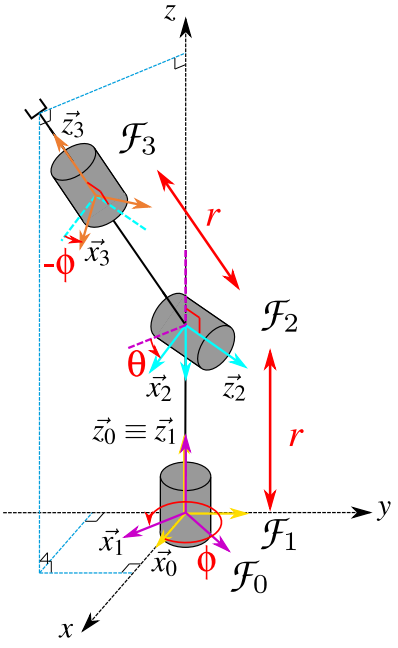


Figure 2.9: Tilt and torsion angle notation [BZG02]

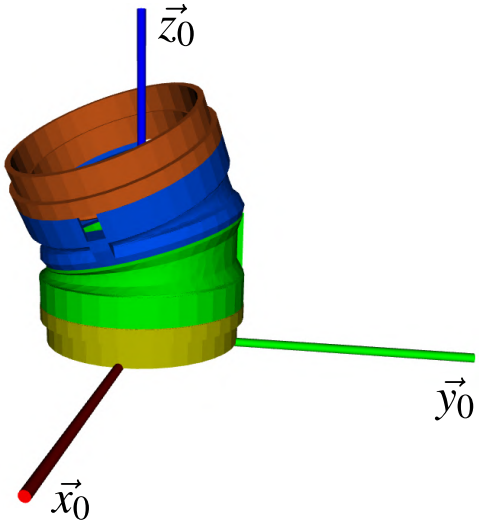
2.3.2 NB-module transformation matrix computation

Here, the geometric model of the NB-module is described. Since the NB-module is a zero-torsion mechanism, the rotation matrix of a frame rigidly attached to *Platform 2* with respect to a frame rigidly attached to *Platform 1* can be described using the notation presented in section 2.3.1. Here, the torsion angle σ is set to zero and the T&T notation becomes the Tilt & Azimuth (T&A) notation. Figure 2.10 shows the T&A based geometric model of the NB-module next to its CAD representation. A series of three revolute joints form the geometric model. The first revolute joint represents the azimuth angle ϕ of the NB-module. The second revolute joint is the tilt angle θ. The third revolute joint is constrained to have the negative value of the azimuth angle -ϕ as a consequence of the NB-module zero-torsion characteristics. Given the kinematic chain of Fig. 2.10a, the NB-module rotation matrix ${}^0\mathbf{R}_3(\phi,\theta)$ and translation vector ${}^0\mathbf{p}_3(\phi,\theta,r)$ pointing from the origin of frame $\mathcal{F}_0$ to the origin of frame $\mathcal{F}_3$ are

$$ {}^0\mathbf{R}_3(\phi,\theta) = \begin{bmatrix} \cos^2\phi\cos\theta + \sin^2\phi & \cos\phi\sin\phi(\cos\theta-1) & \cos\phi\sin\theta \\ \sin\phi\cos\phi(\cos\theta-1) & \sin^2\phi\cos\theta + \cos^2\phi & \sin\phi\sin\theta \\ -\sin\theta\cos\phi & -\sin\theta\sin\phi & \cos\theta \end{bmatrix}, \quad (2.3) $$



(a) Geometric model scheme



(b) CAD representation of geometric model posture

Figure 2.10: Tilt and azimuth geometric model of the NB-module with CAD representation

and

$$ {}^{0}\mathbf{p}_3(\phi,\theta,r) = \begin{bmatrix} r\sin\theta\cos\phi \\ r\sin\theta\sin\phi \\ r + r\cos\theta \end{bmatrix}. \tag{2.4} $$

The equation defining the translation vector ${}^{0}\mathbf{p}_3$ also represents the so-called spherical coordinates. The complete homogeneous transformation matrix of the NB-module from frame $\mathcal{F}_0$ to frame $\mathcal{F}_3$ is expressed as

$$ {}^{0}\mathbf{T}_3(\phi,\theta,r) = \left[\begin{array}{c|c} {}^{0}\mathbf{R}_3(\phi,\theta) & {}^{0}\mathbf{p}_3(\phi,\theta,r) \\ \hline \mathbf{0}_{1\times3} & 1 \end{array}\right]. \tag{2.5} $$

The azimuth ϕ and tilt θ angles express the NB-module transformation matrix in the T&A notation. However, these angles do not represent the tubes angular position, called q_1 and q_2. So, the azimuth ϕ and tilt θ angles need to be expressed as functions of the actuation variables q_1 and q_2,

$$ \begin{cases} \phi = \dfrac{q_1 + q_2 - \pi}{2} \\ \theta = \arctan\left(-\dfrac{2\tan\alpha\sin\left(\frac{q_1-q_2}{2}\right)}{1-\tan^2\alpha\sin^2\left(\frac{q_1-q_2}{2}\right)} \right) \end{cases}, \tag{2.6} $$

where *arctan* is the tangent inverse function. The sign given as input to *arctan* is very important since multiple angles can return the same tangent value So, it is necessary to consider the *arctan* input sign to accurately determine the correct angle. Similarly, q_1 and q_2 can be expressed as functions of ϕ and θ,

$$ \begin{cases} q_1 = \phi + \arccos\left(-\dfrac{\cos\alpha\,(\cos\theta - 1)}{\sin\alpha\sin\theta} \right) \\ q_2 = \phi - \arccos\left(-\dfrac{\cos\alpha\,(\cos\theta - 1)}{\sin\alpha\sin\theta} \right) + \pi \end{cases}, \tag{2.7} $$

where α is the slope of the oblique planes in *Tube 1* and *Tube 2*. When the tilt θ is equal to 0, the value of the actuation variables is $q_1 = q_2 = \phi + \pi/2$.

The azimuth ϕ and tilt θ angles simplify visualizing the NB-module orientation. In fact, the planes identified by these two angles help understanding the NB-module direction. Figure 2.11 shows the Azimuth Plane with the azimuth angle ϕ. So, ϕ gives the orientation along which *Platform 2* is tilted. Figure 2.12 shows the Tilt

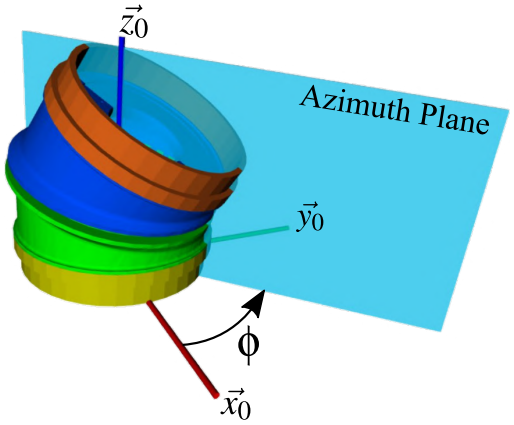


Figure 2.11: Azimuth plane on the NB-module

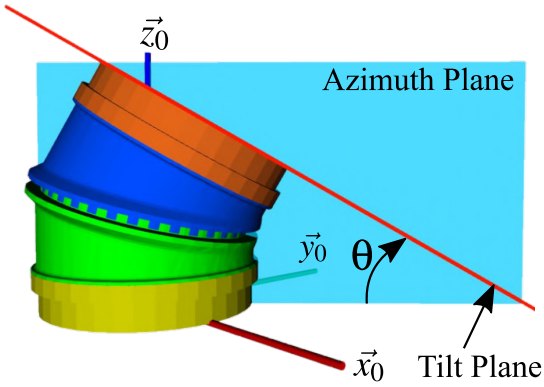


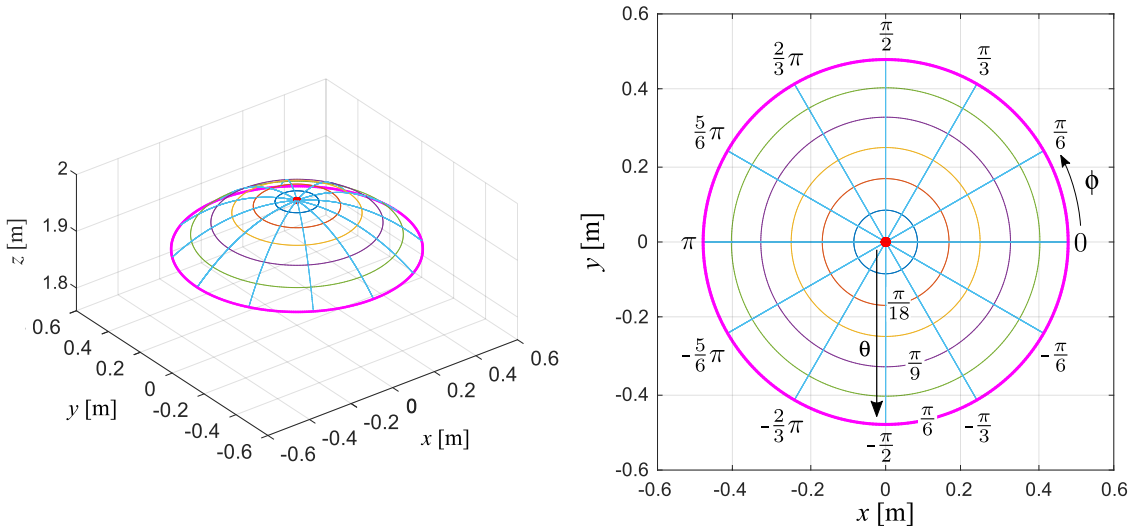
Figure 2.12: Tilt and azimuth planes on the NB-module

Plane, which is parallel to the top of the *Platform 2* and oriented along with the Azimuth Plane. The angle between the plane spanned by axes $\vec{x_0}$ and $\vec{y_0}$ and the Tilt Plane is the tilt angle θ.

2.3.3 NB-Module workspace and joint space for specific design parameter values

The NB-module workspace can be computed using the translation vector ${}^0\mathbf{p}_3$ expressed in Eq. (2.4), representing the spherical coordinates. In fact, the workspace is a portion of a sphere whose dimension depends on the length of r and the amplitude of α. Here, the design parameters are set to $r = 1$ m and $\alpha = \pi/12$ rad $= 15°$ to give an example of the NB-module workspace. Figure 2.13 shows the 3D and 2D views of the workspace. It corresponds to all the positions reached by frame $\mathcal{F}_3$ for all the possible values of q_1 and q_2 in $(-\pi, \pi]$ rad. Each point on the sphere portion can be reached by two combinations of q_1 and q_2, both of them corresponding to the same orientation of the moving platform. Figure 2.13b plots the tilt θ and azimuth ϕ angles on the 2D view of the workspace. The value of ϕ stays in the range $(-\pi, \pi]$ rad and θ in $[-\pi/6, \pi/6]$ rad. In fact, the maximum absolute possible value of θ is twice the slope of each tube, i.e. $2\alpha = \pi/6$ rad. The NB-module workspace is symmetric with respect to the z-axis.

Figure 2.14 shows the NB-module joint space. The joint space is the space of q_1 and q_2. It can be divided into two aspects because the NB-module inverse geometric model has up to two solutions. Both the areas can cover the entire workspace with the same orientation for the ending platform. There are two limit cases where the



(a) 3D view of workspace

(b) 2D view of workspace

Figure 2.13: NB-module workspace views for $r = 1$ m and $\alpha = \pi/12$ rad $= 15°$

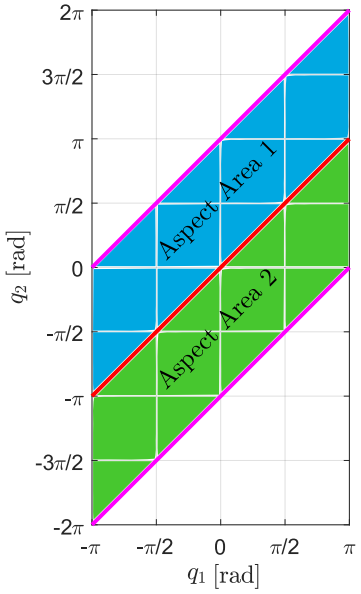


Figure 2.14: NB-module joint space with two aspect areas

module does not have strictly two solutions for a single tip pose. When the ending platform of the module is flat, i.e. $\theta = 0$, there are infinite possible couples of q_1 and q_2. In fact, the only condition that leads to $\theta = 0$ is $q_1 = q_2$. This case is underlined in red in Figs. 2.13a, 2.13b and 2.14. The second limit case is when the tilt reaches its maximum value, i.e. $|\theta| = \pi/6$ rad. Here, there exists only one possible couple of q_1 and q_2 for each ϕ with $|\theta| = \pi/6$ rad. This case happens when $|q_1 - q_2| = \pi$ rad. This case is underlined in magenta in Fig.s 2.13a, 2.13b and 2.14.

2.4 Kinematic model of NB-module

This section presents the kinematic model of the NB-module based on the parameterization defined in Fig. 2.10a. At first, the kinematic Jacobian matrix of the NB-module is computed and explained. Then, the kinematic performance are measured using a kinematic metric, setting the design parameters to specific values. The results are plotted on the workspace and joint space. Finally, a general kinematic performance analysis is performed on the NB-module as a function of its design parameters.

2.4.1 NB-module Jacobian matrix computation

Here, the NB-module kinematic Jacobian matrix $\mathbf{J}_{\text{NB}}$ computation is presented. This matrix is computed as a function of the joint values $\dot{\mathbf{q}} = [q_1, q_2]^\top$. It maps the joints velocities $\dot{\mathbf{q}} = [\dot{q}_1, \dot{q}_2]^\top$ to the NB-module tip twist $\mathbf{t} = \left[\dot{\mathbf{p}}^\top, \boldsymbol{\omega}^\top\right]^\top \in \mathbb{R}^6$ where $\dot{\mathbf{p}} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ are the linear and angular velocity vectors of frame $\mathcal{F}_3$, respectively. So, the relation between $\mathbf{t}$ and $\dot{\mathbf{q}}$ is

$$\mathbf{t} = \mathbf{J}_{\text{NB}}\ \dot{\mathbf{q}}. \tag{2.8}$$

The NB-module kinematic Jacobian matrix $\mathbf{J}_{\text{NB}}$ is computed in two steps. First of all, the kinematic Jacobian matrix $\mathbf{J}_1(\phi, \theta, r) \in \mathbb{R}^{6\times 2}$ is calculated as a function of the angles $[\phi, \theta]^\top$. This matrix maps the tilt and azimuth angles time derivatives $\left[\dot{\phi}, \dot{\theta}\right]^\top$ to the NB-module tip twist $\mathbf{t}$. The Jacobian $\mathbf{J}_1$ results to be

$$\mathbf{t} = \mathbf{J}_1(\phi, \theta, r) \begin{bmatrix} \dot{\phi} & \dot{\theta} \end{bmatrix}^\top \tag{2.9}$$

with

$$\mathbf{J}_1(\phi, \theta, r) = \begin{bmatrix} -r\sin\phi\sin\theta & r\cos\phi\cos\theta \\ r\cos\phi\sin\theta & r\sin\phi\cos\theta \\ 0 & -r\sin\theta \\ -\cos\phi\sin\theta & -\sin\phi \\ -\sin\phi\sin\theta & \cos\phi \\ 1-\cos\theta & 0 \end{bmatrix}. \tag{2.10}$$

The matrix $\mathbf{J}_1$ is derived from the geometric model presented in Fig. 2.10a and Eq. (2.5).

Then, the kinematic Jacobian matrix $\mathbf{J}_2(q_1, q_2) \in \mathbb{R}^{2\times 2}$, which maps the joints

velocities $\dot{\mathbf{q}} = [\dot{q}_1, \dot{q}_2]^\top$ to the angular velocities $[\dot{\phi}, \dot{\theta}]^\top$, is obtained upon time differentiation of Eq. (2.7), and takes the form

$$[\dot{\phi}, \dot{\theta}]^\top = \mathbf{J}_2(q_1, q_2)\,[\dot{q}_1, \dot{q}_2]^\top \tag{2.11}$$

with

$$\mathbf{J}_2(q_1, q_2) = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -b & b \end{bmatrix} \tag{2.12}$$

where

$$b = \frac{2\tan\alpha\cos\left(\dfrac{q_1 - q_2}{2}\right)}{1 + \tan^2\alpha\sin^2\left(\dfrac{q_1 - q_2}{2}\right)}. \tag{2.13}$$

The complete kinematic Jacobian matrix $\mathbf{J}_{\text{NB}}$ of the NB-module is computed as

$$\mathbf{J}_{\text{NB}} = \mathbf{J}_1\,\mathbf{J}_2. \tag{2.14}$$

2.4.2 Kinematic performance analysis for specific design parameter values

Here, the kinematic performance of the module is evaluated and, then plotted on the NB-module workspace and joint space for some specific design parameter values. The kinematic index used to evaluate the NB-module performance is called dexterity η. The dexterity $\eta(\mathbf{J})$ of a generic kinematic Jacobian matrix $\mathbf{J}$ characterizes the kinematic performance of a generic manipulator in a given configuration. It is defined as the inverse of the conditioning number $\kappa(\mathbf{J})$ [ALC92],

$$\kappa(\mathbf{J}) = ||\mathbf{J}||_2\,||\mathbf{J}^{-1}||_2 \quad \text{and} \quad \eta(\mathbf{J}) = 1/\kappa(\mathbf{J}), \tag{2.15}$$

where $||\mathbf{J}||_2$ is the 2-norm of $\mathbf{J}$. Since the 2-norm of $\mathbf{J}$ is employed, the conditioning number κ can also be defined as the ratio between the bigger $\sigma_{\max}$ and smaller $\sigma_{\min}$ singular values of $\mathbf{J}$.

$$\kappa(\mathbf{J}) = \frac{\sigma_{\max}(\mathbf{J})}{\sigma_{\min}(\mathbf{J})} \quad \text{and} \quad \eta(\mathbf{J}) = \frac{\sigma_{\min}(\mathbf{J})}{\sigma_{\max}(\mathbf{J})}. \tag{2.16}$$

The conditioning number κ is bounded by 1 and ∞. So, the dexterity η is bounded by 0 and 1. The higher η, the better the manipulator dexterity and the better the robot can move along or rotate around all directions. The manipulator reaches an

isotropic posture when $\eta = 1$. The smaller η, the worse the manipulator dexterity and the closer to a singularity. When $\eta = 0$, the robot is in a singular configuration and loses one or more degrees of freedom, meaning that a movement/rotation is not possible anymore.

The NB-module is a two degrees of freedom mechanism, providing two rotational movements around the axis $\vec{x}$ and $\vec{y}$. So, the NB-module dexterity η is computed as a function of the kinematic Jacobian matrix $\mathbf{J}_3$ that is a part of $\mathbf{J}_{\text{NB}}$ and maps $\dot{\mathbf{q}}$ to $[\omega_x, \omega_y]^\top$,

$$\begin{bmatrix} \omega_x & \omega_y \end{bmatrix}^\top = \mathbf{J}_3 \dot{\mathbf{q}} \tag{2.17}$$

where

$$\mathbf{J}_3 = \begin{bmatrix} -\cos\phi\sin\theta & -\sin\phi \\ -\sin\phi\sin\theta & \cos\phi \end{bmatrix} \mathbf{J}_2. \tag{2.18}$$

Figures 2.15 and 2.16 show the isocontours of the dexterity η plotted on the NB-module's workspace and joint space, respectively. In this case, the design parameters are again set to $r = 1$ m and $\alpha = \pi/12$ rad $= 15°$. It is apparent that the NB-module

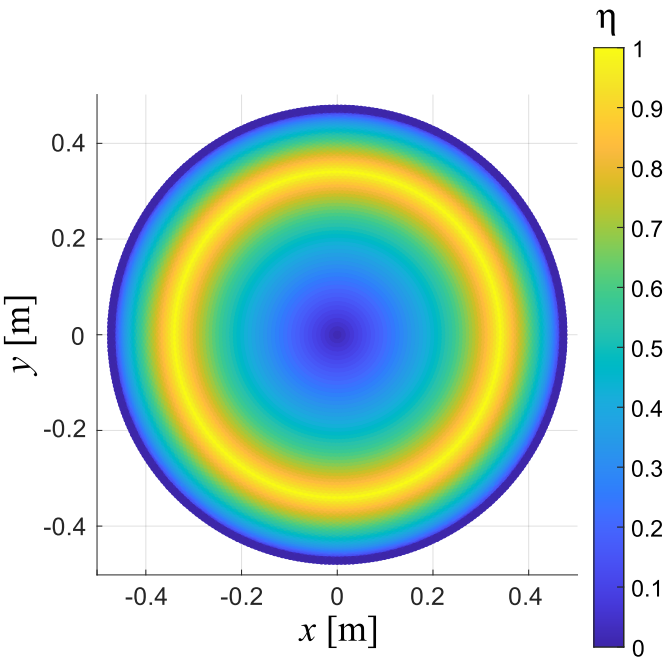


Figure 2.15: Dexterity η shown on NB-module workspace for $r = 1$ m and $\alpha = \pi/12$ rad $= 15°$

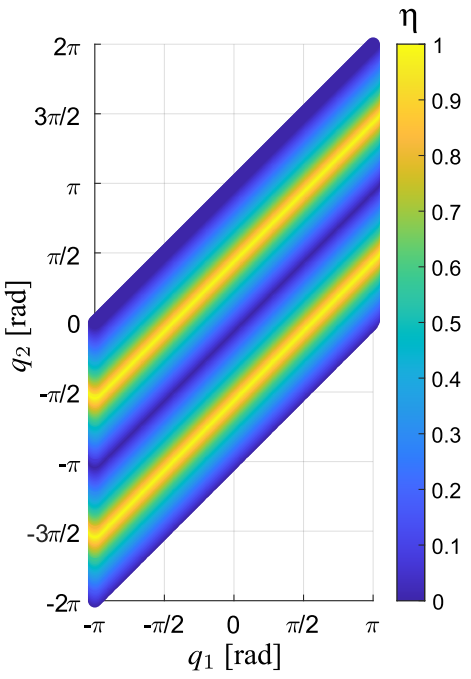


Figure 2.16: Dexterity η shown on NB-module joint space

reaches a singular configuration, i.e. $\eta = 0$, when

$$\begin{cases} \theta = 0 & \text{i.e.} \quad q_1 = q_2 \\ |\theta| = \pi/6 & \text{i.e.} \quad |q_1 - q_2| = \pi \end{cases}. \tag{2.19}$$

Furthermore, the NB-module reaches an isotropic configuration, i.e. $\eta = 1$, when

$$|\theta| = \arctan\left(\pm\frac{\sqrt{2}\,\tan\alpha}{2+\tan^2\alpha}\right) \quad \text{i.e.} \quad |q_1 - q_2| = \pi/2. \tag{2.20}$$

2.4.3 Kinematic performance analysis for generalized design parameter values

In the previous sections, the geometric and kinematic performance are analyzed setting the NB-module design parameter α to $\alpha = \pi/12$ rad $= 15°$. However, this parameter affects both the dimension of the Cartesian workspace and the kinematic performances of the NB-module. Therefore, this section focuses on the effect of α on the NB-module workspace and dexterity. Two different indices are considered in this analysis. The first one is the workspace size and the second one is the global conditioning index H. The global conditioning index H is a performance index based

on the distribution of the conditioning number of the kinematic Jacobian matrix $\mathbf{J}_3$ over the entire robot workspace [GA91]. The global conditioning index H is defined as

$$H = \frac{A}{B}, \tag{2.21}$$

where

$$A = \int_W \eta \, dW \quad \text{and} \quad B = \int_W dW, \tag{2.22}$$

which represents the sum of the dexterity η on the workspace W over the area of the workspace W. Since it is easier to perform the integration in the joint space and then transform it into the Cartesian space, A and B can be rewritten as

$$A = \int_{q_1} \int_{q_2} \eta \, |\Delta| \, dq_2 \, dq_1 \quad \text{and} \quad B = \int_{q_1} \int_{q_2} |\Delta| \, dq_2 \, dq_1. \tag{2.23}$$

The absolute value of the determinant of the kinematic Jacobian matrix $|\Delta|$ is necessary to transform the sum from the joint space into the Cartesian workspace. The variable q_1 is integrated in the range $(-\pi, \pi]$ rad while q_2 in $[q_1 - \pi, q_1 + \pi]$ rad. So, Δ and η take the form of

$$\Delta = \frac{4 \sin(q_1 - q_2) \tan^2(\alpha)}{(\tan^2(\alpha) - \cos(q_1 - q_2) \tan^2(\alpha) + 2)}, \tag{2.24}$$

and

$$\eta = |\sin(q_1 - q_2)|. \tag{2.25}$$

Interestingly, the dexterity η is independent of the value of α, while the determinant Δ is dependent. So, the isocontours of the dexterity on the joint space, shown in Fig. 2.16, are not affected by a changing value of α. On the other hand, Δ depends from α and, as follows, the isocontours of the dexterity in the Cartesian workspace change with α.

Figure 2.17 shows the graph of the global conditioning index H as a function of α. It should be noted that H remains high as long as $\alpha \leq \pi/4$ rad. Figure 2.18 depicts the surface area S of the NB-module workspace as a function of α. The design parameter r is set to 1 m. The larger α, the larger S and the closer the NB-module workspace shape to a sphere. Figure 2.19 shows the isocontours of the dexterity for three different values of α.

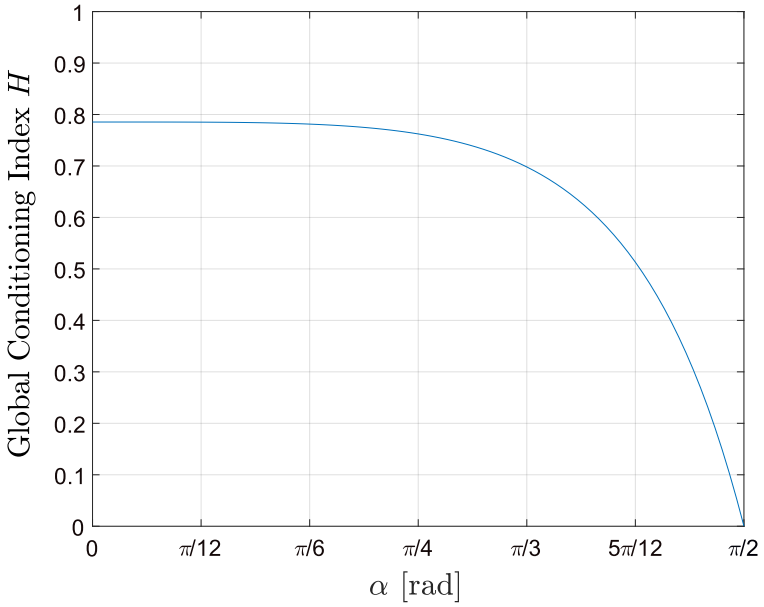


Figure 2.17: Global conditioning index H of kinematic Jacobian matrix $\mathbf{J}_3$ as a function of the NB-module tube slope α

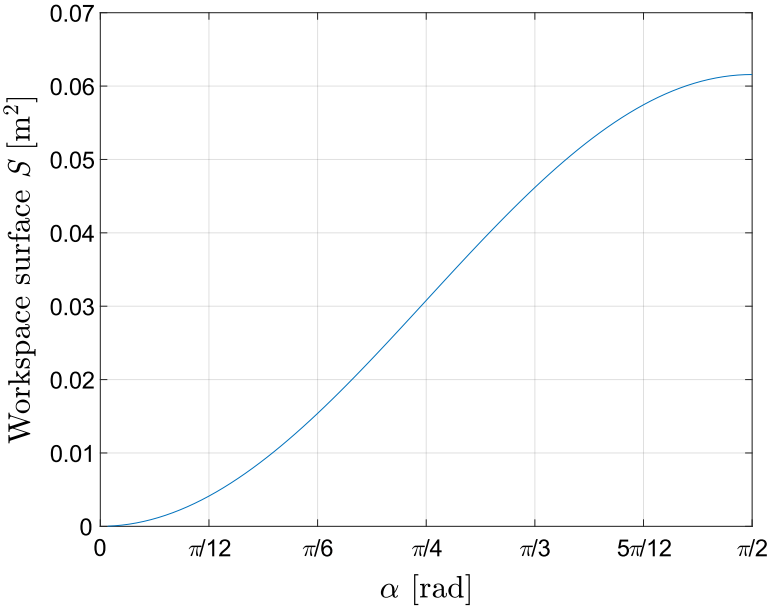
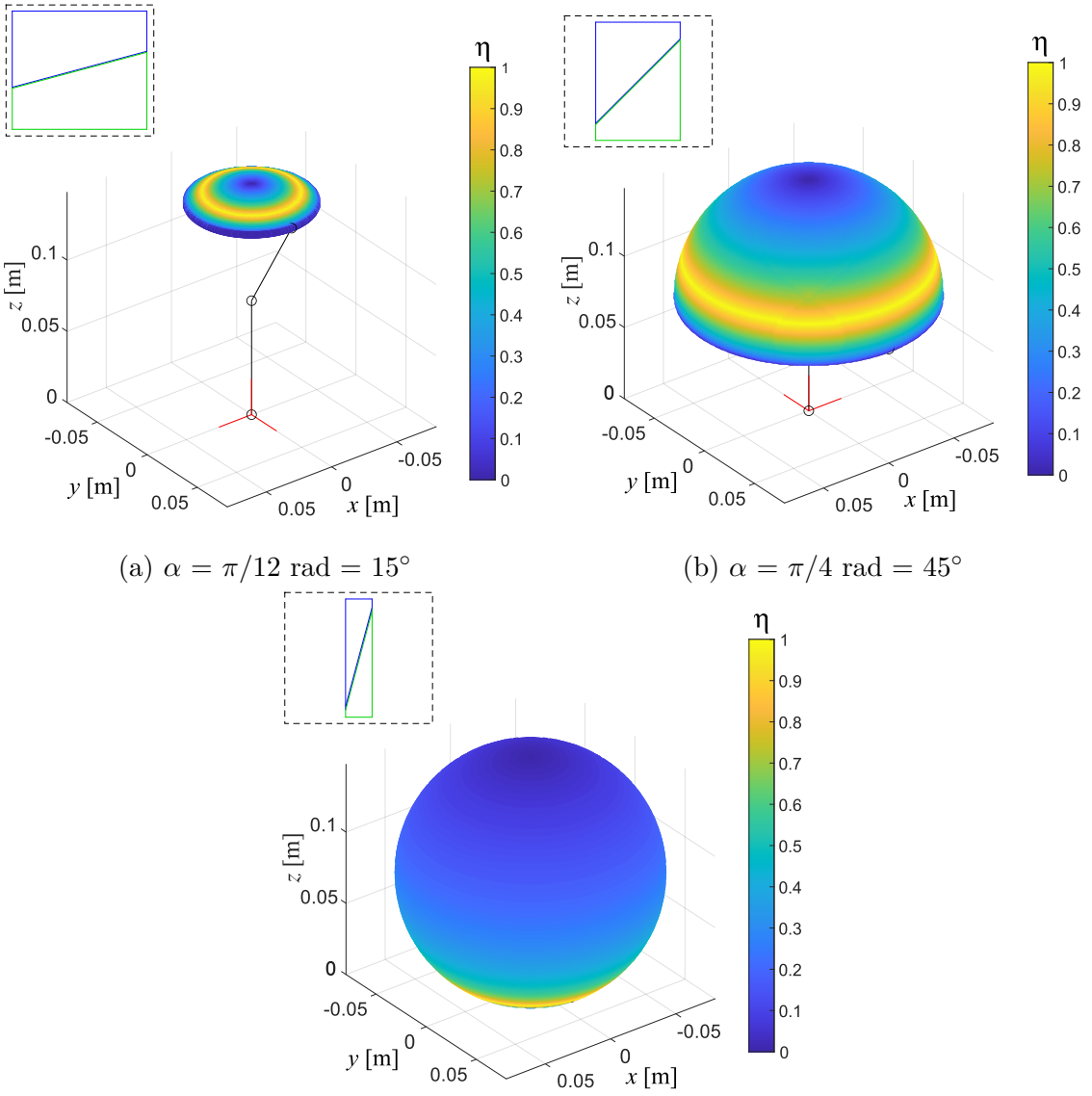


Figure 2.18: Surface area S of the NB-module workspace as a function of the NB-module tube slope α



(a) $\alpha = \pi/12$ rad $= 15°$

(b) $\alpha = \pi/4$ rad $= 45°$

(c) $\alpha = 5\pi/12$ rad $= 75°$

Figure 2.19: Dexterity η shown in the NB-module workspace and schematics of the NB-module for three values of tube slope α, origin axis in red. In the top-left corner, the configuration of *Tube 1* and *Tube 2* for the corresponding tube slope α.

2.5 Conclusions of the NB-module analyses

This chapter presented the NB-module design and features. This mechanism was built to ensure stability and flexibility to robotic manipulators. One of the most interesting feature of the NB-module is the constant velocity joint placed in the internal kinematic chain which leads to the zero-torsion characteristic. Then, its geometric and kinematic models were investigated, plotting the workspace and

joint space and its kinematic performance for specific design parameter values. As proved, the NB-module reaches isotropic configurations when $|q_1 - q_2| = \pi/2$ rad and singularities when $|q_1 - q_2| = \pi$ rad or $q_1 = q_2$. Finally, the NB-module kinematic performance was analyzed as a function of its design parameters using the global conditioning index H. The tube height variable r has no effect on the NB-module kinematic performance. On the contrary, the tube slope α affects the index H. It is shown that α equal to $\pi/4$ rad is the best trade-off between the global conditioning index H and workspace surface S. Part of the work presented in this chapter was published in [GLC+21, GCSL23a].

Chapter 3

Task Priority Based Inverse Kinematics of Redundant Manipulators

This chapter describes a task priority based kinematic control algorithm for redundant manipulators and proposes some tasks for the robot kinetostatic performance improvement. These algorithm are used to kinematically control a redundant manipulator to track a set of different trajectories while optimizing the kinetostatic performance. It is presented in Chapter 1 and shown in Fig. 1.7 on page 11. Since this robot has a high amount of degrees of freedom, it is useful to test the TPIK algorithm and kinetostatic optimization task result while tracking different trajectories. The chapter is organized as follow. Section 3.1 introduces the topic of kinematic control and trajectory tracking in case of redundancy via task augmentation. Section 3.2 presents the TPIK algorithm and its features. Section 3.3 describes the kinetostatic indices employed to improve the robot configuration and how to include these indices in the TPIK algorithm. Section 3.4 presents the trajectory tracking test of the NB-R1 for a series of trajectories. Section 3.5 analyzes and compares the obtained kinetostatic results with and without performance optimization. The conclusions related to the trajectory tracking results using the NB-R1 robot are described in Section 3.6.

3.1 Task priority based kinematic redundancy resolution

In trajectory tracking applications, the kinematic redundancy of robotic manipulators can be viewed as a possible way to improve the machine abilities and

performance [GST19]. As already introduced by Section 1.3.1, the kinematic redundancy can be used for solving several tasks simultaneously via task augmentation [SW95, SKK08]. The concept behind task enhancement consists of adding new objectives to the main task that the robot has to perform. This augmentation is helpful to exploit the degrees of freedom advance in redundant robots. It is performed in the task-space and the kinematic Jacobian matrix used to solve the main task is extended, including new rows related to the additional task solutions. The inverse, or pseudo-inverse, of the extended Jacobian provides a joint velocity solution to satisfy the simultaneous tasks [SKK08]. Different priorities can be added to the tasks based on their relevance to ensure the satisfaction of more important tasks. In this case, the solution for each task is searched in the null space of the higher priority tasks [ODAS15]. The main problem of this technique lies in the algorithmic singularities [SKK08, FDL14]. This type of singularity differs from the kinematic one and may arise even when all the considered Jacobian matrices are full rank. The algorithmic singularities appear when the extended kinematic problem is singular and the desired task velocity cannot be realized because of an incompatibility between all the tasks.

Some early works on task priority based kinematic resolution techniques were proposed in [NHY87, SS91, SK05]. These works have no defense against the kinematic and algorithmic singularities. Later, more complex methods were developed to deal with both these types of singularities. In [FDLK12], the authors proposed a so-called Saturation in the Null Space (SNS) algorithm that implements a predictive prioritizing technique for multiple tasks. This method was designed to handle the joint-space limits in the context of a single task and extended to handle prioritized tasks. The SNS algorithm shows satisfying performance, never violating the hard bounds, preserving the correct task priority hierarchy even in unfeasible cases and performing an automatic task scaling. Then, the SNS algorithm was modified in a constrained quadratic programming problem to minimize both the joint velocity norm and the task scaling, as presented in [FDL13]. The new algorithm was named Optimal Saturation in the Null Space (Opt-SNS). However, this method can still suffer from algorithmic singularities. In [FDL14], a new inverse kinematic solver, called Reverse Priority (RP), was developed to avoid the algorithmic singularities. The RP method computes joint motion contributions from the lowest-priority task to the primary task. It employs a special projection matrix to maintain the correct priority order. In [EMW14], the authors present a hierarchical quadratic programming control algorithm used to find a solution to multiple and antagonistic objectives for humanoid robot motion generation. This quadratic programming algorithm aimed

to reduce the time consumption required to solve a prioritized list of tasks. Another multiple tasks control framework is presented in [DLCA19], called Set-Based Multi-Task Priority Inverse Kinematics Framework. This method can handle both equality and inequality tasks using a priority system. For inequality tasks, the control objective has to keep the task value inside a specific interval. Additionally, the proposed multiple task method can accommodate optimization tasks that usually have lower priorities. One of the main drawbacks of all the mentioned task priority-based algorithms is that activating or deactivating one or more tasks can generate discontinuities in the joint velocity solutions [SC16].

In [SC16, SCWA18, SCWA19], the authors proposed a new kinematic control algorithm for redundant robots, named Task Priority Inverse Kinematic (TPIK). This algorithm finds the robot joint velocities that better fits the set of prioritized tasks. Each task is solved by searching for a solution in the null space of all the higher priority tasks. This control framework has a mechanism of prevention for kinematic and algorithmic singularities. Moreover, it can activate and deactivate one or more tasks without generating algorithmic discontinuities. This feature is handy for deactivating those tasks that do not require to be fulfilled at a specific moment, avoiding an over-constrain of the robotic system. In this case, it is employed to deal with the NB-R1 high redundancy for tracking some machining trajectories. In addition to the tracking trajectory task, some optimization tasks are included in the algorithm to improve the kinetostatic performance of the robot.

3.2 Task priority based inverse kinematic algorithm

This section describes the kinematic control algorithm used to kinematically control in simulation the NB-R1 while performing some tracking trajectory test. Before introducing the kinematic control algorithm, some definitions are recalled from the work [SCWA18]. The vector $\mathbf{q} \in \mathbb{R}^n$ is the joint variable vector, describing the arm configuration, where n is the number of joints. The joint velocities are collected in the vector $\dot{\mathbf{q}} \in \mathbb{R}^n$.

The notion of control objectives defines the goals of the robot. A control objective is a scalar variable $\mathrm{x}(\mathbf{q})$ computed as a function of the robot configuration vector $\mathbf{q}$ and represents the state of one task. A control objective can be of two different types, equality and inequality. Equality control objectives aim to satisfy

the relationship $\mathrm{x}(\mathbf{q}) = \mathrm{x}_0$. Inequality control objectives take the form $\mathrm{x}(\mathbf{q}) \leq \mathrm{x}_M$, or $\mathrm{x}(\mathbf{q}) \geq \mathrm{x}_m$, or both simultaneously, where x_m and x_M are the lower and upper bounds of the variable $\mathrm{x}(\mathbf{q})$ [SCWA18]. Control objectives can be divided into categories depending on their scope:

- system safety objectives, e.g. joint limits or obstacle avoidance,
- action oriented objectives, e.g. reaching a desired pose or following a desired trajectory,
- optimization objectives, e.g. minimizing the joint velocities or optimizing the kinetostatic performance metrics.

This division is purely semantic and helps identify the correct priority level for each control objective. Then, each scalar control objective is associated with a feedback reference rate $\dot{\mathrm{x}}$. The closed-loop rate control law drives the actual variable $\mathrm{x}(\mathbf{q})$ to the desired point x^* with the associated feed-forward changing rate $\dot{\mathrm{x}}^*$ and is defined as

$$\dot{\mathrm{x}} = \lambda(\mathrm{x}^* - \mathrm{x}(\mathbf{q})) + \dot{\mathrm{x}}^*, \tag{3.1}$$

where λ is a positive gain related to the target convergence rate. The actual derivative of x is defined as a function of the joint velocity vector $\dot{\mathbf{q}}$ as follows:

$$\dot{\mathrm{x}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}_{\text{task}}(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} \frac{\partial \mathrm{x}}{\partial q_1} & \dots & \frac{\partial \mathrm{x}}{\partial q_n} \end{bmatrix} \dot{\mathbf{q}}, \tag{3.2}$$

where $\mathbf{q} = [q_1 \ \dots \ q_n]$.

An activation function $a^i(\mathrm{x}) \in [0, 1]$ is associated to each control objective $\mathrm{x}(\mathbf{q})$ and represents whether the objective is relevant or not in a given time instant. The tasks associated with inequality control objectives are relevant only when the scalar variable $\mathrm{x}(\mathbf{q})$ is near or out of the validity region. So, the activation function assumes zero values within the validity region of the associated inequality objective and one when it is not, with a smooth transition between the two states. For tasks associated with equality control objectives, the activation function is set to $a^i(\mathrm{x}) = 1$ because they always need to be active.

A specific priority is assigned to each task based on the relative importance of each objective. The meaning of the priority is that the highest priority tasks are solved first using the available robot degrees of freedom and are not affected by the lower priority ones. Hence, lower priority tasks are solved if enough robot degrees of freedom remains. When two or more tasks have the same priority, they are grouped

in a multidimensional control task. A specific list of prioritized tasks is called control action $\mathscr{A}$.

With the previous definitions, the following quantities associated with each priority level in a control action $\mathscr{A}$ can be defined [SCWA19]:

- $\dot{\bar{\mathbf{x}}}_k = [\dot{\bar{\mathrm{x}}}_{1,k},\ \dot{\bar{\mathrm{x}}}_{2,k},\ \ldots\ ,\ \dot{\bar{\mathrm{x}}}_{m_k,k}]^\top$ is the vector collecting all the reference rates of the scalar control tasks, where m_k is the number of scalar tasks for the priority level k.
- $\mathbf{J}_k$ is the Jacobian matrix associated with the k^{th} task vector $[\dot{\mathrm{x}}_{1,k},\ \ldots\ ,\ \dot{\mathrm{x}}_{m_k,k}]^\top$ with respect to the joint velocity vector $\dot{\mathbf{q}}$.
- $\mathbf{A}_k = \mathrm{diag}(a_{1,k},\ \ldots\ ,\ a_{m_k,k})$ is a diagonal matrix of the activation functions.

To find the system velocity reference vector $\dot{\bar{\mathbf{q}}}$ that meets the priority requirements of a given action, the TPIK algorithm solves a sequence of nested minimization problems

$$S_k = \arg\ \mathrm{R} - \min_{\dot{\bar{\mathbf{q}}} \in S_{k-1}} ||\mathbf{A}_k(\dot{\bar{\mathbf{x}}}_k - \mathbf{J}_k\dot{\bar{\mathbf{q}}})||^2, \tag{3.3}$$

where S_{k-1} is the manifold of all the previous priority level solutions. The notation $\mathrm{R} - \min$ highlights that each minimization is performed through specific regularized space projections to implement priorities among the tasks defined in [SC16]. In addition to Eq. (3.3), other regularization costs are included. These regularization costs avoid discontinuities in the system velocity vector due to kinematic and algorithmic singularities. In [SC16], the authors fully describe these regularization costs that are not analyzed here.

A significant advantage of the TPIK algorithm is the use of the activation functions to handle inequality control objectives without over-constraining the system. Both equality and inequality control require a certain amount of robot degrees of freedom specified by the associated task. When an inequality task is inside its validity region, the activation function goes to zero, therefore not consuming any degrees of freedom. So, safety tasks, like joint limits, can be placed at the top of the hierarchy without over-constraining the system.

Finally, the TPIK algorithm adopts another continuous sigmoidal function $a^P(\mathbf{P})$ to perform a smooth activation/deactivation transition between two actions. This function is related to the vector $\mathbf{P}$ that includes the previous and current executed actions and the time elapsed in the actual step. This function $a^P(\mathbf{P})$ is used together with $a^i(\mathrm{x})$. More details are presented in [SC16].

3.3 Proposed tasks for robot kinetostatic performance optimization

In the proposed case, the TPIK algorithm employs some tasks based on kinetostatic performance indices to optimize the robot configuration while performing the desired application. These indices are dexterity, manipulability and robot transmission ratio. They are defined starting from the kinematic Jacobian matrix $\mathbf{J}_e$ that relates the end-effector velocity with respect to the robot base. The kinematic Jacobian matrix $\mathbf{J}_e$ can be written as

$$\mathbf{t} = \begin{bmatrix} \dot{\mathbf{p}} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{J}_e(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{J}_l(\mathbf{q}) \\ \mathbf{J}_a(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}}, \tag{3.4}$$

where $\mathbf{t} = \left[\dot{\mathbf{p}}^\top, \boldsymbol{\omega}^\top\right]^\top \in \mathbb{R}^6$ is the robot end-effector twist, with $\dot{\mathbf{p}} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ the linear and angular velocity vectors of the end-effector, respectively. Since the kinematic Jacobian matrix $\mathbf{J}_e$ contains non-homogeneous terms, namely linear and angular, it needs to be weighted to compute the kinetostatic performance indices correctly. The weighting of $\mathbf{J}_e$ employs the characteristic length L that was introduced in [Ang92] to solve the absence of dimensional homogeneity in the kinematic Jacobian matrix entries and is computed in [KA05]. To weight $\mathbf{J}_e$, the revolute joint columns of the linear kinematic Jacobian matrix part are divided by L. The weighted kinematic Jacobian matrix is written as $\mathbf{J}_w$. The weighting is a critical issue when analyzing the kinetostatic performance of $\mathbf{J}_e$ [KAW15, ZKA12]. In the rest of this Section, the three kinetostatic indices are presented and their Jacobian matrix is explained.

3.3.1 Manipulability

The manipulability is an index that measures the kinematic abilities of the robotic system through its weighted kinematic Jacobian matrix $\mathbf{J}_w$ [Yos85]. The manipulability of a manipulator is defined as

$$\mu = \sqrt{\det(\mathbf{J}_w \mathbf{J}_w^\top)}, \tag{3.5}$$

and amounts to the product of all the singular values of $\mathbf{J}_w$. The higher the manipulability value, the larger the manipulability hyper-ellipsoid and the better the kinematic performance of the mechanism [Ang03]. It should be noted that the ma-

nipulator reaches a kinematic singularity when μ vanishes.

The derivative of the manipulability as a function of the joint variables is explained in [Par00] and used in [MKYC02]:

$$\frac{\partial \mu}{\partial q_i} = \mu \ \text{trace}\left\{ \frac{\partial \mathbf{J}_w}{\partial q_i} \mathbf{J}_w^+ \right\}, \tag{3.6}$$

where the matrix $\mathbf{J}_w^+$ is the pseudo-inverse of the weighted kinematic Jacobian matrix [SC16]. Hence, the manipulability Jacobian matrix $\mathbf{J}_\mu$ as a function of the joint variables is:

$$\mathbf{J}_\mu = \begin{bmatrix} \frac{\partial \mu}{\partial q_1} & \dots & \frac{\partial \mu}{\partial q_n} \end{bmatrix}, \tag{3.7}$$

where n, which represents the number of columns of $\mathbf{J}_w$, is the dimension of joint space.

3.3.2 Dexterity

The dexterity index was already presented in Section 2.4.2 to analyze the NB-module kinematic performance. Here, the dexterity $\eta(\mathbf{J}_w)$ characterizes the kinematic performance of a complete manipulator in a given configuration. It is defined as the inverse of the conditioning number $\kappa(\mathbf{J}_w)$ of its weighted kinematic Jacobian matrix $\mathbf{J}_w$ [ALC92]:

$$\kappa(\mathbf{J}_w) = ||\mathbf{J}_w||_2 \, ||\mathbf{J}_w^{-1}||_2 \tag{3.8}$$

and

$$\eta(\mathbf{J}_w) = 1/\kappa(\mathbf{J}_w). \tag{3.9}$$

To recall, the index η is bounded by 0 and 1. The higher η, the better the manipulator dexterity. The manipulator reaches an isotropic posture when $\eta = 1$. The smaller η, the worse the manipulator dexterity and the closer to a singularity. Moreover, η can be defined as the ratio between the smallest and highest singular values of $\mathbf{J}_w$ indicating how close the manipulability hyper-ellipsoid is to being a hyper-sphere [PC06].

The formula proposed in Eq. (3.9) can not be derived because it is not in an analytical form. So, the Frobenius norm of $\mathbf{J}_w$ can be used [RCC08] to obtain the analytical expression of η:

$$\eta(\mathbf{J}_w) = \frac{m}{\sqrt{\text{trace}(\mathbf{J}_w \mathbf{J}_w^\top) \ \text{trace}[(\mathbf{J}_w \mathbf{J}_w^\top)^{-1}]}}, \tag{3.10}$$

where m, which represents the number of rows of $\mathbf{J}_w$, is the dimension of the task space. The following definitions are introduced to improve the readability of the equations:

$$\gamma_1 \triangleq \sqrt{\text{trace}(\mathbf{J}_w \mathbf{J}_w^\top)} \tag{3.11}$$

and

$$\gamma_2 \triangleq \sqrt{\text{trace}[(\mathbf{J}_w \mathbf{J}_w^\top)^{-1}]}, \tag{3.12}$$

where $\triangleq$ is the define operator. With these definition, it follows that

$$\eta(\mathbf{J}_w) = \frac{m}{\gamma_1(\mathbf{J}_w)\,\gamma_2(\mathbf{J}_w)}. \tag{3.13}$$

Then, the dexterity Jacobian matrix is determined to relate the velocity rate of η with respect to the joint velocity vector $\dot{\mathbf{q}}$. The Frobenius formula used in Eq. (3.10) expresses η as a function of joint position vector $\mathbf{q}$ in an analytical way allowing its derivation. So, the derivative of Eq. (3.10) with respect to each joint position $q_i \in \mathbf{q}$ is

$$\frac{\partial \eta}{\partial q_i} = -\eta \left(\frac{\partial \gamma_1}{\partial q_i} \frac{1}{\gamma_1} + \frac{1}{\gamma_2} \frac{\partial \gamma_2}{\partial q_i} \right), \tag{3.14}$$

where

$$\frac{\partial \gamma_1}{\partial q_i} = \frac{1}{\gamma_1} \text{trace} \left\{ \mathbf{J}_w \frac{\partial \mathbf{J}_w^\top}{\partial q_i} \right\} \tag{3.15}$$

and

$$\frac{\partial \gamma_2}{\partial q_i} = \frac{1}{\gamma_2} \text{trace} \left\{ -\mathbf{J}_w \frac{\partial \mathbf{J}_w^\top}{\partial q_i} (\mathbf{J}_w \mathbf{J}_w^\top)^2 \right\}. \tag{3.16}$$

In conclusion, the dexterity Jacobian matrix $\mathbf{J}_\eta$ as a function of the joint variables is:

$$\mathbf{J}_\eta = \begin{bmatrix} \frac{\partial \eta}{\partial q_1} & \dots & \frac{\partial \eta}{\partial q_n} \end{bmatrix}, \tag{3.17}$$

where n is the number of columns of $\mathbf{J}_w$ and dimension of joint space.

3.3.3 Robot transmission ratio

The robot transmission ratio (RTR) $\rho(\mathbf{J}_w)$ quantifies the effectiveness of the actuator force in producing a prescribed robot motion [ZKA12]. It corresponds to the angle between the joint velocity $\dot{\mathbf{q}}$ and torque $\boldsymbol{\tau}$ vectors in the joint space and is

defined as

$$\rho = \frac{|\boldsymbol{\tau}^\top \dot{\mathbf{q}}|}{||\boldsymbol{\tau}||\,||\dot{\mathbf{q}}||} = |\cos\angle(\boldsymbol{\tau}, \dot{\mathbf{q}})|. \tag{3.18}$$

This metric is bounded between 0 and 1. In case of kinetostatic redundancy, ρ can be expressed in terms of the end-effector twist $\mathbf{t}$ and the wrench $\mathbf{w}$ applied to it, leading to

$$\rho = \frac{|\mathbf{w}^\top \mathbf{t}|}{||\mathbf{J}_w^\top \mathbf{w}||\,||\mathbf{J}_w^+ \mathbf{t}||}, \tag{3.19}$$

where $\mathbf{t}$ is the robot end-effector twist defined in Eq. (3.4) and $\mathbf{w} = [\mathbf{f}^\top, \mathbf{m}^\top]^\top$ is the wrench that collects the forces $\mathbf{f}$ and moments $\mathbf{m}$ exerted by the environment on the end-effector. To ensure that ρ is dimensionless, the linear part $\dot{\mathbf{p}}$ in $\mathbf{t}$ and the moment $\mathbf{m}$ in $\mathbf{w}$ are divided by the characteristic length L.

The RTR Jacobian matrix is obtained upon differentiation of Eq. (3.19) with respect to each joint position $q_i \in \mathbf{q}$:

$$\frac{\partial \rho}{\partial q_i} = \rho\, \frac{\mathbf{w}^\top \mathbf{J}_w \frac{\partial \mathbf{J}_w^\top}{\partial q_i}\mathbf{w}||\mathbf{J}_w^+\mathbf{t}||^2 - ||\mathbf{J}_w^\top \mathbf{w}||^2 \mathbf{t}^\top \mathbf{J}_w^{+\top} \frac{\partial \mathbf{J}_w^+}{\partial q_i}\mathbf{t}}{(||\mathbf{J}_w^\top \mathbf{w}||\,||\mathbf{J}_w^+ \mathbf{t}||)^2}, \tag{3.20}$$

where the values of the end-effector twist $\mathbf{t}$ and wrench $\mathbf{w}$ are known from the trajectory planning. The pseudo-inverse weighted kinematic Jacobian matrix derivative $\partial \mathbf{J}_w^+ / \partial q_i$ is defined in [GP73]. The RTR Jacobian matrix $\mathbf{J}_\rho$ as a function of the joint variables is:

$$\mathbf{J}_\rho = \begin{bmatrix} \frac{\partial \rho}{\partial q_1} & \cdots & \frac{\partial \rho}{\partial q_n} \end{bmatrix}, \tag{3.21}$$

where $n = \text{columns}(\mathbf{J}_w)$ is the dimension of joint space.

3.4 Trajectory tracking test in simulation description

This section describes the tests performed in a computer simulation on the NB-R1 robot, shown in Fig. 1.7 on page 11. The test consists in making the NB-R1 track different trajectories with and without the tasks related to dexterity, manipulability and RTR collecting their values. Both the dexterity and the manipulability are kinematic performance indices. Maximizing them simultaneously forces the manipulability hyper-ellipsoid to be as big as possible and close to a hyper-sphere. So, ideally, the robot will be able to move with the same higher velocity amplification factor in all directions while reducing actuator velocity limits. Moreover, the

RTR index forces the joint velocity and torque vectors to align, making the robot movement along the desired direction more effective. After collecting the metric values on each trajectory, these are compared to demonstrate the benefit of using the optimization tasks and identify the best robot configuration series to follow each trajectory.

The NB-R1 has to track four trajectories of the same shape and size. Figure 3.1 shows the NB-R1 next to the four trajectories. Two of them are oriented horizontally and the others are vertical. These trajectories describe a cubic area whose side are 0.5 m × 0.5 m centered in $(x, y, z) = (0.0, 1.05, 0.45)$. The machining tool is shown in the top right corner of Fig. 3.1. The tool ending part is rotated of 45° around the red point. This allows the robot to reach all the points on each trajectory. The trajectories are planned to cut a squared shape using a machining tool and the measures are shown in Fig. 3.2. The tool trajectory is divided in four parts (a), (b), (c) and (d). Figure 3.2 also shows the orientation of the velocity vector $\vec{\mathbf{v}}$ and the tangential and radial force vectors $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ applied on the machining tool. The magnitudes of $\vec{\mathbf{v}}$, $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ are constant along the entire trajectory. The profiles of $\vec{\mathbf{v}}$, $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ are depicted in Fig. 3.3. The gravity force and the cutting one along axis $\vec{z}_p$ are neglected in this work. The details about NB-R1 and trajectory features are in Tables 3.1 and 3.2, respectively. The details of the machine and the implementation are given

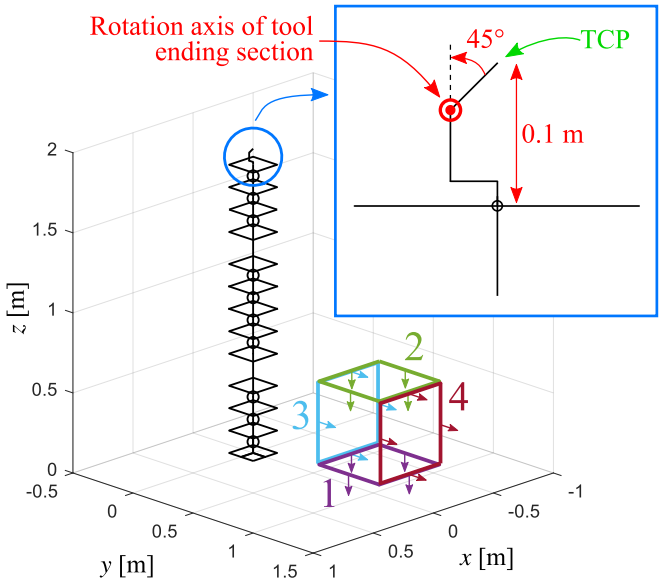


Figure 3.1: Schematics of the NB-R1 with the four trajectories to track in 3D space. In the top right corner, the machining tool attached to the NB-R1 end-effector used in the cutting phase. Tool contact point (TCP) highlighted in green. Tool ending section rotated around red point of 45°.

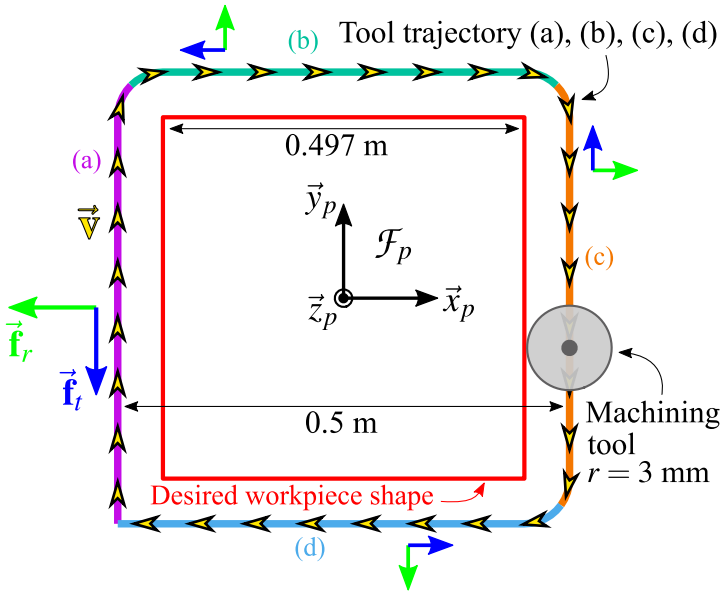
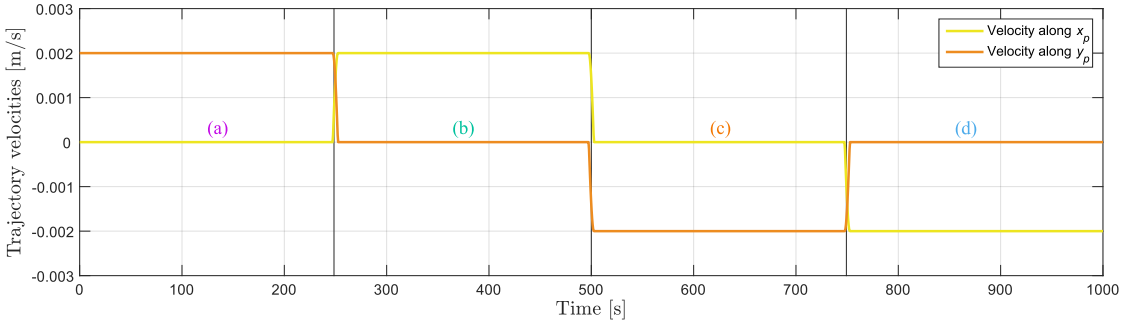
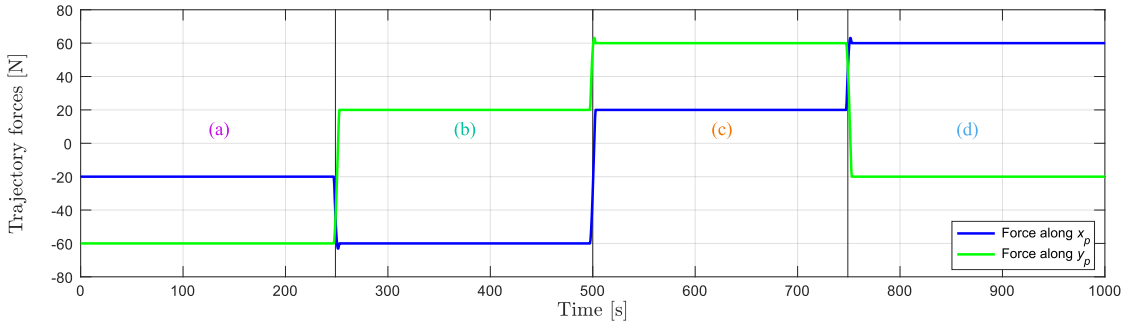


Figure 3.2: Measures of the desired workpiece, machining tool and trajectory tool with workpiece frame $\mathcal{F}_p$. Orientation of velocity vector $\vec{\mathbf{v}}$ (yellow) plus tangential and radial force vectors $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ (blue and green). The tool trajectory is divided into four parts (a), (b), (c) and (d).



(a) Velocity profiles



(b) Cutting force profiles

Figure 3.3: Velocity and cutting force profiles applied to the machining tool in frame $\mathcal{F}_p$. Each sector is labeled (a), (b), (c) and (d) to match the corresponding trajectory part.

Table 3.1: Robot dimensions plus joint velocity and acceleration limits

NB-module half height r	0.07 m
NB-module tube slope α	15°
Link length	0.2 m
Tool height	0.1 m
Tool offset	45°
Robot + tool total height	1.9 m
Max/min joint velocity	±1.0 rad/s
Max absolute joint acceleration/deceleration	2.0 rad/s^2

Table 3.2: Test trajectory details, velocities and forces exerted on end-effector and time for tracking entire trajectory

Square side	0.5 m
Steps	401
Magnitude velocity vector $\|\|\vec{\mathbf{v}}\|\|_2$	0.002 m/s
Magnitude tangential force vector $\|\|\vec{\mathbf{f}}_t\|\|_2$	60 N
Magnitude radial force vector $\|\|\vec{\mathbf{f}}_r\|\|_2$	20 N
Time	1000 s

Table 3.3: Machine and test implementation details

Operating System	Linux
CPUs number	4
CPU model	Intel Core i7 10th Gen, 1.30GHz
Language	C++
Control frequency	10Hz
Time to track one trajectory	5 s

in Table 3.3. The NB-R1 features, trajectory size and velocity/force magnitudes were provided by Nimbl'Bot.

A convex combination ϵ of the three kinetostatic performance indices is used to rate and compare the NB-R1 kinetostatic performance. It should be noted that η and ρ are bounded between $[0, 1]$ whereas μ is not bounded, $[0, \infty)$. So, it is necessary to bound μ in the range $[0, 1]$ before writing the convex combination. A new index called bounded manipulability ν is defined as:

$$\nu = 1 - \frac{1}{1 + \mu}. \tag{3.22}$$

When $\mu = 0$ then $\nu = 0$, and when $\mu = \infty$ then $\nu = 1$. Now, η, ν and ρ are all bounded between 0 and 1 and can then be used in a convex combination:

$$\epsilon(\eta, \nu, \rho) = \lambda_1\eta + \lambda_2\nu + \lambda_3\rho, \tag{3.23}$$

where λ_1, λ_2 and λ_3 are scaling factors. Since all the kinetostatic performance indices are valid in the same range, the weighting factors λ_1, λ_2 and λ_3 are selected equal to $= 1/3$. So, ϵ becomes valid in the same range $[0, 1]$. When $\epsilon = 1$, the robot is in an isotropic configuration, the manipulability hyper-ellipsoid becomes a hyper-sphere and the angle between the joint velocity and torque vectors tends to 0°.

Four different actions are used in the simulation. When the kinetostatic optimization tasks, namely dexterity, manipulability and RTR, are deactivated, $\mathscr{A}_1$ is the action used to reach the starting pose and $\mathscr{A}_2$ to follow the trajectory. When the

Table 3.4: Details about the task names, control objective types, and hierarchy levels. Symbol (E) represents the equality control objective tasks and (I) the inequality ones. The last four columns list the hierarchy level for each task in actions $\mathscr{A}_1$ (Reach Pose), $\mathscr{A}_2$ (Follow Trajectory), $\mathscr{A}_3$ (Reach Pose Optimized) and $\mathscr{A}_4$ (Follow Trajectory Optimized). When symbol "/" is used, it means that a task is not present in the action and has no hierarchy level.

			Hierarchy levels			
Task	Category	Type	$\mathscr{A}_1$	$\mathscr{A}_2$	$\mathscr{A}_3$	$\mathscr{A}_4$
End-Effector Pose	action oriented	E	1^{st}	/	1^{st}	/
End-Effector Velocity	action oriented	E	/	1^{st}	/	1^{st}
Dexterity	optimization	I	/	/	2^{nd}	2^{nd}
Manipulability	optimization	I	/	/	2^{nd}	2^{nd}
RTR	optimization	I	/	/	2^{nd}	2^{nd}

optimization tasks are activated, $\mathscr{A}_3$ brings the robot to the starting pose and $\mathscr{A}_4$ follows the trajectory. Table 3.4 shows the task details and their hierarchy inside each action.

The tests are developed as follows. The robot is started from a random configuration and reaches the starting point on one trajectory. From here, it tracks the entire trajectory and collects dexterity, bounded manipulability and RTR values at

Algorithm 3.1 Procedure to collect kinetostatic optimized and not optimized results during machining operations

Require: Actions $\mathscr{A}_1$ = Reach Pose, $\mathscr{A}_2$ = Follow Trajectory, $\mathscr{A}_3$ = Reach Pose Optimized and $\mathscr{A}_4$ = Follow Trajectory Optimized, details in Table 3.4.

Variables: Number of trajectories (m_t) is 4 and number of repetitions (m_r) is 100. Variable t is time instant.

1: **for** $i := 1 \rightarrow m_t$ **do**
2: **for** $j := 1 \rightarrow m_r$ **do**
3: Randomly initialize starting robot configuration vector $\mathbf{q}_0$.
4: Save configuration vector $\mathbf{q}_0$ saved.
5: Load i^{th} trajectory.
6: **while** End-effector not on i^{th} trajectory starting pose **do**
7: Run action $\mathscr{A}_1$ to reach i^{th} trajectory starting pose.
8: **end while**
9: **while** End-effector not on i^{th} trajectory ending pose **do**
10: Run action $\mathscr{A}_2$ to move robot to next pose at t.
11: Compute $\epsilon(\mathbf{q}, t)$ for robot configuration $\mathbf{q}$ at t.
12: Save not optimized $\epsilon(\mathbf{q}, t)$, $\eta(\mathbf{q}, t)$, $\mu(\mathbf{q}, t)$, $\rho(\mathbf{q}, t)$ for j^{th} repetition.
13: **end while**
14: Restart robot in $\mathbf{q}_0$, generated at step 3.
15: **while** End-effector not on i^{th} trajectory starting pose and $\dot{\eta}$,$\dot{\nu}$,$\dot{\rho} > \delta$ **do**
16: Run action $\mathscr{A}_3$ to reach i^{th} trajectory starting pose.
17: **end while**
18: **while** End-effector not on i^{th} trajectory ending pose **do**
19: Run action $\mathscr{A}_4$ to move robot to next pose at time instant t.
20: Compute $\epsilon(\mathbf{q}, t)$ for robot configuration $\mathbf{q}$ at time instant t.
21: Save optimized $\epsilon(\mathbf{q}, t)$, $\eta(\mathbf{q}, t)$, $\nu(\mathbf{q}, t)$, $\rho(\mathbf{q}, t)$ for j^{th} repetition.
22: **end while**
23: **end for**
24: **end for**

each step. These actions are repeated hundred times starting from different random robot configurations to obtain a general pool of results without using the optimization tasks. Then, the same process is repeated with the dexterity, manipulability and RTR tasks, both when approaching the starting point and following the trajectory. The robot is initialized using the same random configurations used in the tests without optimization. Moreover, when the optimization tasks are employed, a monitoring is added while reaching the starting point to check if the control algorithm is still optimizing the robot configuration even though the end-effector frame has already reached the desired pose. So, the robot stats tracking the desired trajectory only when the kinetostatic index velocities $\dot{\eta}$, $\dot{\nu}$ and $\dot{\rho}$ are under a certain threshold $\delta = 10^{-6}$. This methodology is applied to each trajectory. A summary of the methodology is presented in Algorithm 3.1.

3.5 Trajectory tracking test in simulation results

This section describes all the results collected during the simulation tests. The results obtained with and without kinetostatic optimization tasks are compared to demonstrate the improvements made. Figure 3.4 collects the values of ϵ for the robot on the starting poses with and without optimization tasks. For each trajectory 1 to 4, two box plots are shown containing the results obtained by repeating the process without (blue) and with (red) optimization a hundred times. When the dexterity,

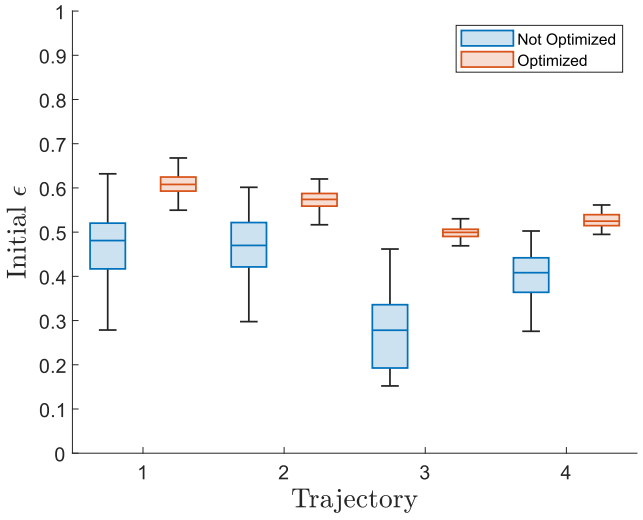


Figure 3.4: Values taken by ϵ at the starting pose of each trajectory (1 to 4, Fig. 3.1) for each one of the hundred repetitions, without (blue) and with (red) kinetostatic optimization

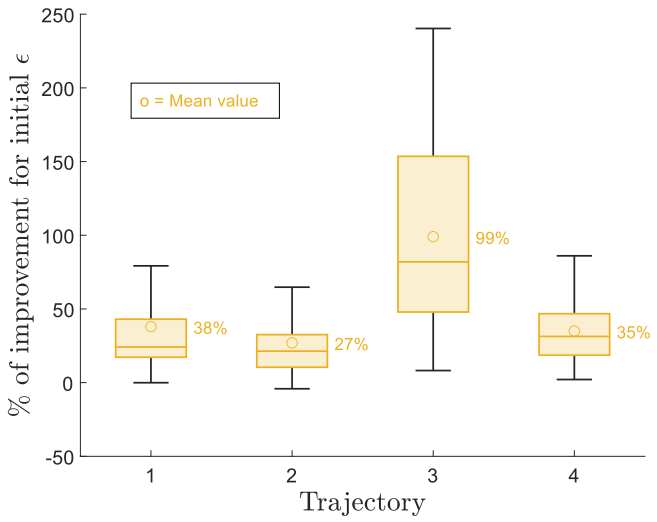


Figure 3.5: Percentages of optimization for ϵ in the starting pose of each trajectory comparing the optimized and non-optimized case for each of the hundred repetitions. The circle $\circ$ highlights the mean percentage of the hundred repetitions.

manipulability and RTR tasks are used, the variance of ϵ is smaller and the minimum and maximum values are high. This means that the optimization tasks always help reaching configurations with high values of ϵ. On the contrary, when the dexterity, manipulability and RTR tasks are removed, the variance of ϵ on the starting poses is bigger. Since the optimization tasks are disabled, the TPIK algorithm runs the robot straightly to the desired pose without performing any optimization on the robot configuration and ϵ can reach higher or lower values. So, the optimization task use provides an improvement to the robot performance without affecting the total simulation time. In fact, the average time for running the TPIK algorithm at each step is 698 μs without the optimization tasks and 744 μs with them. The computational time difference is negligible. Figure 3.5 presents the improvement of ϵ at the starting pose of each trajectory, with and without the optimization tasks for each of the hundred repetitions. The improvement of ϵ are almost always high. For the third trajectory, the range is larger than the other trajectories, over 200% in some cases. This happens because some non-optimized tests on the third trajectory reached very low kinetostatic values compared to the optimized tests, as shown in Fig. 3.4. It can be noticed that few cases show a negative percentage. This is due to the control algorithm converging to a local maxima when optimizing the metrics. In fact, the optimized control algorithm reaches a local maxima in these few cases while the non-optimized one moves the robot in a configuration that escapes the local maxima, reaching higher kinetostatic performance unintentionally. This behavior happens a few times, which justifies the need to run the algorithm several

times to obtain the best robot performance.

Figure 3.6 collects the mean values $\bar{\epsilon}$ of ϵ reached on each trajectory. Again, for each trajectory 1 to 4, there are two box plots containing the results obtained by repeating a hundred times the process without (blue) and with (red) optimization. The optimization tasks lead to a smaller variance for $\bar{\epsilon}$ compared to the non-optimized results. However, the minimum values of $\bar{\epsilon}$ in the non-optimized cases are higher than the minimum ϵ obtained on the non-optimized starting poses. This means that

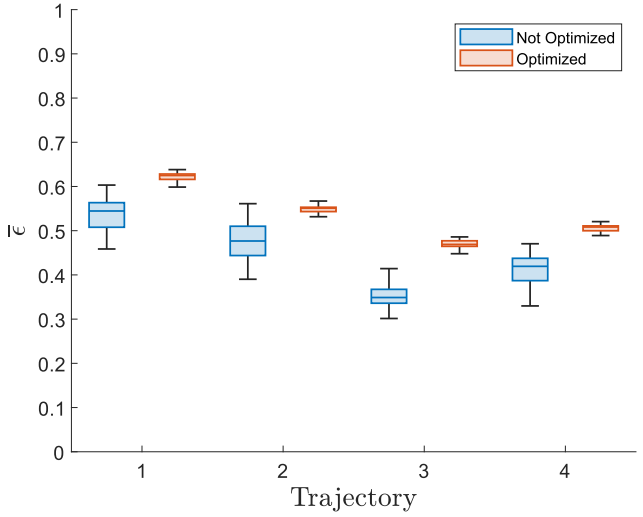


Figure 3.6: Mean values $\bar{\epsilon}$ of ϵ reached along each trajectory (1 to 4, Fig. 3.1) for each one of the hundred repetitions, without (blue) and with (red) kinetostatic optimization

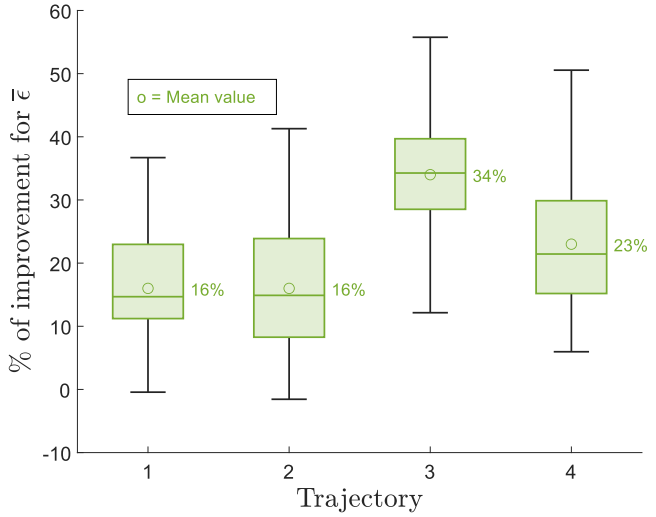
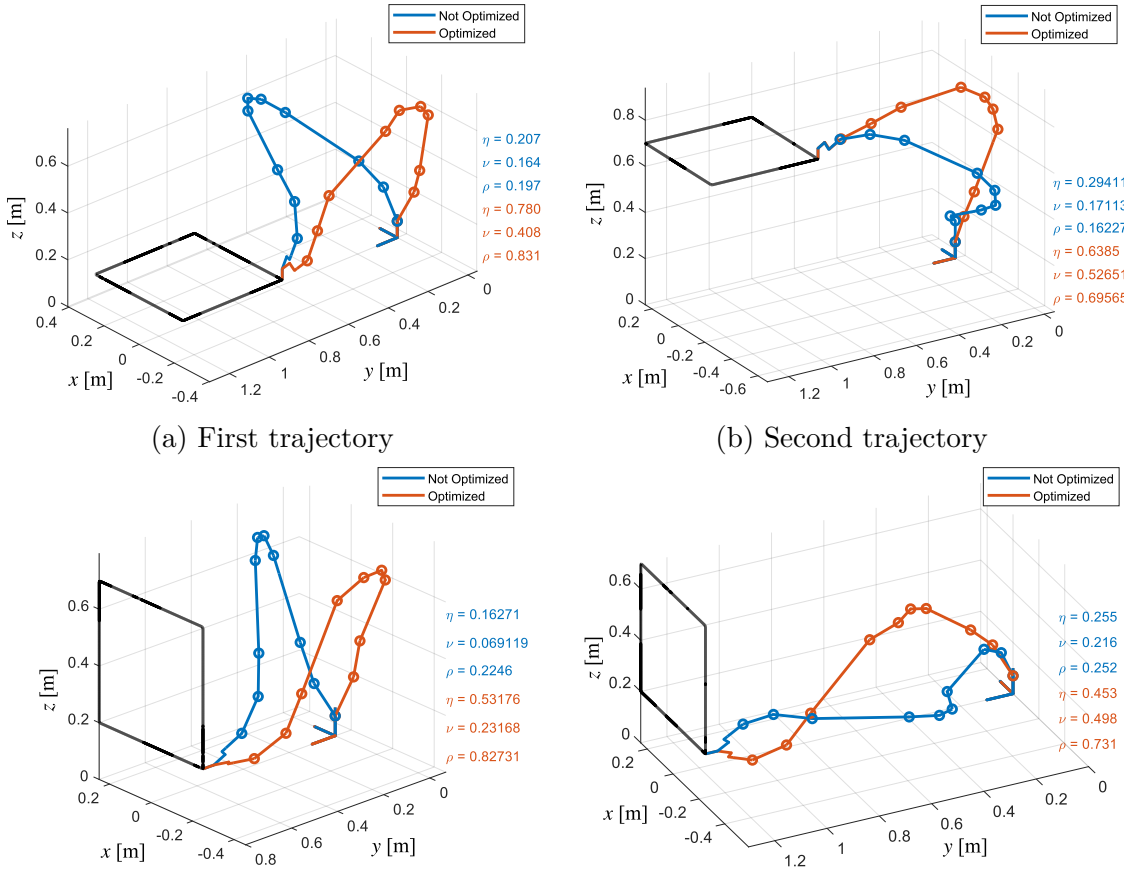


Figure 3.7: Percentages of optimization for mean value $\bar{\epsilon}$ of ϵ on each trajectory comparing the optimized and non-optimized case for each of the hundred repetitions. The circle ∘ highlights the mean percentage of the hundred repetitions.

the NB-R1 can maintain good kinetostatic performance on the trajectories even if the optimization tasks are not used and the starting ϵ is low. Moreover, the first trajectory shows the best performance, followed by the second, the third and the fourth. So, this design has better kinetostatic performance when the trajectory is horizontal and closer to the base. Figure 3.7 presents the improvement of ϵ on each trajectory, with and without the optimization tasks for each of the hundred repetitions. Again, it can be noticed that the percentage is negative in few cases due local maxima issue.

Figure 3.8 shows the robot on the starting pose of each trajectory for the minimum value of ϵ in case of no optimization and the maximum ϵ when optimization tasks were used. The values η, ν and ρ for the optimized and not optimized configurations are shown next to the robots in each figure. In the configurations assumed by the optimized robots, the x pose difference of one NB-module center and the next is



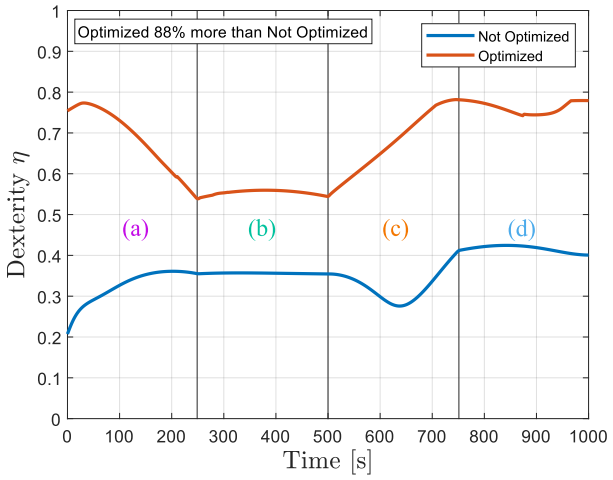
(a) First trajectory

(b) Second trajectory

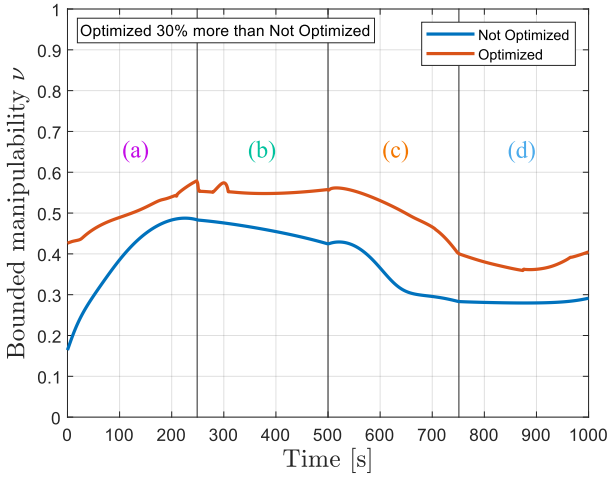
(c) Third trajectory

(d) Fourth trajectory

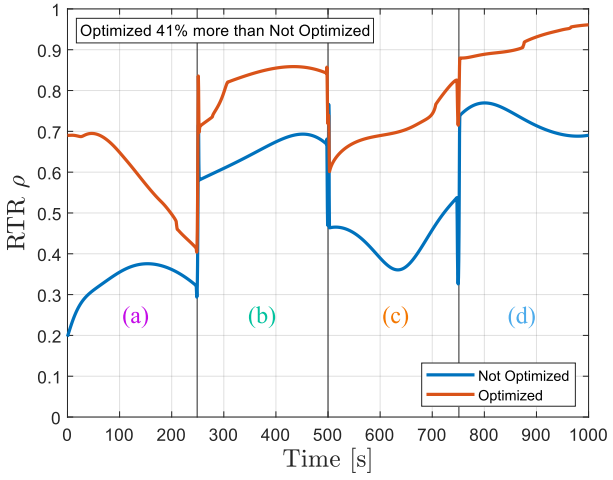
Figure 3.8: Robot in starting configuration on each trajectory for least value of ϵ in case of no optimization (blue) and highest value of ϵ with optimization (red). Values of η, ν and ρ in both cases are shown on the right of each sub-figure.



(a) Dexterity metric η



(b) Bounded manipulability metric ν



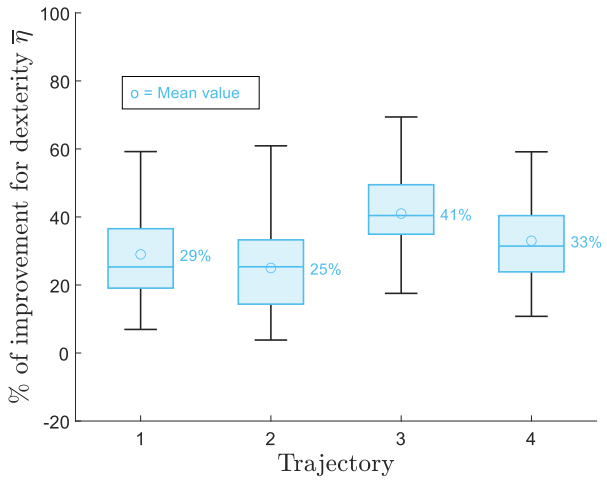
(c) RTR metric ρ

Figure 3.9: Graph of η, ν and ρ while following the first trajectory for least value of $\bar{\epsilon}$ in case of no optimization (blue) and highest value of $\bar{\epsilon}$ with optimization (red). Each sector is labeled (a), (b), (c) and (d) to match the corresponding part of the trajectory.

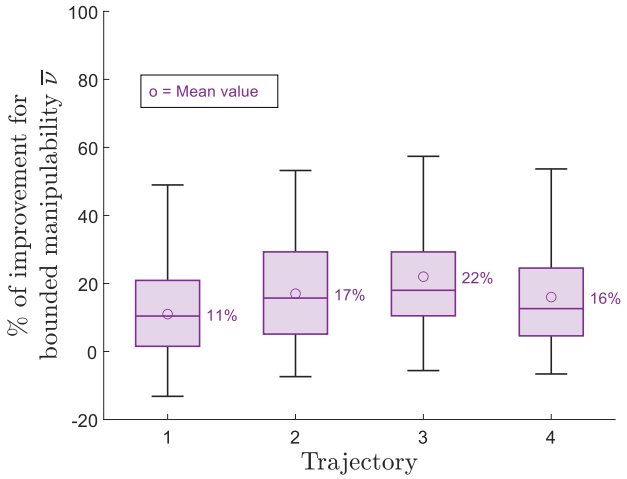
lower than in the non-optimized cases. This leads to a smaller angle between the joint velocity and torque vectors and increases the RTR value. Moreover, the vectors from the joint frames to the end-effector frame are different in length and orientation, giving distinct contributions to the kinematic Jacobian matrix and increasing the dexterity value.

Figure 3.9 shows the robot dexterity, bounded manipulability and RTR profiles along the first trajectory for the minimum value of $\bar{\epsilon}$ in case of no optimization and the maximum $\bar{\epsilon}$ with optimization tasks. The dexterity, bounded manipulability and RTR graphs are divided in the four trajectory sectors (a), (b), (c) and (d). The curves are higher when their tasks are used. The graphs also show the percentage of optimization for each curve. The activation of the dexterity task can improve the performance of almost a 90%. It can also be noticed how the RTR curve has discontinuities in correspondence to the trajectory corners since its value is directly affected by the orientation of the velocity and force vectors applied to the ending tool. Here, only the graphs for the first trajectory are shown since all the other trajectories showed similar behaviors. Comparing the dexterity, bounded manipulability and RTR results on each trajectory with and without optimization repeated a hundred times shows that the optimization tasks averagely increase the dexterity of 32%, the bounded manipulability of 17% and the RTR of 21%. Figure 3.10 shows the improvement percentage of $\overline{\eta}$, $\overline{\nu}$ and $\overline{\rho}$ with and without the optimization tasks on all trajectories and for a hundred repetitions. These percentages demonstrate how the use of optimization tasks generally improves the robot performance. In few cases, the negative percentage issue is met, which appears only for the bounded manipulability and the RTR.

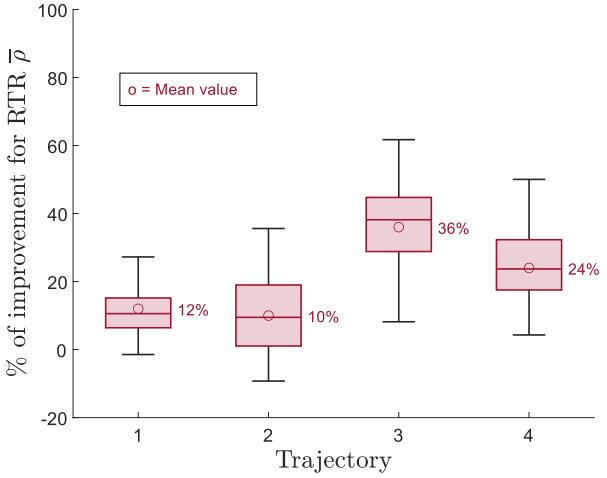
Another consideration that can be pointed out is the relation between the RTR value and the velocity and force vectors orientation. The RTR results shown in Fig. 3.10c are better in (b)-(d) than in (a)-(c) sectors. Taking into account the first and second horizontal trajectories in case of no optimization, the RTR values are in general 23% higher when the robot moves along axis x, (b)-(d) sectors, than along axis y, (a)-(c) sectors. When the RTR task is used, its performance difference between (b)-(d) and (a)-(c) sectors becomes 16% because the task helps maintaining higher values in all the sectors. Then, considering the third and fourth vertical trajectories in case of no optimization, the RTR values are 24% higher when the robot moves along axis x, (b)-(d) sectors, than along axis z, (a)-(c) sectors. When the RTR task is used, its performance difference between (b)-(d) and (a)-(c) sectors becomes 3%. So, the robot has higher RTR performance when the end-effector does



(a) Optimization percentages for mean value $\overline{\eta}$



(b) Optimization percentages for mean value $\overline{\nu}$

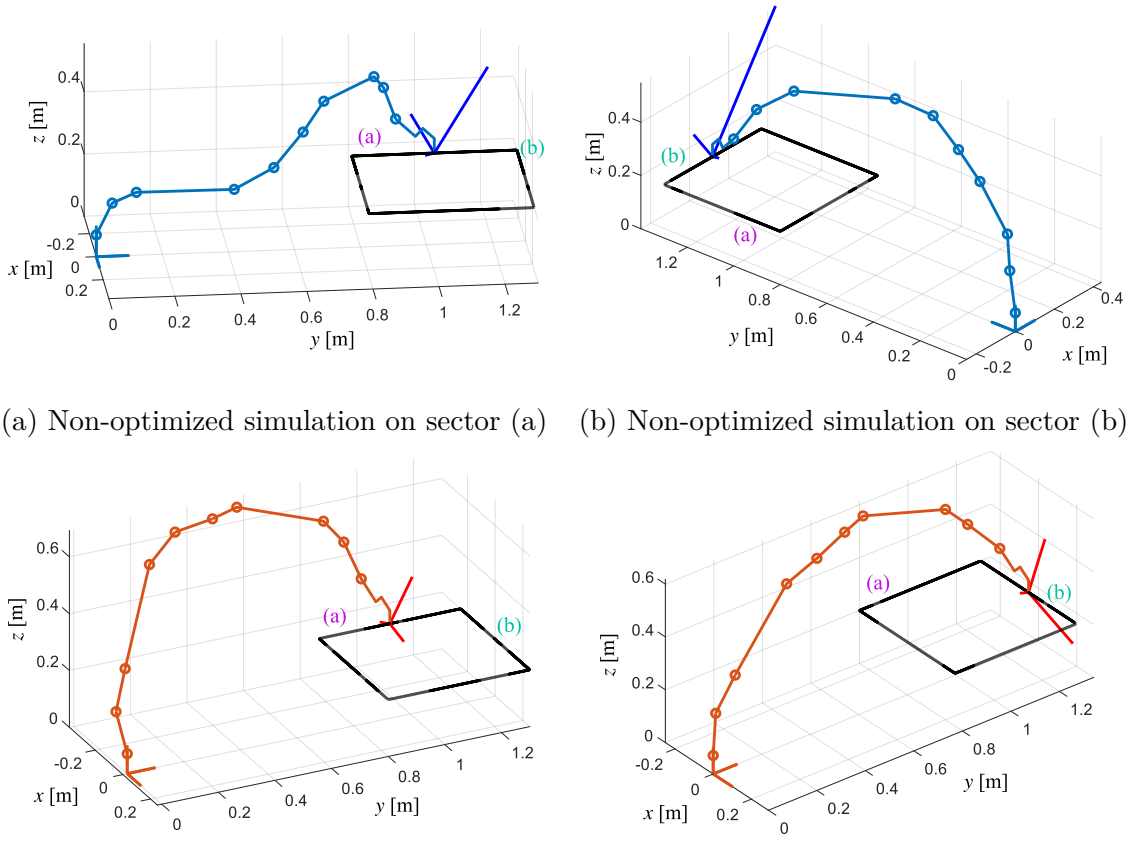


(c) Optimization percentages for mean value $\overline{\rho}$

Figure 3.10: Percentages of optimization for mean value of all the optimization metrics on each trajectory comparing the optimized and non-optimized case for each of the hundred repetitions. The circle ∘ highlights the mean percentage of the hundred repetitions.

a tangential horizontal movement than radial and vertical movements. A similar behavior can not be noticed in the dexterity or bounded manipulability because it is not affected by the magnitude or orientation of the velocity and force vectors applied to the end-effector.

To further investigate the RTR behavior, it is useful to analyze the robot linear kinematic Jacobian matrix singular vectors applied to the end-effector. In this case, only the linear kinematic Jacobian matrix is considered since no end-effector twist or wrench is planned to be applied to the end-effector while tracking the trajectories. By definition, the RTR index is related to the angle between the joint velocity $\dot{\mathbf{q}}$ and torque $\boldsymbol{\tau}$ vectors and, as consequence, the angle between the end-effector twist $\mathbf{t}$ and the wrench $\mathbf{w}$, identifying the effectiveness in producing a desired motion. This concept is also highlighted by the robot linear kinematic Jacobian matrix singular vectors. A singular vector closer to zero means that the robot can no longer perform



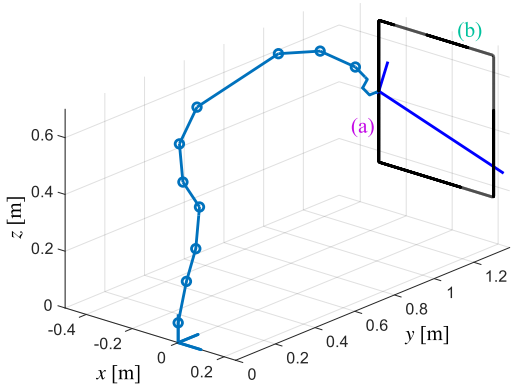
(a) Non-optimized simulation on sector (a)

(b) Non-optimized simulation on sector (b)

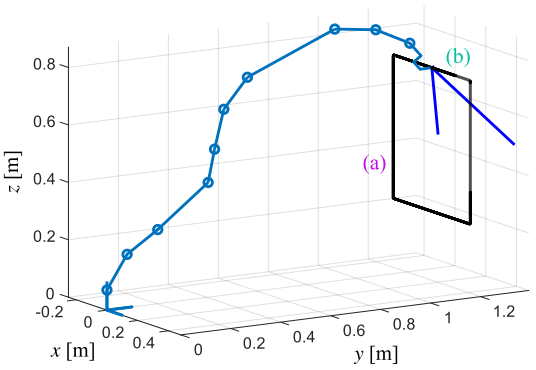
(c) Optimized simulation on sector (a)

(d) Optimized simulation on sector (b)

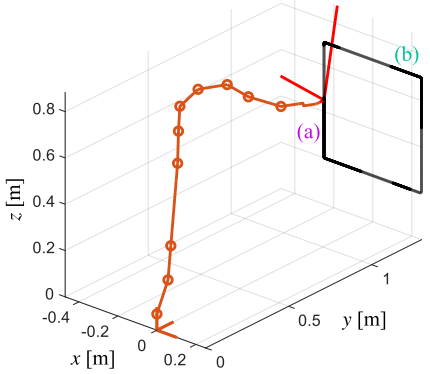
Figure 3.11: Robot configurations and linear kinematic Jacobian matrix singular vectors applied to the end-effector along first trajectory on sectors (a) and (b) for least value of $\bar{\epsilon}$ in case of no optimization (blue) and highest value of $\bar{\epsilon}$ with optimization (red)



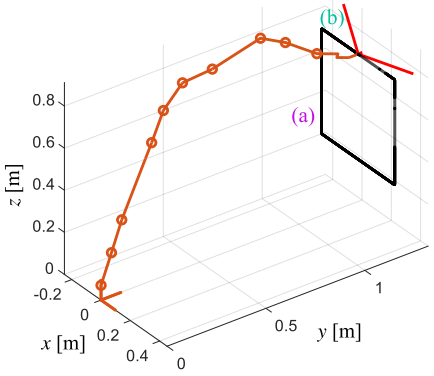
(a) Non-optimized simulation on sector (a)



(b) Non-optimized simulation on sector (b)



(c) Optimized simulation on sector (a)



(d) Optimized simulation on sector (b)

Figure 3.12: Robot configurations and linear kinematic Jacobian matrix singular vectors applied to the end-effector along fourth trajectory on sectors (a) and (b) for least value of $\bar{\epsilon}$ in case of no optimization (blue) and highest value of $\bar{\epsilon}$ with optimization (red)

a movement along the direction identified by the singular vector. On the contrary, a bigger singular vector implies that the robot can freely move along that direction. Figure 3.11 shows the robot configurations and linear kinematic Jacobian matrix singular vectors applied to the end-effector along the first trajectory. The blue robot is related to the least $\bar{\epsilon}$ obtained without optimization and the red ones to the highest $\bar{\epsilon}$ with optimization. Figures 3.11a and 3.11b present the robot and the linear singular vectors for the non-optimized simulation on the sectors (a) and (b), respectively. Figures 3.11c and 3.11d present the robot and the linear singular vectors for the optimized simulation on the sectors (a) and (b), respectively. It can be noticed that one singular vector is always close to zero, the one oriented along axis y. On the contrary, the other two singular vectors are longer. This justifies why the RTR performance are higher when moving along axis x, (b)-(d) sectors, than along axis y, (a)-(c) sectors. Moreover, this vector length difference is high in both

the optimized and not optimized cases. This clarifies why the RTR performance are still a 16% higher along axis x than along axis y even if the RTR optimization task is employed while following the horizontal trajectories. Then, Fig. 3.12 shows the robot configurations and linear kinematic Jacobian matrix singular vectors applied to the end-effector along the fourth trajectory. The blue robot is related to the least $\bar{\epsilon}$ obtained without optimization and the red ones to the highest $\bar{\epsilon}$ with optimization. Figures 3.12a and 3.12b present the robot and the linear singular vectors for the non-optimized simulation on the sectors (a) and (b), respectively. Figures 3.12c and 3.12d present the robot and the linear singular vectors for the optimized simulation on the sectors (a) and (b), respectively. In this case, the singular vector oriented along axis y is still close to zero. However, the robot does not have to move along y. In the non-optimized case, the singular vectors when moving along axis x and z clearly show that the robot can apply a more effective movement along axis x. However, this difference decreases in the optimized case dropping the improvement percentage between axis x and z movement from 24% to 3%. It happens because the RTR task helps maintaining higher values in all the trajectory sectors and the performance drop is lower from one sector to the other.

3.6 Task priority based kinematic control conclusions

This chapter described a kinematic control algorithm called Task Priority Inverse Kinematic (TPIK) used to kinematically control the redundant robots, proposed in [SC16, SCWA18, SCWA19]. This kinematic control algorithm is tested on the NB-R1 to exploit its kinematic redundancy to solve simultaneous tasks. Three new tasks are introduced to improve the robot kinetostatic performance. One is based on the dexterity, the second one on the manipulability and the last one on the robot transmission ratio (RTR). The NB-R1 robot tracks a series of trajectories thanks to the TPIK algorithm with and without the kinetostatic optimization tasks. The major limitation of this algorithm is its attraction by local maxima. So, the process needs to be run several times to come up with the best robot configurations for the desired set of tasks and optimization metrics. When the optimization tasks are used, the results clearly show an improvement in the robot performance, both dexterity, manipulability and RTR, without affecting the time consumption. In fact, the average time consumed by the algorithm at each step with or without the optimization tasks is almost equal, the difference is less than 50 μs. To rate

the improvement given by the optimization tasks, a linear combination of dexterity, bounded manipulability and RTR is used, named ϵ. When the robot reaches the trajectory starting pose using the optimization tasks, ϵ is averagely 50% higher than the non-optimized case. Along each trajectory, the mean value $\bar{\epsilon}$ is averagely 22% higher in the optimized case compared to the non-optimized one. Moreover, it can be noticed that the kinetostatic performance are affected by the trajectory placement and by the velocity and force vectors orientation. Finally, from the linear kinematic Jacobian matrix singular vectors study it can be deduced that the singular vector along axis y is always closer to zero, reducing the allowed movements along that direction. This reduces the movement effectiveness along the direction y. Part of the work presented in this chapter was published in [GCSL23a].

Chapter 4

Task Priority Based Kinematic Redundant Robot Design Optimization

This chapter proposes a design optimization process for kinematic redundant manipulators. The design optimization process is developed for kinematic redundant robots and employs the TPIK algorithm presented in Chapter 3 [SC16]. The chapter is organized as follow. Section 4.1 describes the existent methods for design optimization of redundant robot and the problems that can arise. Section 4.2 explains the new method for design optimization of robotic manipulators, which is the central topic. Section 4.3 tests the proposed optimization algorithm on the 21 degrees of freedom NB-R1 robot, presented in Chapter 1 and shown in Fig. 1.7 on page 11. Section 4.4 discusses and compares the obtained results. The conclusions about this new design optimization process based on the desired application are presented in Section 4.5.

4.1 Robot design optimization problem

When building a new kinematic redundant robot, an important step is the optimization of its design with respect to relevant performance indices. Angeles, in [Ang92], proposed an approach to design isotropic redundant manipulators by minimizing the condition number of the robot kinematic Jacobian matrix. However, an overall way to optimize redundant robots is to consider global indices. In [KSR+14], the authors optimized a redundant serial manipulator using the global conditioning index. The design can also be optimized through both kinematic and dynamic global indices to obtain a robot with a high global dexterity over its whole

workspace, ensuring high dynamic performance and energy efficiency [HKC+17].

It is important to consider the main robot tasks during the robot design optimization process. An example limited to non-redundant manipulators is presented in [PK93]. In [KK93], the authors worked with re-configurable modular manipulator systems to address the problem of task-based robot design. The work presented in [WC18] explored a method to optimize the design parameters and the desired trajectory together, considering the desired task with interesting results. A critical issue in these task-oriented robot optimization processes is the complexity caused by the problem non-linearity. There are many ways to reduce the complexity, for example, breaking down the problem into multiple easier steps [KK93]. In [CGSBK18], the authors adopted a particle swarm optimization algorithm to determine the design parameter values of two collaborative robots. Other solutions employed a grid method [PCY03] or a complex direct search method [ADJM13]. However, none of these techniques benefits from the robot redundancy to perform an optimization as a function of multiple tasks.

In [MDA22], the authors developed a novel method for optimizing manipulator design using kinematic redundancy resolution. The authors use a combination of kinematic redundancy resolution and a multi-objective optimization algorithm. This type of design optimization algorithm involves replacing some design parameters with additional degrees of freedom to optimize in the robot architecture. This allows performing the same task with different robot configurations. Through the Jacobian null-space projection, the optimization algorithm modifies the chosen design parameters together with the robot configuration, solving the main task and some additional performance optimization sub-tasks. This algorithm gave promising results being able to identify some optimal parameter values. However, the algorithm was tested only on a two degrees of freedom non-redundant robot adding other two degrees of freedom for redundancy resolution optimization. Moreover, the proposed process does not present any mechanisms to avoid the kinematic and algorithmic singularities that can arise. Here, the main contribution is a new design optimization process for redundant manipulators with respect to the robot kinetostatic performance and all the tasks that will be executed at run-time. This goal is achieved by employing again the TPIK algorithm, described in Chapter 3, during both design and testing phases. As in [MDA22], virtual prismatic or revolute joints replace the design parameters to be optimized in the candidate generation phase. In this way, the kinematic control algorithm considers the design parameters as extra robot degrees of freedom and their value is optimized accordingly. The dexterity,

manipulability and RTR tasks, described in Chapter 3, are included to improve the robot kinetostatic behavior during the design candidate generation and selection phases. The TPIK algorithm identifies some optimal robot design and configuration simultaneously solving the desired tasks.

4.2 Proposed design optimization method

The idea proposed in this chapter is to optimize the design of kinematic redundant manipulators based on the desired application, exploiting the TPIK algorithm already used for the online control. The TPIK framework is specifically designed to optimally control highly redundant robot manipulators, solving simultaneous tasks. In this case, the design optimization process is developed with respect to a tracking trajectory application while optimizing the robot kinetostatic performance. The problem formulation and all the phases that compose the optimization method are presented. In the considered case study, the robot main application requires following a trajectory inside a defined workspace while maximizing its kinetostatic performance. The kinetostatic performance is measured using the dexterity, manipulability and RTR indices.

4.2.1 Problem formulation

The application considered for the design optimization is tracking a set of trajectories that delimits a specific workspace area. While following these trajectories, the robot kinetostatic performance needs to be maximized. So, two principal inputs are defined and given to the optimization algorithm. The first one is a series of p trajectories with desired orientations and velocities. These trajectories delimit the workspace area where the robot will be applied. The second input is the list of tasks that the robot should achieve while tracking the trajectories. The main tasks are related to the end-effector pose and velocity to correctly track the desired trajectories. A safety task is added for joint limit compliance. Three tasks based on dexterity, manipulability and RTR evolution are used to improve the robot kinetostatic performance. Both the dexterity and manipulability are kinematic indices. As mentioned in Chapter 3, maximizing them simultaneously forces the manipulability hyper-ellipsoid to be as big as possible and close to a hyper-sphere. So, ideally, the robot will be able to move with the same high velocity amplification factor in all directions while reducing actuator velocity limits. The RTR task tries to align

Table 4.1: Details about the task names, control objective types, and hierarchy levels. Symbol (E) represents the equality control objective tasks and (I) the inequality ones. The last two columns list the task hierarchies for actions $\mathscr{A}_1$ (Reach Pose) and $\mathscr{A}_2$ (Follow Trajectory). The symbol "/" means that a task is not present in an action.

			Hierarchy level	
Task	Category	Type	$\mathscr{A}_1$	$\mathscr{A}_2$
Joint Limit	system safety	I	1^{st}	1^{st}
End-Effector Pose	action oriented	E	2^{nd}	/
End-Effector Velocity	action oriented	E	/	2^{nd}
Dexterity	optimization	I	3^{rd}	3^{rd}
Manipulability	optimization	I	3^{rd}	3^{rd}
RTR	optimization	I	3^{rd}	3^{rd}

the joint velocity and torque vectors making the robot movement along the desired direction more effective.

Table 4.1 reports all the task information used during the optimization. The dexterity, manipulability and RTR tasks have the same priority level since they have the same relevance in the design optimization process. The optimization algorithm uses the linear combination $\epsilon(\eta, \nu, \rho)$ of the three kinetostatic performance indices to rate and compare the kinetostatic performance of the obtained designs. The computation of $\epsilon(\eta, \nu, \rho)$ was described at Eq. (3.23) on page 49. To recall its formula, ϵ is computed as a function of the dexterity η, bounded manipulability ν and RTR ρ:

$$\epsilon(\eta, \nu, \rho) = \lambda_1\eta + \lambda_2\nu + \lambda_3\rho, \tag{4.1}$$

where the bounded manipulability ν is the manipulability μ limited between $[0, 1]$,

$$\nu = 1 - \frac{1}{1+\mu}. \tag{4.2}$$

4.2.2 Preliminary phase

The optimization algorithm is divided into two main phases. The first is the candidate generation phase, where several design candidates are generated. The second phase is the candidate selection one, where the robot designs obtained from the optimization process are evaluated and compared. Before these phases, some preliminary steps are necessary. The robot design parameters $\boldsymbol{\zeta}$ to be optimized need to be selected, for example, the link length or the angular offset amplitude between two consecutive joints. Then, controllable virtual joints are inserted in the robot architecture to substitute these design parameters. A prismatic joint substitutes a link to control its length, while a revolute joint replaces an angular offset to modify its angle. So, the design parameter vector $\boldsymbol{\zeta}$ is converted into virtual joint vector $\mathbf{q}_{n_v}$. During the candidate generation phase, the joint variable vector contains both the real and virtual joint positions $\mathbf{q} = [\mathbf{q}_{n_r}, \mathbf{q}_{n_v}] \in \mathbb{R}^n$, with $\mathbf{q}_{n_r} \in \mathbb{R}^{n_r}$, $\mathbf{q}_{n_v} \in \mathbb{R}^{n_v}$ and the robot degrees of freedom is $n = n_r + n_v$. The joint limit task constrains the virtual joint values $\mathbf{q}_{n_v}$ between the desired limits, resulting in limiting the design parameter values $\boldsymbol{\zeta}$.

4.2.3 Candidate generation phase

This phase employs the robot with real and virtual joints $\mathbf{q} = [\mathbf{q}_{n_r}, \mathbf{q}_{n_v}]$. The robot configuration vector $\mathbf{q}$ is randomly initialized. From here, the robot is moved to track all the p trajectories in a random order to ensure more general results. The robot reaches the starting pose of the trajectories and tracks it entirely under the kinematic control of the TPIK algorithm. When the robot finishes tracking a trajectory, it is moved to another one until it follows all the p trajectories. At equidistant time steps t_i on each trajectory, the optimization algorithm saves the virtual joint values $\mathbf{q}_{n_v}$ into the design parameter variable $\boldsymbol{\zeta}^{t_i}$ and its kinetostatic performance value in ϵ^{t_i}. Once the robot has tracked all the p trajectories, the optimization algorithm makes the weighted average $\overline{\boldsymbol{\zeta}}$ of all the collected $\boldsymbol{\zeta}^{t_i}$ using their associated ϵ^{t_i} as weighting factor:

$$\overline{\boldsymbol{\zeta}} = \frac{\epsilon^{t_0}\boldsymbol{\zeta}^{t_0} + \cdots + \epsilon^{t_\mathrm{f}}\boldsymbol{\zeta}^{t_\mathrm{f}}}{\epsilon^{t_0} + \cdots + \epsilon^{t_\mathrm{f}}}, \tag{4.3}$$

where $[t_0, \ldots, t_\mathrm{f}]$ are the equidistant time steps in which the design parameter vector $\boldsymbol{\zeta}$ was saved. This weighted average is useful to give more importance to those design parameters that provided better kinetostatic performance. The average re-

Algorithm 4.1 Candidate generation phase

Require: Actions $\mathscr{A}_1$ = *Reach Pose* and $\mathscr{A}_2$ = *Follow Trajectory* and trajectory vector $\mathbf{p}$.

for $m_r := 1 \rightarrow$ number of repetitions **do**
  Random initialization of $\mathbf{q} = [\mathbf{q}_{n_r}, \mathbf{q}_{n_v}]$.
  Random shuffle of $\mathbf{p}$.
  for $k := 1 \rightarrow p$ **do**
    Reach starting pose of $\mathbf{p}(k)$ using $\mathscr{A}_1$.
    while Trajectory $\mathbf{p}(k)$ not finished **do**
      Move to next step on $\mathbf{p}(k)$ using $\mathscr{A}_2$.
      if $t_i =$ equidistant time step **then**
        Save $\mathbf{q}_{n_v}$ as $\boldsymbol{\zeta}^{t_i}$.
        Save ϵ^{t_i}.
      end if
    end while
  end for
  Compute weighted average $\overline{\boldsymbol{\zeta}}$ for all $[t_0, \ldots, t_\mathrm{f}]$.
end for

sult $\overline{\boldsymbol{\zeta}}$ is stored. The process described above is repeated several times. At the end of this phase, the optimization algorithm has collected a list of candidate designs described by $\overline{\boldsymbol{\zeta}}$. Algorithm 4.1 sums up all the steps of the candidate generation phase.

During this phase, the optimization algorithm computes and collects the values of the kinetostatic indices η, ν and ρ for the real robot architecture without any virtual joints. In fact, the additional revolute/prismatic joints alter the kinetostatic performance of the real robotic design. This performance alteration risks damaging the optimization process and obtaining not optimized designs. So, the columns corresponding to the virtual joints $\mathbf{q}_{n_v}$ in $\mathbf{J}_w$ and its derivative $\partial \mathbf{J}_w / \partial q_i$, $\forall q_i \in \mathbf{q}$ are set to zero to correctly compute the kinetostatic metrics and their derivatives

4.2.4 Candidate selection phase

The candidate selection phase tests the designs obtained from the candidate generation phase to identify the best one. Here, the virtual joints are removed from the robot architecture vector and replaced by constant links and offsets whose value was stored inside all the obtained $\overline{\boldsymbol{\zeta}}$. In this way, several new robotic designs are

Algorithm 4.2 Candidate selection phase

Require: Actions $\mathscr{A}_1$ = *Reach Pose* and $\mathscr{A}_2$ = *Follow Trajectory* and trajectory vector $\mathbf{p}$.

```
for m_d := 1 → number of optimized designs do
    for m_r := 1 → number of repetitions do
        Random initialization of q = [q_{n_r}].
        for k := 1 → p do
            Reach starting pose of p(k) using A_1.
            while Trajectory p(k) not finished do
                Move to next step on p(k) using A_2.
                Save ε^{t_i} at the time step t_i.
            end while
        end for
        Compute ε̂ = 0.5 ε_min + 0.25 (ε̄ + ε_max).
    end for
end for
Select the best design as max(ε̂).
```

generated. From now on $\mathbf{q} = [\mathbf{q}_{n_r}]$ and $n_v = 0$, and the total amount of degrees of freedom $n = n_r$. For each optimized robotic design, the configuration vector $\mathbf{q}$ is randomly initialized. Then, each robot is moved to the starting pose of one trajectory and tracks it entirely under the kinematic control of the TPIK algorithm. When one robot finishes tracking a trajectory, it is moved to the next one until it has followed all the p trajectories. At each trajectory time step t_i, the optimization algorithm stores the value of ϵ^{t_i} for that robot configuration. This process is repeated several times to obtain more general results. When the robot has tracked all the trajectories, the optimization algorithm computes $\hat{\epsilon}$ as

$$\hat{\epsilon} \triangleq 0.5\ \epsilon_{\min} + 0.25\ (\overline{\epsilon} + \epsilon_{\max}), \tag{4.4}$$

where $\epsilon_{\min}$, $\overline{\epsilon}$ and $\epsilon_{\max}$ are respectively the minimum, mean and maximum of ϵ values along all the p trajectories. The $\hat{\epsilon}$ value is used to compare the designs and identify the best one. The process is repeated for all the designs obtained from the previous phase. The design that has the highest $\hat{\epsilon}$ is identified as the best one. Algorithm 4.2 sums up all the steps of the candidate selection phase.

4.3 Optimization test set up description

The design optimization process proposed here is tested on the NB-R1 robot. The following section describes the optimized robot design parameters and their features. Then, the trajectories employed for the optimization are described in terms of size and placement. Moreover, the velocities and forces required to follow them are listed. The machine and implementation details remained the same ones proposed in Chapter 3 and shown in Table 3.3 on page 48.

4.3.1 Robot under study

The employed robot for testing the design optimization algorithm is the NB-R1, shown in Fig. 1.7. In this case, the dimensions of the NB-module are constant and not included in the optimization. Their values are the same of the ones shown in Table 4.2. The optimized robot design parameters are the link lengths l_1 and l_2 and the amplitude of three angular offsets β_1, β_2 and β_3, shown in Fig. 4.1. The angular offsets are respectively inserted between the first link and first mechanism of the elbow, the second link and first mechanism of the wrist, and between the second-to-last and last mechanisms of the wrist. The angles β_1, β_2 and β_3 rotate about axis x, depicted in red in Fig. 4.1. An ending tool is mounted on the last revolute joint of the robot. The design parameters of the tool are inserted in the

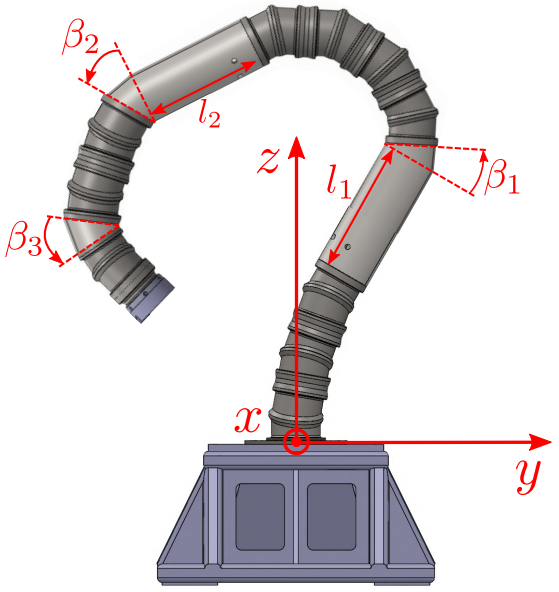


Figure 4.1: Robot design parameters under optimization: link lengths l_1 and l_2, angular offsets β_1, β_2 and β_3

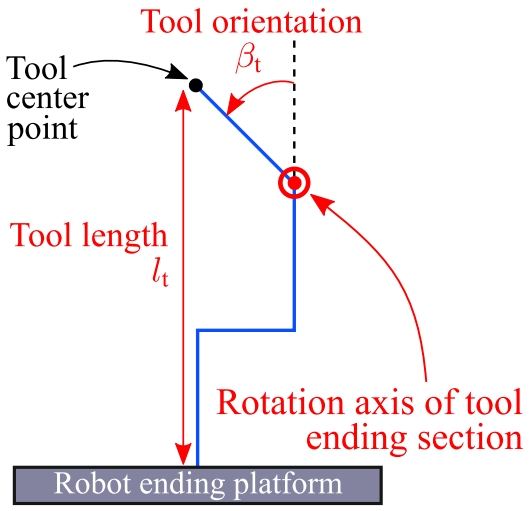


Figure 4.2: Ending tool design parameters under optimization: tool length l_t and orientation β_t

Table 4.2: NB-module dimensions plus real and virtual joint details

NB-module half height r	0.07 m
NB-module tube slope α	15°
Max/min revolute joint velocities	±1.0 rad/s
Max absolute revolute joint accelerations/decelerations	2.5 rad/s^2
Max/min prismatic joint velocities	±1.0 m/s
Max absolute prismatic joint accelerations/decelerations	2.5 m/s^2

optimized variables, namely its length l_t and orientation β_t, shown in Fig. 4.2. So, the number of virtual joints is $n_v = 7$ and the robot with virtual joints has 28 degrees of freedom, i.e. $n_r + n_v = 28$. Table 4.2 gives the main parameter values and limits of the robot.

The design parameters under optimization are limited to avoid too massive results with no sense. However, the limits are large enough trying to not over-constrain the optimization process. The link lengths l_1 and l_2 are constrained in the range $[0, 3]$ m. The angular offset amplitude β_1 is constrained in $[-3\pi/4, \pi/4]$ rad and β_2 and β_3 in $[-3\pi/4, 3\pi/4]$ rad. The tool length l_t is valid in the range $[0, 0.5]$ m, and the tool orientation offsets β_t in $[0, 3\pi/4]$ rad. These ranges are used by the joint limits task to constrain the virtual joints.

4.3.2 Employed trajectories

The robot application requires tracking horizontal and vertical trajectories in a cube whose sides are 0.7 m × 0.7 m long and centered in $(x, y, z) = (0.0, 0.75, 0.65)$ m. The orientation of the end-effector is expressed in terms of roll ϕ, pitch θ and yaw ψ. The angle values for the horizontal trajectories are $(\phi, \theta) = (\pi, 0),\ \forall\ \psi \in [-\pi, \pi]$, and for the vertical trajectories $(\phi, \theta) = (\pi/2, 0),\ \forall\ \psi \in [-\pi, \pi]$.

Figure 4.3 shows the $p = 4$ trajectories used during the candidate generation and selection phases of the optimization process. The two horizontal trajectories, green and magenta, have the corners placed in $x = (-0.35, 0.35)$ m and $y = (0.4, 1.1)$ m at the height of $z = 0.3$ m, green trajectory, and $z = 1.0$ m, magenta trajectory. The two vertical trajectories, blue and black, have the corners placed in $x = (-0.35, 0.35)$ m and $z = (0.3, 1.0)$ m at the depth of $y = 0.4$ m, blue trajectory, and $y = -1.1$ m, black trajectory. The small arrows along the trajectories express axis z orientation of the end-effector frame while following the trajectories. The trajectories are planned to cut a squared shape using a machining tool and the measures are shown in Fig. 4.4. Figure 4.4 also shows the orientation of the velocity vector $\vec{\mathbf{v}}$ and the tangential and radial force vectors $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ applied on the machining tool. The magnitudes of $\vec{\mathbf{v}}$, $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ are constant along the entire trajectory. The profiles of $\vec{\mathbf{v}}$, $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ are depicted in Fig. 4.5. The gravity force and the cutting one

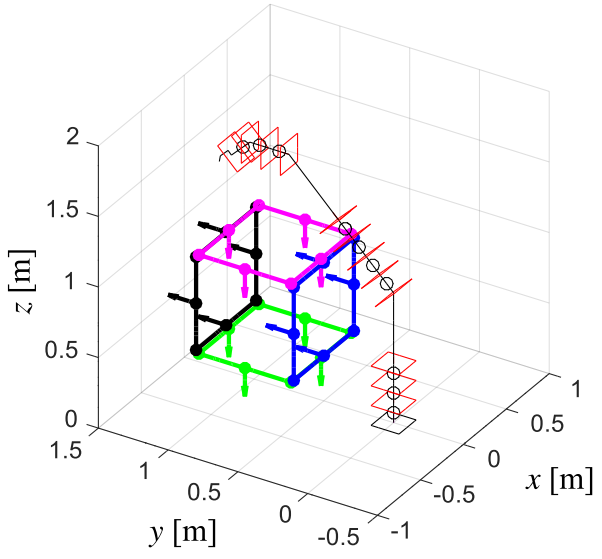


Figure 4.3: The four trajectories used in the optimization process and a version of Nimbl'Bot robot of size: $l_1 = 0.5$ m, $l_2 = 0.5$ m, $\beta_1 = \pi/4$ rad, $\beta_2 = \pi/4$ rad, $\beta_3 = \pi/4$ rad, $l_t = 0.1$ m and $\beta_t = \pi/4$ rad

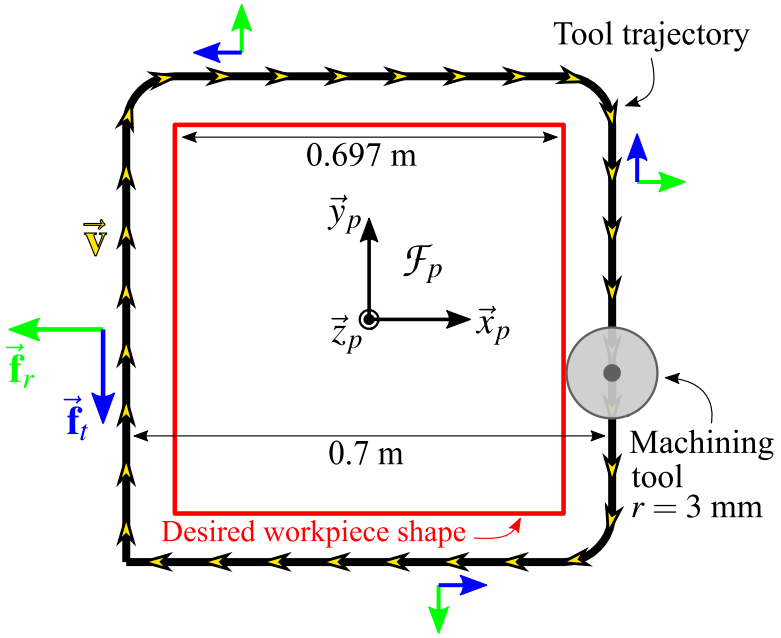
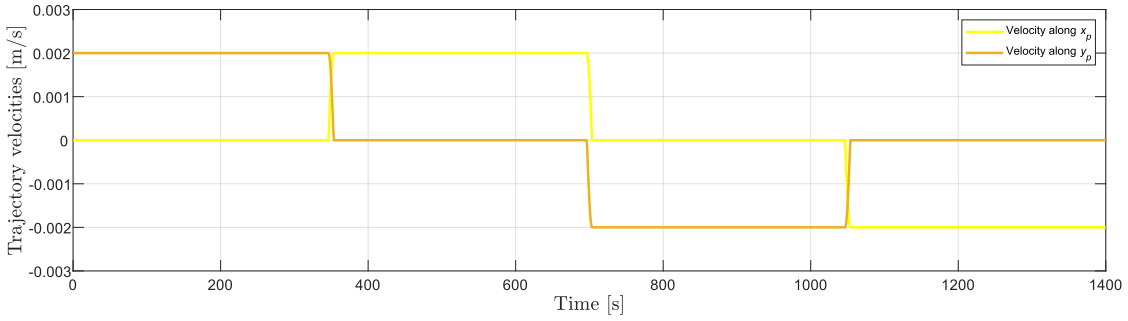
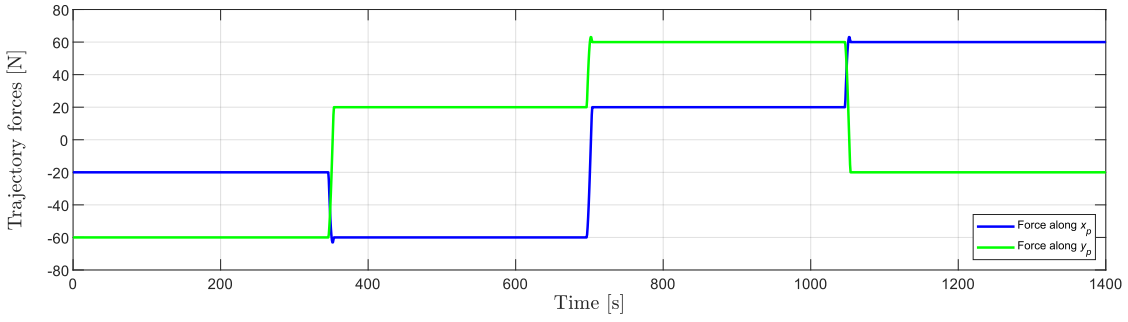


Figure 4.4: Measures of the desired workpiece, machining tool and trajectory tool with workpiece frame $\mathcal{F}_p$ in optimization test. Orientation of velocity vector $\vec{\mathbf{v}}$ (yellow) plus tangential and radial force vectors $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ (blue and green).



(a) Velocity profiles



(b) Cutting force profiles

Figure 4.5: Velocity and cutting force profiles applied to the machining tool in frame $\mathcal{F}_p$ during optimization tests

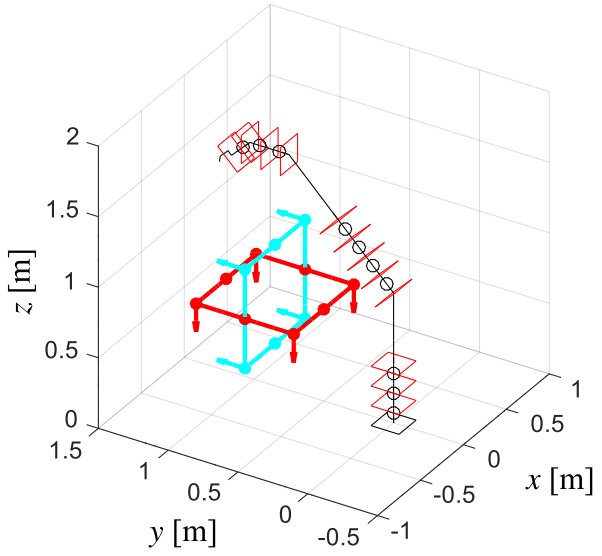


Figure 4.6: The two trajectories used to validate the designs obtained from the optimization process and a version of Nimbl'Bot robot of size: $l_1 = 0.5$ m, $l_2 = 0.5$ m, $\beta_1 = \pi/4$ rad, $\beta_2 = \pi/4$ rad, $\beta_3 = \pi/4$ rad, $l_t = 0.1$ m and $\beta_t = \pi/4$ rad

Table 4.3: Trajectory details in optimization test

Square side	0.7 m
Steps	560
Magnitude velocity vector $\|\|\vec{\mathbf{v}}\|\|_2$	0.002 m/s
Magnitude tangential force vector $\|\|\vec{\mathbf{f}}_t\|\|_2$	60 N
Magnitude radial force vector $\|\|\vec{\mathbf{f}}_r\|\|_2$	20 N
Time	1400 s

along $\vec{z}_p$ are neglected in this work. These four trajectories were chosen for the design optimization because they describe the cube where the robot is supposed to work in the real world. Then, the kinetostatic performance of the best and worst designs are compared on two new trajectories, one horizontal and one vertical, placed inside the cubic workspace area. These trajectories are used to confirm the optimization process ability to identify optimal kinetostatic designs for a desired application in a specific workspace area. Figure 4.6 shows the two trajectories used for testing the design obtained from the optimization process. The horizontal trajectory has the corners placed in $x = (-0.35, 0.35)$ m and $y = (0.4, 1.1)$ m at the height of $z = 0.65$ m, red trajectory. The vertical trajectory has the corners placed in $x = (-0.35, 0.35)$ m and $z = (0.3, 1.0)$ m at the depth of $y = 0.75$ m, cyan trajectory. Table 4.3 gives

the trajectory details.

4.4 Optimization results analysis

This section analyzes the results obtained from the performed optimization tests. First, the optimization method is tested only on two trajectories optimizing two design parameters. The results are compared with a discretized brute force optimization process to demonstrate the validity of the methodology proposed in this chapter. Then, the optimization method is tested on all the trajectories optimizing all the design parameters and the results are analyzed. Two tests are performed, constraining and not the resulting design size.

4.4.1 Computational time evaluation and comparison

Before optimizing all the design variables of Figs. 4.1 and 4.2, a brute force optimization process is run to have a baseline for the comparison with the proposed methodology. In this case, only the design variables l_2 and β_2 are optimized with respect to the green and blue trajectories shown in Fig. 4.3. The other design variables are set to $l_1 = 0.2$ m, $\beta_2 = \beta_3 = 0$, $l_t = 0.1$ m and $\beta_t = \pi/4$ rad. Several robot designs are generated combining the discretization of l_2 and β_2 over their allowed ranges. Namely, l_2 is bounded between $[0, 1]$ m with a step equal to 0.1 m, leading to 11 possible values. The parameter β_2 is bounded between $[-3\pi/4, 3\pi/4]$ rad with a step equal to 5°, leading to 55 values. Combining l_2 and β_2 generates 605 designs that are tested on the green and blue trajectories. The goal is to identify the design with the highest performance in terms of $\hat{\epsilon}$. Each design is tested ten times on each trajectory to obtain a general pool of results. If a design is not able to follow one or more trajectories, it is discarded.

Then, the design optimization process is run optimizing l_2 and β_2 as a function of the green and blue trajectories and the kinetostatic performance. The candidate generation phase is repeated four times and produces only 76 designs. So, the candidate selection phase takes less time in testing the 76 designs on the desired trajectories. Each design is tested ten times along each trajectory to obtain more general results. If one design is not able to follow one or more trajectories, it is discarded.

Table 4.4 shows the designs obtained from the two tests described above and their

Table 4.4: Design parameter and kinetostatic performance values $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ for the best designs obtained from discretization and optimization, respectively

Method	l_2	β_2	$\hat{\epsilon}$	$\hat{\eta}$	$\hat{\nu}$	$\hat{\rho}$	Time
Discretization	0.1 m	-105°	0.269	0.222	0.087	0.558	120 min
Optimization	0.06 m	-102°	0.281	0.221	0.086	0.595	20 min

kinetostatic performance $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$. The values of $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ are computed in the same way of Eq. (4.4) using the η, ν and ρ values collected at each step on each trajectory. The best designs identified by the two methods are highly similar. However, the optimization process reached a higher $\hat{\epsilon}$ because the identified values for l_2 and β_2 were not included in the discretized values. Moreover, the time required by the optimization process, 20 minutes, is lower than testing all the designs obtained from the discretization, more than 120 minutes. Therefore, the optimization method is six times faster in this case. In this case, only two design variables were considered. Testing all the possible combinations for seven design parameters, which correspond to more than two billion designs, would lead to more than four hundred million minutes of simulation.

4.4.2 Optimization results

Now, the optimization process is applied to all the design variables of Figs. 4.1 and 4.2. The optimization is performed with respect to the four desired trajectories of Fig. 4.3 and the kinetostatic performance indices, namely the dexterity η, bounded manipulability ν and RTR ρ. The candidate generation phase is run ten times generating 370 designs. Then, the candidate selection phase tests these designs on the four trajectories collecting ϵ, η, ν and ρ and the mean center of mass distance from the robot base $\overline{\text{CoM}}$ at each step. The mean center of mass distance $\overline{\text{CoM}}$ helps rating the reached robot size. In total, 307 designs passed the candidate selection phase. The others could not properly track one or more the trajectories and were discarded. During the candidate selection phase, each design follows each trajectory ten times starting from a random configuration to obtain a general pool of results.

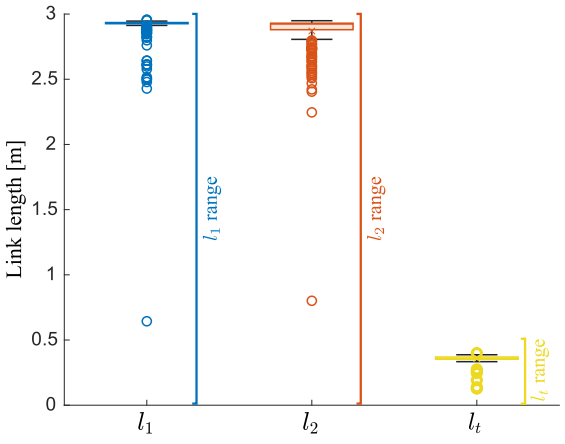
Table 4.5 reports the correlation coefficients between the kinetostatic performance indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$, the mean center of mass distance $\overline{\text{CoM}}$ and the optimized design parameters for all the solutions obtained from the optimization process. The higher the absolute value of the correlation coefficient, the darker the shade of blue

associated. A positive correlation coefficient indicates a directly proportional relation between two parameters. A negative correlation coefficient means an indirectly proportional relationship. The value of $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ is computed in the same way of Eq. (4.4) using the η, ν and ρ values collected at each step on each trajectory

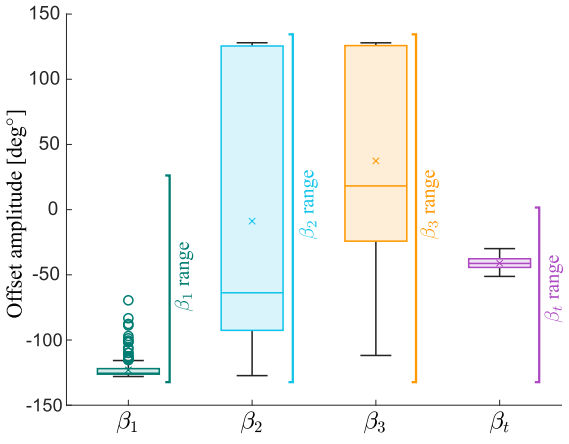
Table 4.5: Correlation coefficients between kinetostatic performance indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$, the mean center of mass distance from the robot base $\overline{\text{CoM}}$ and design parameters, bottom left, and their standard deviation along the diagonal, gray. The higher the correlation coefficient absolute value, the darker the blue shade. Correlation coefficients $\in [-1, 1]$.

	$\hat{\epsilon}$	$\hat{\eta}$	$\hat{\nu}$	$\hat{\rho}$	$\overline{\text{CoM}}$	l_1	l_2	β_1	β_2	β_3	l_t	β_t
$\hat{\epsilon}$	0.041											
$\hat{\eta}$	0.929	0.058										
$\hat{\nu}$	0.788	0.757	0.037									
$\hat{\rho}$	0.876	0.806	0.542	0.032								
$\overline{\text{CoM}}$	0.303	0.182	0.412	0.256	0.25 m							
l_1	0.164	0.096	0.481	-0.142	0.412	0.15 m						
l_2	-0.018	-0.066	0.327	-0.263	0.036	0.723	0.16 m					
β_1	-0.099	-0.049	-0.218	-0.012	-0.122	-0.142	-0.133	7°				
β_2	-0.346	-0.250	-0.352	-0.399	-0.818	-0.048	0.099	0.025	102°			
β_3	-0.239	-0.146	-0.231	-0.312	-0.834	-0.023	0.111	-0.012	0.804	73°		
l_t	0.279	0.285	0.136	0.289	0.05	0.051	-0.060	-0.024	0.041	-0.08	0.028 m	
β_t	-0.255	-0.324	-0.223	-0.26	0.25	0.131	0.047	0.065	-0.167	-0.229	-0.13	4°

Standard deviation



(a) Box plot for the links l_1, l_2 and l_t



(b) Box plot for the offset β_1, β_2, β_3 and β_t

Figure 4.7: Box plots of the values assumed by the design parameters for the selected candidates. Symbol $\times$ represents the mean and $\circ$ indicates the outliers. The design variable range is displayed next to each plot.

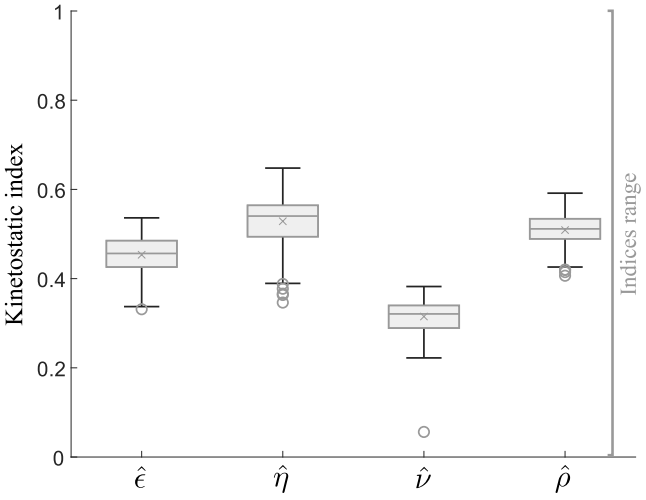


Figure 4.8: Box plot of the values assumed by the kinetostatic indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ for the selected candidates. Symbol $\times$ represents the mean and $\circ$ indicates the outliers. The kinetostatic index range is displayed next to the plots.

during the validation phase. The diagonal of Table 4.5 collects the standard deviation of the kinetostatic performance indices, mean center of mass distance and design parameters between all the solutions identified by the optimization process. Figures 4.7 and 4.8 show the box plots of all the values assumed by the design parameters for the selected candidates and the kinetostatic indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$, respectively. A first interesting aspect concerning the kinetostatic indices can be extracted from Table 4.5. The higher $\hat{\eta}$, the higher both $\hat{\nu}$ and $\hat{\rho}$. This means that the same design parameters influence them. The correlation coefficient between $\hat{\nu}$ and $\hat{\rho}$ is lower meaning that different design variables influence these two metrics. However, the correlation coefficients between $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ are always positive meaning that these performance indices are not antagonistic. Then, $\hat{\nu}$ results to have the stronger positive relationship with the length of l_1 and l_2. In fact, the longer the links, the higher $\hat{\nu}$. Moreover, a longer link l_1 tends to be associated with a longer l_2 and a farther $\overline{\text{CoM}}$. However, this positive relation is lower between l_2 and $\overline{\text{CoM}}$. Furthermore, $\overline{\text{CoM}}$ has a positive relationship $\hat{\nu}$. This is due to the fact that longer links lead to a higher $\hat{\nu}$ and a farther center of mass. Another consideration can be done on the correlation coefficients between the design parameters β_2, β_3 and β_t and the indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$, which are negative. This means that the lower β_2, β_3 and β_t the higher $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$. Moreover, a farther center of mass distance $\overline{\text{CoM}}$ is related to lower angles for β_2 and β_3. On the contrary, there is a relevant positive correlation between β_2 and β_3, meaning that a bigger β_2 is associated to a bigger β_3. Finally, higher kinetostatic indices are linked to longer l_t and lower β_t. However, these relations are feeble. The other design parameters do not present significant

correlation coefficients, but it is important to consider that this fact does not mean that any value can be used for these design parameters. Considering the design parameters, the standard deviation of l_t is much lower than l_1 and l_2. This means that almost all the obtained designs have a similar value for l_t. So, the optimization algorithm identified a small optimal range of values for l_t. This can also be noticed by comparing the interquartile size, i.e. the distance between the lower and upper quartile, of the box plots in Fig. 4.7a. However, the interquartile of l_1 and l_2 is also small meaning that almost all the links have a similar value. Then, β_1 and β_t have a standard deviation lower than β_2 and β_3 meaning that the optimization algorithm found a smaller optimal range of values for these parameters. Moreover, the β_2 and β_3 interquartile size in Fig. 4.7b is much bigger than the others. This can justify why the correlation coefficients between β_1 and β_t and $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ are low as well. Finally, almost all the values of l_1 and l_2 are close while β_2 and β_3 fall in a bigger range. This means that different combinations of β_2 and β_3 with similar l_1 and l_2 leads to the valid designs. Furthermore, the standard deviation of the kinetostatic indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ is small meaning that the obtained results are closer to each other. The plots in Fig. 4.8 show how each index is concentrated in a smaller range compared to the available one.

There is no design that can be identified globally as the best one, but several combinations can lead to a good kinetostatic design for the Nimbl'Bot robot. This is mainly due to the high kinetostatic redundancy of the robot that allows infinite possible solutions for the same problem. Moreover, more than one kinetostatic performance index is considered in the optimization process. So, the designs identified by the optimization process can perform better in dexterity, bounded manipulability or RTR. It is necessary to make a trade-off for choosing the best design. By the way, the values of l_1 and l_2 are almost always close to the upper limit of 3 m, which makes the designs unfeasible to be built in the real world. This test is performed to provide some general guidelines.

An analysis of the designs that performed the best and the worst in terms of $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ is accomplished. The right side of Table 4.6 shows the parameter values of the designs that performed the best and worst for each index, i.e. $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$, along the four trajectories used in the optimization process, shown in Fig. 4.3. In some cases, one design had the best or worst performance for more than one index. Figures 4.9 and 4.10 present the link lengths and offset amplitudes collected in Table 4.6 on graphs. The middle of Table 4.6 reports the values of $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ obtained during the candidate selection phase. The green and red colors highlight the highest and

Table 4.6: Design parameters and kinetostatic performance indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ for best and worst designs obtained during candidate selection phase. Green and red colors indicate the highest and lowest values for $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ obtained during the candidate selection and testing phases.

	l_1	l_2	β_1	β_2	β_3	l_t	β_t	Candidate selection				Testing			
								$\hat{\epsilon}$	$\hat{\eta}$	$\hat{\nu}$	$\hat{\rho}$	$\hat{\epsilon}$	$\hat{\eta}$	$\hat{\nu}$	$\hat{\rho}$
Best $\hat{\epsilon}$	2.94 m	2.79 m	-127°	-111°	8°	0.38 m	-47°	0.536	0.645	0.371	0.55	0.561	0.654	0.443	0.579
Best $\hat{\eta}$ $\hat{\nu}$	2.94 m	2.63 m	-120°	-89°	-26°	0.38 m	-43°	0.526	0.648	0.381	0.557	0.599	0.676	0.437	0.627
Best $\hat{\rho}$ and worst $\hat{\nu}$	0.64 m	0.8 m	-110°	-65°	38°	0.36 m	-37°	0.352	0.432	0.056	0.592	0.415	0.567	0.067	0.614
Worst $\hat{\epsilon}$ $\hat{\eta}$ $\hat{\nu}$	2.93 m	2.93 m	-119°	16°	32°	0.35 m	-37°	0.331	0.346	0.254	0.415	0.346	0.393	0.297	0.388
Worst $\hat{\rho}$	2.93 m	2.93 m	-100°	-62°	108°	0.27 m	-40°	0.337	0.363	0.234	0.407	0.402	0.371	0.238	0.472

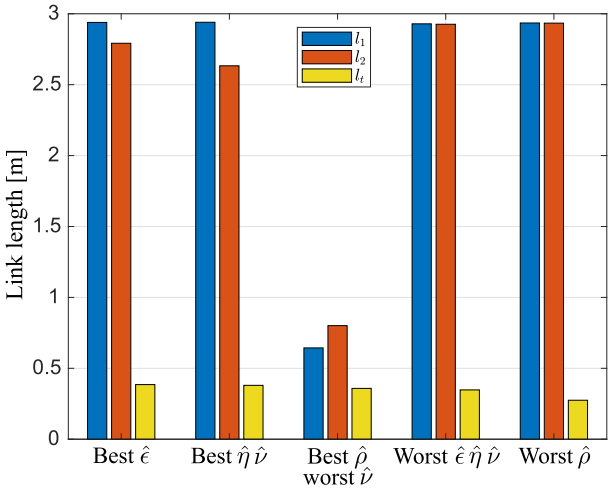


Figure 4.9: Link lengths of best and worst design for indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during the candidate selection phase

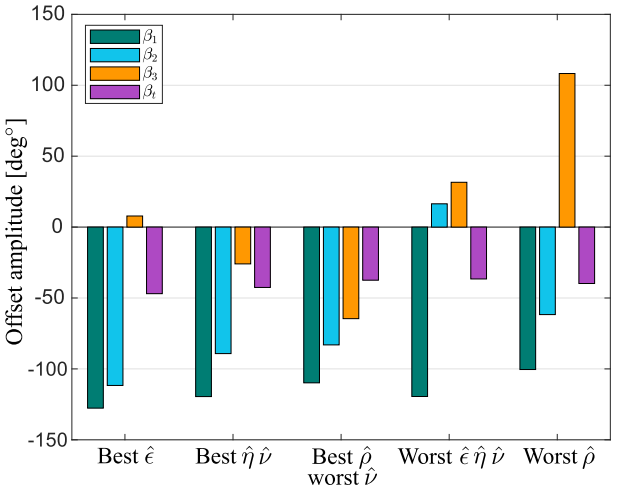
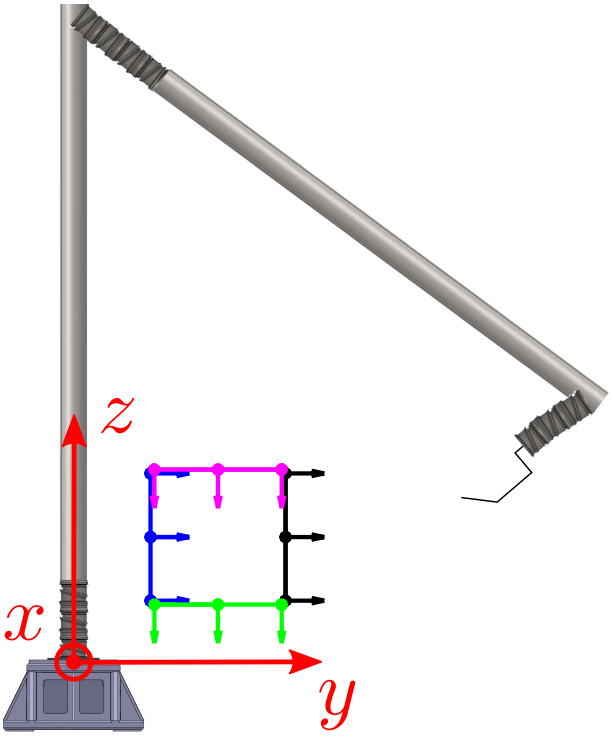
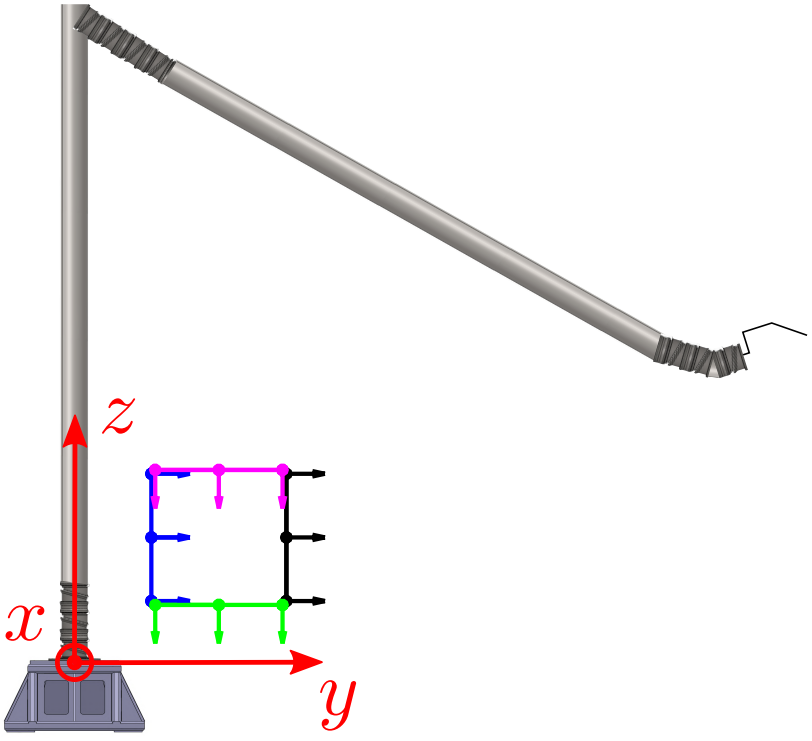


Figure 4.10: Offset amplitudes of best and worst design for indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during the candidate selection phase



(a) Best result for $\hat{\epsilon}$



(b) Worst result for $\hat{\epsilon}$

Figure 4.11: Representation of the design with the best and worst $\hat{\epsilon}$ together with the four trajectories used during the optimization process. The design parameter values are in the first line of Table 4.6.

lowest values for $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during the candidate selection phase. The value of $\hat{\epsilon}$, $\hat{\eta}$ and $\hat{\rho}$ are generally high for all the designs. This means that the optimization process succeed in generating only good designs. On the contrary, one design presents a really small $\hat{\nu}$ characterized by small links. This proves the positive relationship identified by Table 4.5 between the links length and the bounded manipulability. Figures 4.11 shows the designs that reached the best and worst $\hat{\epsilon}$ during the candidate selection phase.

The designs collected in Table 4.6 are tested on two trajectories inside the desired workspace area, shown in Fig. 4.6, which are different from the ones used during the candidate generation and selection phases. This testing phase is performed to show that the best and worst designs maintain similar performance in tracking trajectories inside the volume defined by the four ones used in the optimization process. The left side of Table 4.6 reports the values of $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ obtained during the testing phase. The green and red colors highlight the highest and lowest values for $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during the testing phase. The results show the same trend during the selection and testing phases meaning that each design has uniform performance in the desired workspace area. To conclude, this optimization process identified several possible optimal designs for a specific set of tasks in the desired workspace area. There is no unique global solution, but more than one design can reach high values for $\hat{\epsilon}$. For designs that share the same global index $\hat{\epsilon}$, choosing the final one is a trade-off between optimizing one of the three indices that contribute to $\hat{\epsilon}$.

4.4.3 Optimization results with center of mass task

Here, the optimization process is again applied to all the design variables of Figs. 4.1 and 4.2 but this time a new task is included. This task is based on the center of mass position with respect to the base. It constraints the robot dimensions, obtaining more compact designs. Table 4.7 shows the new task list with center of mass task details. The performed tests are the same presented in Section 4.4.2 except for the center of mass task. In this case, only 99 designs passed the candidate selection phase over the generated 370. In this case, these obtained designs have shorter links due to the center of mass task and fewer designs passed the candidate selection phase.

Table 4.8 reports the correlation coefficients between the kinetostatic performance indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$, the mean center of mass distance $\overline{\text{CoM}}$ and the optimized design parameters for all the solutions obtained from the optimization process. The higher

Table 4.7: Details about the task names, control objective types, and hierarchy levels. Symbol (E) represents the equality control objective tasks and (I) the inequality ones. The last two columns list the task hierarchies for actions $\mathscr{A}_1$ (Reach Pose) and $\mathscr{A}_2$ (Follow Trajectory). The symbol "/" means that a task is not present in an action.

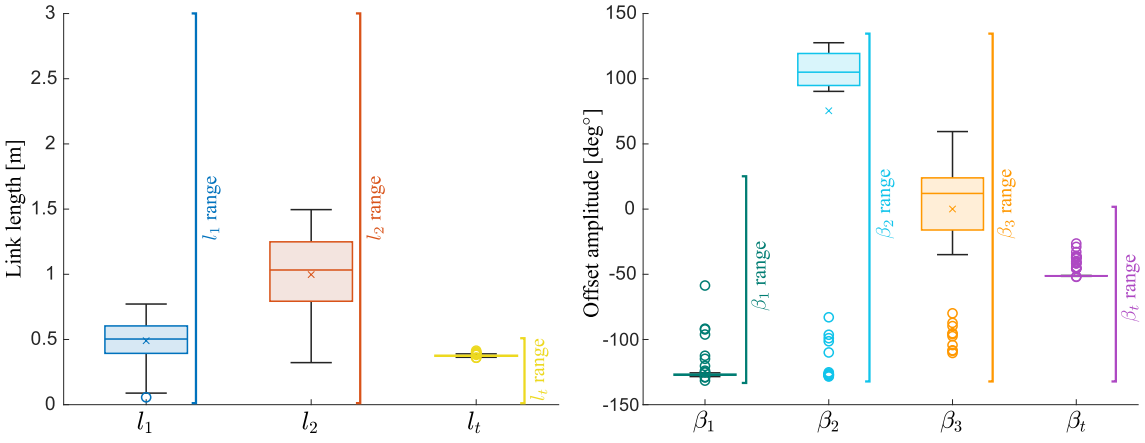
			Hierarchy level	
Task	Category	Type	$\mathscr{A}_1$	$\mathscr{A}_2$
Joint Limit	system safety	I	1^{st}	1^{st}
End-Effector Pose	action oriented	E	2^{nd}	/
End-Effector Velocity	action oriented	E	/	2^{nd}
Dexterity	optimization	I	3^{rd}	3^{rd}
Manipulability	optimization	I	3^{rd}	3^{rd}
RTR	optimization	I	3^{rd}	3^{rd}
Center of Mass Position	optimization	I	4^{th}	4^{th}

Table 4.8: Correlation coefficients between kinetostatic performance indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$, the mean center of mass distance from the robot base $\overline{\text{CoM}}$ and design parameters, bottom left, and their standard deviation along the diagonal, gray, obtained with the center of mass task. The higher the correlation coefficient absolute value, the darker the blue shade. Correlation coefficients $\in [-1, 1]$.

	$\hat{\epsilon}$	$\hat{\eta}$	$\hat{\nu}$	$\hat{\rho}$	$\overline{\text{CoM}}$	l_1	l_2	β_1	β_2	β_3	l_t	β_t
$\hat{\epsilon}$	0.02											
$\hat{\eta}$	0.768	0.021										
$\hat{\nu}$	0.759	0.791	0.009									
$\hat{\rho}$	0.849	0.516	0.502	0.021								
$\overline{\text{CoM}}$	-0.444	-0.442	-0.525	-0.180	0.11 m							
l_1	0.625	0.702	0.917	0.325	-0.484	0.15 m						
l_2	0.664	0.751	0.947	0.366	-0.539	0.891	0.27 m					
β_1	-0.181	-0.208	-0.208	0.010	0.384	-0.345	-0.275	9°				
β_2	0.223	0.204	0.126	0.039	-0.855	0.094	0.132	-0.364	82°			
β_3	0.192	0.195	0.025	0.120	-0.551	-0.030	-0.063	-0.089	0.704	39°		
l_t	0.125	0.167	-0.012	0.193	0.099	-0.105	-0.070	0.118	-0.049	0.228	0.01 m	
β_t	-0.138	-0.407	-0.371	0.131	0.557	-0.440	-0.550	0.348	-0.384	-0.203	0.139	6°

Standard deviation

the absolute value of the correlation coefficient, the darker the shade of blue associated. The diagonal of Table 4.8 collects the standard deviation of the kinetostatic performance indices, mean center of mass distance and design parameters between all the solutions identified by the optimization process. Figures 4.12 and 4.13 show the box plots of all the values assumed by the design parameters for the selected candidates and the kinetostatic indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$, respectively. Thanks to the center of mass task, the length of the links is highly reduced respect to the results obtained in Section 4.4.2, as shown in Fig. 4.12a. Comparing the correlation coefficients in Table 4.8 with the ones in Table 4.5, where the center of mass task is disabled, it can be noticed that several relationships are still valid. For example, the influence between the kinetostatic indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ is again positive. The coefficients between l_1 and l_2 and between β_2 and β_3, respectively, are positive and high as in the previous no center of mass task case. Now, the $\overline{\text{CoM}}$ has a negative relationship with all the kinetostatic indices contrarily to what appeared in Section 4.4.2. This is probably due to the new resulting design sizes that are all smaller thanks to the center of mass task use. Now, the links l_1 and l_2 has high positive correlation coefficients with the kinematic indices $\hat{\eta}$, $\hat{\nu}$. This happens because the l_1 and l_2 values fall in a bigger range due to the center of mass task, as show in Fig. 4.12a. So, a higher correlation between the links and kinetostatic indices can be identified. Finally, $\overline{\text{CoM}}$ has an unintuitive negative relationship with l_1 and l_2. This is probably due to the design parameter combinations found by the algorithm. However, $\overline{\text{CoM}}$ maintains a negative relationship with β_2 and β_3.



(a) Box plot for the links l_1, l_2 and l_t (b) Box plot for the offset β_1, β_2, β_3 and β_t

Figure 4.12: Box plots of the values assumed by the design parameters for the selected candidates obtained with the center of mass task. Symbol × represents the mean and ∘ indicates the outliers. The design variable range is displayed next to each plot.

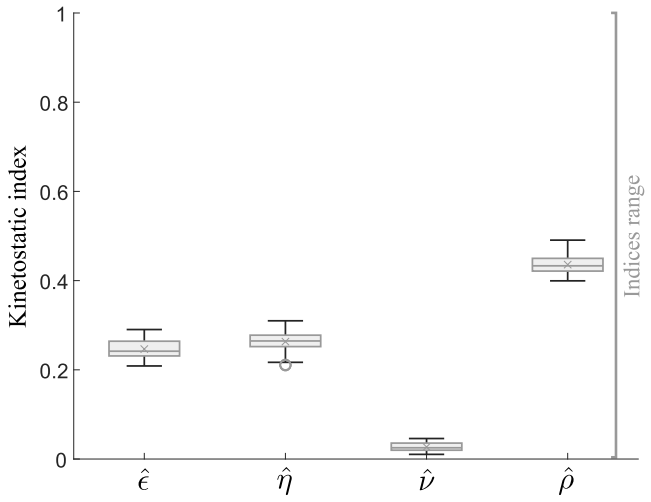


Figure 4.13: Box plot of the values assumed by the kinetostatic indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ for the selected candidates obtained with the center of mass task. Symbol $\times$ represents the mean and $\circ$ indicates the outliers. The kinetostatic index range is displayed next to the plots.

The standard deviation of the kinetostatic indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ is small meaning that the obtained results are closer to each other. The plots in Fig. 4.13 show how each index is concentrated in a smaller range compared to the available one. The index of $\hat{\nu}$ assumes lower values than the one in Fig. 4.13 because of the smaller link length due to the center of mass task use. The other standard deviation values show a similar behavior presented in Section 4.4.2. The β_2 and β_3 interquartile are smaller in this case than in Section 4.4.2 meaning that their values fall in a smaller range, as show in Fig. 4.12b. Doing these consideration it is necessary to recall that only 99 design passed the candidate selection phase with the center of mass task. So, less designs were used in the comparison.

Table 4.9: Design parameters and kinetostatic performance indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ for best and worst designs obtained during candidate selection phase with the center of mass task. Green and red colors indicate the highest and lowest values for $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ obtained during the candidate selection and testing phases.

	l_1	l_2	β_1	β_2	β_3	l_t	β_t	Candidate selection				Testing			
								$\hat{\epsilon}$	$\hat{\eta}$	$\hat{\nu}$	$\hat{\rho}$	$\hat{\epsilon}$	$\hat{\eta}$	$\hat{\nu}$	$\hat{\rho}$
Best $\hat{\epsilon}$	0.7 m	1.41 m	-126°	103°	-18°	0.37 m	-51°	0.29	0.289	0.041	0.482	0.262	0.317	0.04	0.446
Best $\hat{\eta}$	0.61 m	1.27 m	-123°	105°	-6°	0.37 m	-51°	0.277	0.31	0.038	0.462	0.297	0.344	0.037	0.468
Best $\hat{\nu}$	0.76 m	1.49 m	-126°	95°	-13°	0.37 m	-51°	0.244	0.277	0.046	0.435	0.271	0.283	0.045	0.453
Best $\hat{\rho}$	0.55 m	1.11 m	-125°	-127°	-104°	0.37 m	-42°	0.269	0.271	0.031	0.491	0.312	0.359	0.037	0.524
Worst $\hat{\epsilon}$ $\hat{\eta}$ $\hat{\nu}$	0.27 m	0.32 m	-120°	-83°	27°	0.37 m	-26°	0.209	0.211	0.01	0.414	0.23	0.188	0.01	0.462
Worst $\hat{\rho}$	0.65 m	1.36 m	-127°	127°	-95°	0.38 m	-51°	0.23	0.266	0.034	0.399	0.284	0.321	0.031	0.465

As before, an analysis of the designs that performed the best and the worst in terms of $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ is performed. The right side of Table 4.9 shows the parameter values of the designs that performed the best and worst for each index, i.e. $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$, along the four trajectories used by the optimization process, shown in Fig. 4.3. In some cases, one design had the best or worst performance for more than one index. Figures 4.14 and 4.15 present the link lengths and offset amplitudes collected in Table 4.9 on graphs. The middle of Table 4.9 reports the values of $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ obtained during the candidate selection phase. The green and red colors highlight the highest and lowest values for $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during the candidate selection phase. The values of $\hat{\epsilon}$, $\hat{\eta}$ and $\hat{\rho}$ are generally high for all the designs. This means that the optimization process succeed in generating only good designs. Moreover, $\hat{\rho}$ is similar to the one presented in Table 4.6. On the contrary, $\hat{\epsilon}$ and $\hat{\eta}$ decreased. This means that $\hat{\epsilon}$ and $\hat{\eta}$ are more affected by the size of the robot than $\hat{\rho}$. This could also be extracted by the coefficients of Table 4.8. Finally, $\hat{\nu}$ is the one that shows the worst performance due to the smaller links, as already noticed for Fig. 4.13. Figure 4.16 shows the designs that reached the best and worst $\hat{\epsilon}$. In this case, the two robots collide in the trajectories while tracking them. This happens because no obstacle avoidance task is employed during the optimization phases.

Finally, the designs collected in Table 4.9 are tested on two trajectories inside the desired workspace area, shown in Fig. 4.6, which are different from the ones used during the candidate generation and selection phases. This testing phase is performed to show that the best and worst designs maintain similar performance in tracking trajectories inside the volume defined by the four ones used in the op-

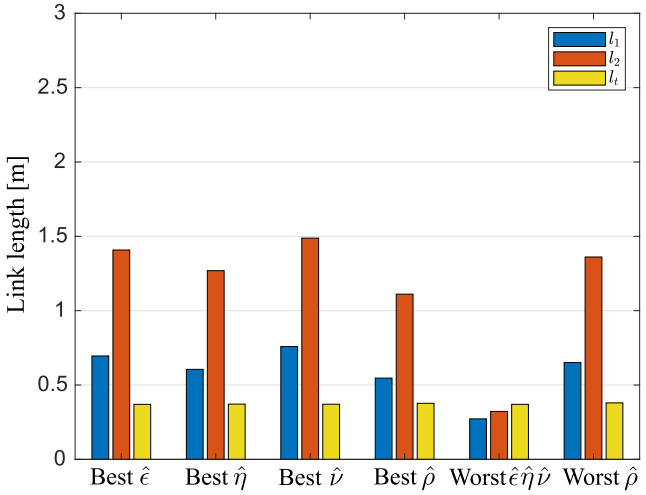


Figure 4.14: Link lengths of best and worst design for indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during the candidate selection phase obtained with the center of mass task

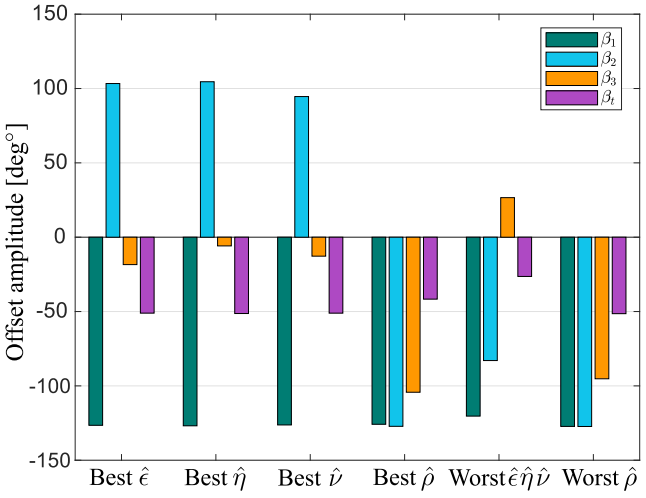
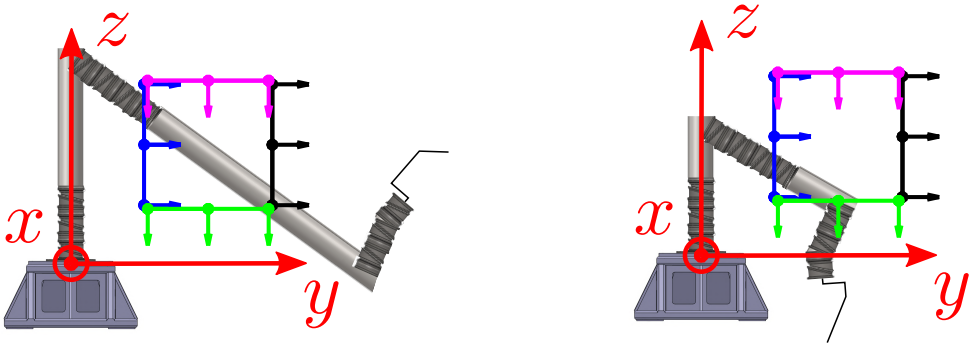


Figure 4.15: Offset amplitudes of best and worst design for indices $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during the candidate selection phase obtained with the center of mass task



(a) Best result for $\hat{\epsilon}$

(b) Worst result for $\hat{\epsilon}$

Figure 4.16: Representation of the design with the best and worst $\hat{\epsilon}$ together with the four trajectories used during the optimization process, employing the center of mass task. The design parameter values are in the first line of Table 4.9.

timization process. The left side of Table 4.9 reports the values of $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ obtained during the testing phase. The green and red colors highlight the highest and lowest values for $\hat{\epsilon}$, $\hat{\eta}$, $\hat{\nu}$ and $\hat{\rho}$ during the testing phase. The results show the same trend during the selection and testing phases meaning that each design has uniform performance in the desired workspace area. Thanks to the center of mass task, the optimization algorithm generated smaller links, which makes the robot being feasible in reality. However, the kinetostatic performance of this second set of designs were worse. Again, it does not exist a global optimal solution, but it is necessary to make a trade-off.

4.5 Design optimization conclusions

This chapter proposed and analyzed a new method for designing and optimizing kinematic redundant manipulators as a function of their application and kinetostatic performance. The process employs a task priority kinematic control algorithm to solve several tasks simultaneously exploiting the robot high kinematic redundancy. First, the process was tested on a simple case with two design variables and two trajectories. The obtained best design was compared with the one reached from testing some discretized combinations of the two design variables. The best designs in terms of the global kinetostatic performance index $\hat{\epsilon}$ obtained from the two tests are close and the optimization process converged toward a better design. This means that the optimization process can identify the best solution for a specific problem without testing all possible combinations of design parameter values and reduce the computational time.

Then, this new method worked on more design parameters identifying several possible Nimbl'Bot manipulators with high kinetostatic performance for the desired workspace area. Firstly, the algorithm was tested without any task related to the robot dimensions. The resulting designs had high kinetostatic performance but the links were too long to realize in the real world. After, the test was performed again with the center of mass task. In this case, the links were smaller generating more realizable designs. However, the kinetostatic performance were lower than in the case with no center of mass task. The optimization algorithm also pointed out some general guidelines for building a kinetostatic efficient Nimbl'Bot robots. Part of the work presented in this chapter was published in [GSCL23].

Chapter 5

Kinematic Redundant Robot Workspace Determination

This chapter presents a newly proposed workspace determination process developed specifically for kinematic redundant manipulators. The presented method employs the already described TPIK algorithm. It is outlined as follows. Section 5.2 describes the workspace determination method proposed in this chapter. Section 5.3 presents the three Nimbl'Bot robots employed to test the new method and some preliminary steps. Section 5.4 analyzes the obtained results testing the proposed workspace determination algorithm and compares the performance with other methods. Section 5.5 proposes an interpolation algorithm to plot the workspace surface. Section 5.6 highlights and discusses some limitations of the proposed approach. Conclusions and future work are given in section 5.7.

5.1 Workspace determination problem

The workspace of a manipulator is the set of positions and orientations reachable by the robot end-effector [DK04]. Planning the robot end-effector movements and trajectories requires careful consideration of the workspace analysis. As a result, the workspace of conventional robots has been the subject of numerous scientific studies throughout the past three decades. The problem becomes more complex in the case of kinematic redundant robots [DOXY20]. Three main types of methods exist for the workspace determination of kinematic redundant robots, as described in [DOXY20]. The first one is the geometric type, mainly applied for the workspace identification of kinematic redundant planar robots [YMM11]. This approach is intuitive but can not accurately describe the workspace of spatial, i.e., non-planar, robots. The second method is the analytic one that employs the kinematic Jacobian

matrix [BMR12]. The workspace boundaries are generated by searching for the rank deficiency of the kinematic Jacobian matrix. By imposing the kinematic Jacobian matrix rank deficiency, a set of equations is obtained and solved to identify the workspace boundaries. This type of approach can result as too complex when applied to kinematic redundant robots [PRG+17]. The third approach is the numerical one, which identifies the workspace boundaries using the robot forward kinematics. This type of method can be applied to any robot, redundant or not, and its solution is easily understandable [DOXY20]. It is usually employed for the workspace analysis of spatial kinematic redundant robots. The most common method of this third category is the Monte Carlo one [GY06]. In the Monte Carlo method, many random robot configurations are generated to determine the robot workspace. However, this method has several drawbacks [DOXY20, PRG+17]. The generated workspace can be inaccurate, especially on its boundaries [DOXY20], and the real workspace can be different from the one obtained [PRG+17]. The coordinate transformation from joint space to workspace in the forward kinematics is nonlinear. This means that a uniform coordinate distribution in the joint space does not necessarily lead to a uniform distribution of points in the Cartesian workspace [LSFY18]. As a result, some areas in the workspace have a low density of points and others have a high density. Unfortunately, the low density of points areas usually corresponds to the boundaries of the workspace [LSFY18]. On the contrary, workspace inner areas are characterized by a high density of points, leading to high computational time waste.

In [PRG+17], the authors employed an improved version of the Monte Carlo method to calculate the robot workspace, called Gaussian Growth. The method consists of generating an amount of starting robot configurations using the classical Monte Carlo method to obtain a seed workspace, which will be inaccurate in some areas. Then, it is grown employing a Gaussian, or normal, random distribution, until the workspace is accurately approximated. To populate the poorly defined workspace regions, the algorithm takes the robot configuration for one position in the inaccurate area and slightly modify this configuration through a multivariate normal distribution. So, new robot configurations and workspace points are identified and the poorly defined regions are improved. In [ZHZ+18], the authors presented another Monte Carlo method combined with the Gaussian distribution plus a Voxel algorithm [FSMVM+13] to analyze the workspace of a nine degrees of freedom kinematic redundant robot. The obtained seed workspace is expanded by setting an accuracy threshold to describe each region accurately. Then, a Voxel algorithm is proposed to compute the obtained workspace volume. All these methodologies based on an improved Monte Carlo method can identify the manipulator workspace.

However, they are still affected by the Monte Carlo drawbacks, although to a lesser extent. Computing only the boundaries is enough to describe the robot workspace and obtain its volume, but not possible with a Monte Carlo based method.

Other types of workspace generation algorithms exist that employ the workspace density and the N-dimensional Euclidean motion group $SE(N)$. In [WC04], the authors formulate the workspace generation problem for kinematically redundant robots as a diffusion process employing the Euclidean motion group $SE(N)$ to describe the evolution of the workspace density function. The workspace density is a powerful tool in the case of planar serial arms with revolute joints, as shown in [DDC13]. The workspace density based approach can also be used for plotting the reachability map of special situations, such as in the case of ball joints [DFDC15]. The approach described in [HPX+21] is a step forward in using Euclidean motion group $SE(N)$ to generate a three dimensional workspace. The authors implement a series of convolutions to reduce the computational complexity from a spatial case to a planar one. All these techniques can describe the reachability map, or workspace, of different robot types. However, their development could be more complex and less intuitive.

The workspace determination algorithm described here is an optimized ray-based method. The ray-based method idea is presented in [AEL19, ZCL20, CL22] to determine the interference free and wrench closure workspace and for the trajectory verification of cable-driven robots. Compared to other numerical approaches like pointwise or interval-based analysis, the ray-based approaches provide information about the interference free workspace continuity, precisely determining interior regions [CL22]. Moreover, ray-based methods decrease the computational time with respect to the other approaches [AEL19]. The proposed workspace determination method takes into account more complex robotic architectures, in particular focusing on redundant ones. However, it can be employed with any robot. This workspace determination method is based on an intuitive idea, avoids the computational redundancy that affects the Monte Carlo based methods and identifies only the robot workspace boundaries. The resulting workspace is easy to visualize and obtain in a short time with continuous boundaries. Starting from a set of configurations inside the workspace, the end-effector is moved from its position along several radial directions. When the end-effector reaches the workspace boundary and stops, its position is saved. So, this new method can quickly identify the boundaries of any robot workspace. The robots employed to test the proposed workspace determination algorithm are highly kinematic redundant robots [CB94]. They are perfect

candidates as describing their workspace is a complex problem [DOXY20].

5.2 Proposed workspace determination method

This section describes the proposed workspace determination procedure. The core idea is to identify the workspace boundaries rapidly without losing time collecting points inside it. After initializing the robot in a starting configuration, the end-effector is moved along a set of linear displacement vectors with a target twist $\mathbf{t}_t$ employing the TPIK algorithm. These vectors radiate from the end-effector initial position along different directions. To avoid the robot getting stuck in a singular configuration inside the workspace, two tasks for the kinematic optimization, based on dexterity η and manipulability μ, are included in the TPIK algorithm. If the actual rates of η and μ, i.e. $\dot{\eta}$ and $\dot{\mu}$, are over the desired threshold $\epsilon_{\text{ kinematic}}$, it means that the TPIK algorithm is still optimizing the robot configuration and the end-effector position is not saved.

When $\dot{\eta}$ and $\dot{\mu}$ are lower than $\epsilon_{\text{ kinematic}}$ and the 2-norm of the end-effector velocity $||\mathbf{t}||_2$ goes under a selected threshold $\epsilon_{\text{ velocity}}$, it means that the end-effector has reached the workspace boundary and its position is saved. Afterwards, the robot is set to the starting configuration again and the end-effector is moved along another vector. After the end-effector has been moved along all the displacement

Algorithm 5.1 Workspace determination

Require: A number of starting configurations and a set of displacement vectors along different directions. The thresholds $\delta_{\text{ kinematic}}$ and $\delta_{\text{ velocity}}$ for $\dot{\eta}$ and $\dot{\mu}$ and end-effector twist $\mathbf{t}$, respectively. The target end-effector twist $\mathbf{t}_t$.

```
for i := 1 → number of starting configurations do
    for k := 1 → number of displacement vectors do
        Initialize robot in i^th configuration.
        while η̇ and μ̇ > δ kinematic do
            while End-effector twist ||t||_2 > δ velocity do
                Move end-effector along k^th vector with t_t.
            end while
        end while
        Save end-effector position.
    end for
end for
```

Table 5.1: Details about task name, category, type and hierarchy level in action $\mathscr{A}$. Symbol (E) represents the equality control objective tasks and (I) the inequality ones. Last column list task hierarchies for action $\mathscr{A}$.

Task name	Category	Type	$\mathscr{A}$
End-Effector Velocity	action oriented	E	1^{nd}
Dexterity	optimization	I	2^{rd}
Manipulability	optimization	I	2^{rd}

vectors, the robot is initialized in a new starting configuration and moved along all the displacement vectors again. This process is repeated several times, starting from different configurations to obtain enough points to describe the boundary of the whole workspace. Algorithm 1 sums up the workspace determination procedure. Table 5.1 collects the tasks employed by the TPIK algorithm for the workspace generation and their hierarchical priority levels. The machine and implementation details remained the same ones proposed in Chapter 3 and shown in Table 3.3 on page 48.

5.3 Workspace determination test description

This section describes the three Nimbl'Bot robots employed for testing the proposed workspace determination algorithm. Then, some initialization steps necessary to run the process are presented.

5.3.1 Robots under analysis

In this chapter, the workspace of three different Nimbl'Bot robots is analyzed. The first robot is the NB-R1, already presented in Section 1.4 and shown in Fig. 1.7. In this chapter, the NB-R1 design is showed again in Fig. 5.1. The second robot, called NB-R2, is shown in Fig. 5.2. It is composed of six NB-modules serially attached and a final revolute joint. In total, it has 12 degrees of freedom. The third robot, called NB-R3 and represented in Fig. 5.3, is organized as the NB-R1, i.e. with ten NB-modules and two links plus a final revolute joint. So, it has 21 degrees of freedom too. However, two angular offsets β_1 and β_2 are inserted between the link l_1 and the first elbow NB-module and between the second link l_2 and the first wrist

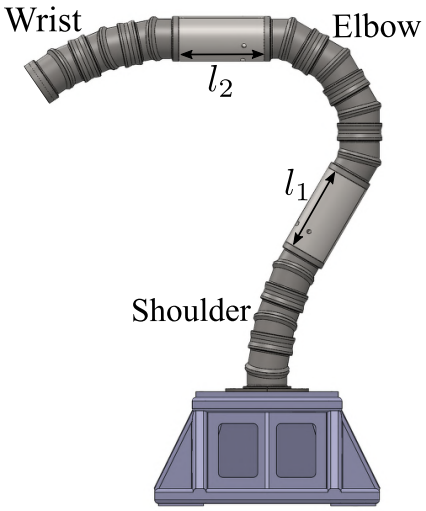


Figure 5.1: Robot NB-R1 formed of ten NB-modules plus two links $l_1 = l_2 = 0.2$ m

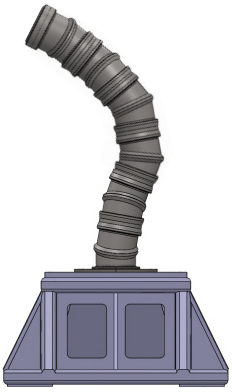


Figure 5.2: Robot NB-R2 formed of six NB-modules

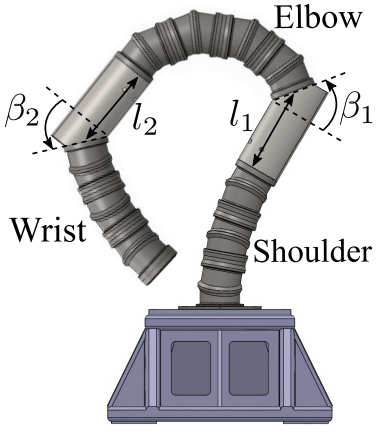


Figure 5.3: Robot NB-R3 formed of ten NB-modules plus two links $l_1 = l_2 = 0.2$ m and two offsets $\beta_1 = \beta_2 = \pi/4$ rad

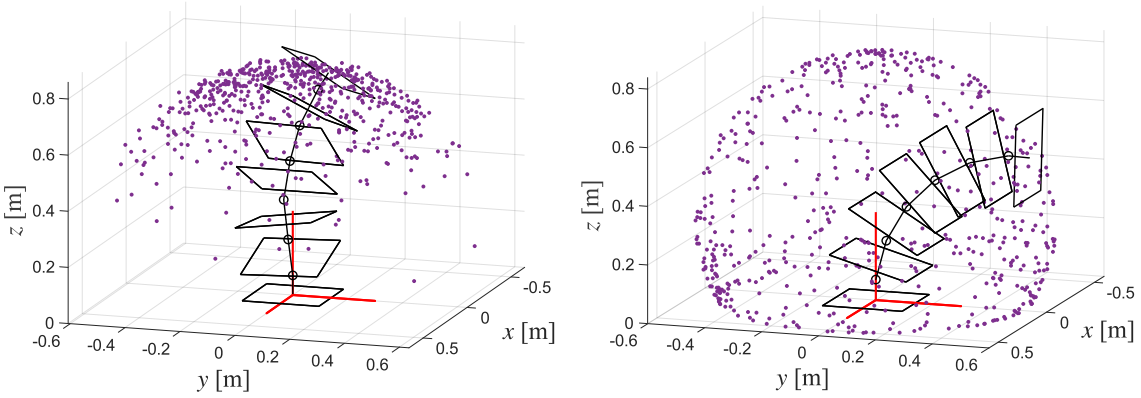
NB-module. The length of both the links l_1 and l_2 is equal to 0.2 m and the the offsets β_1 and β_2 equal to -45°. Table 5.2 recalls the robot dimensions.

Table 5.2: Design details of robots used for workspace analysis

NB-module half height r	0.07 m
NB-module slope α	15°
Link lengths l_1 and l_2	0.2 m
Offsets β_1 and β_2	-45°

5.3.2 Workspace determination test features

Before applying the workspace determination algorithm, a rule to initialize the robot has to be defined. A uniform random generation is employed to obtain the starting configurations in this case. Considering one NB-module, the values of the motor positions q_1 and q_2 are randomly generated. These values are applied to all the NB-modules included in one robot. So, each NB-module in the robot is initialized in the same configuration. After several tests, this type of initialization was selected because it generated the most uniform point distribution on the workspace boundaries. The generated points were not uniformly distributed if all the NB modules were randomly initialized with different joint angle values. Figure 5.4 shows the end-effector position distribution for several starting configuration using the random or proposed initialization. It appears how the randomly generated points are located in the upper area of the workspace while the proposed initialization distributes uniformly the starting point.



(a) Random initialization

(b) Proposed initialization, all NB-module are initialized with the same joint couple

Figure 5.4: Comparison between random and proposed initialization for robot starting configurations. The purple points are the starting end-effector position generated through the two methods.

Then, the robot end-effector is moved along a set of displacement vectors. In the simulation, a total of 14 unit vectors is used. The first six vectors are oriented along the positive and negative direction of axes $\vec{x}$, $\vec{y}$ and $\vec{z}$, respectively,

$$\vec{v}_{1,2} = \begin{bmatrix} \pm 1 \\ 0 \\ 0 \end{bmatrix}, \quad \vec{v}_{3,4} = \begin{bmatrix} 0 \\ \pm 1 \\ 0 \end{bmatrix}, \quad \vec{v}_{5,6} = \begin{bmatrix} 0 \\ 0 \\ \pm 1 \end{bmatrix}. \tag{5.1}$$

The other eight vectors are a combination of displacements along all the axes $\vec{x}_0$, $\vec{y}_0$ and $\vec{z}_0$,

$$\begin{aligned} \vec{v}_7 &= \begin{bmatrix} -c \\ -c \\ -c \end{bmatrix}, \quad \vec{v}_8 = \begin{bmatrix} c \\ -c \\ -c \end{bmatrix}, \quad \vec{v}_9 = \begin{bmatrix} -c \\ c \\ -c \end{bmatrix}, \quad \vec{v}_{10} = \begin{bmatrix} c \\ c \\ -c \end{bmatrix}, \\ \vec{v}_{11} &= \begin{bmatrix} -c \\ -c \\ c \end{bmatrix}, \quad \vec{v}_{12} = \begin{bmatrix} c \\ -c \\ c \end{bmatrix}, \quad \vec{v}_{13} = \begin{bmatrix} -c \\ c \\ c \end{bmatrix}, \quad \vec{v}_{14} = \begin{bmatrix} c \\ c \\ c \end{bmatrix}, \end{aligned} \tag{5.2}$$

where c is equal to $1/\sqrt{3}$. The displacement vectors applied to the end-effector are shown in Fig. 5.5. These 14 vectors are chosen for the simulation to move the robot uniformly in all the directions. In future, more analysis could be done to identify if more or fewer displacement vectors gives better results. To ensure a uniform point distribution on the whole workspace, the displacement vectors are rotated around axis $\vec{z}_1$ oriented as the base frame axis $\vec{z}_0$ and applied to the end-effector frame origin, as shown in Fig. 5.6. The rotation angle equals the azimuth angle ϕ of the transformation matrix between the base frame and the end-effector frame.

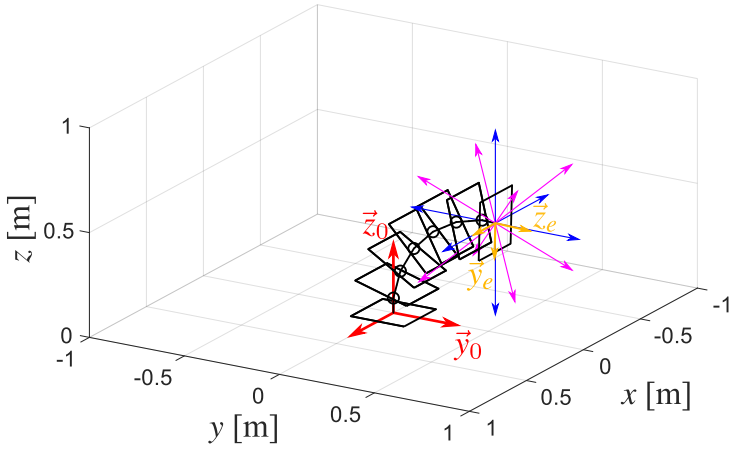


Figure 5.5: Representation of the NB-R2 and displacement vectors applied to the robot end-effector position. $\mathcal{F}_0$ is the base frame and $\mathcal{F}_e$ is the end-effector frame.

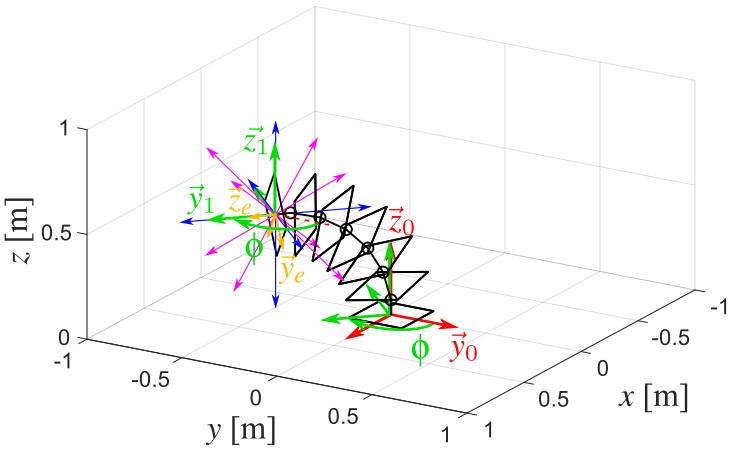


Figure 5.6: Representation of the NB-R2 with $\phi = 135°$ and displacement vectors applied to the end-effector position rotated of the same ϕ. $\mathcal{F}_0$ is the base frame, $\mathcal{F}_1$ is the base frame rotated of ϕ around $z_0 \equiv z_1$ and $\mathcal{F}_e$ is the end-effector frame.

Table 5.3: Details of variables used in workspace determination algorithm

$\delta_{\text{ kinematic}}$	10^{-6}
$\delta_{\text{ velocity}}$	10^{-3} m/s
$\|\|\mathbf{t}_t\|\|_2$	0.14 m/s

The azimuth angle ϕ can be obtained using the notation presented in [GLC+21]. Table 5.3 provides the test implementation details and values.

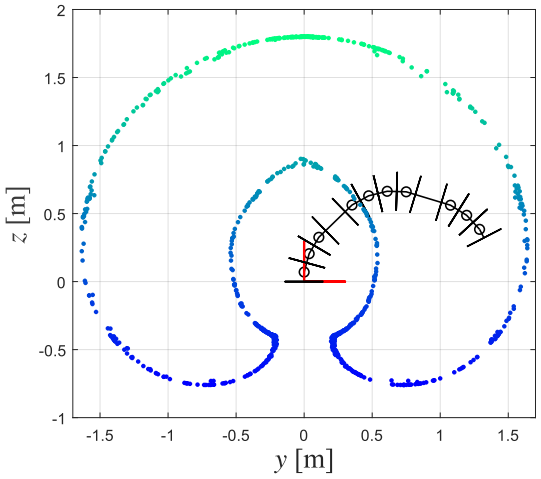
In this case, the process of following the 14 vectors is repeated 580 times. So, the starting number of robot configurations is 580 and the total number of obtained points on the workspace boundaries is 8120. In case of fewer repetitions, there were not enough points to correctly identify the boundaries for the analyzed robots. In case of more repetitions, no substantial change was seen in the boundary identification quality. So, the repetition number is set to 580 times, obtaining enough points on all the boundary areas and limiting the time consumption. This amount of points allows for identifying both the smaller NB-R2 and the bigger NB-R1/3 workspaces. There is no rule to identify the best repetition number, which is obtained by testing the process several times and checking the result quality. In fact, a larger number of points could be necessary to detect the workspace of robots with bigger dimensions.

5.4 Workspace determination results

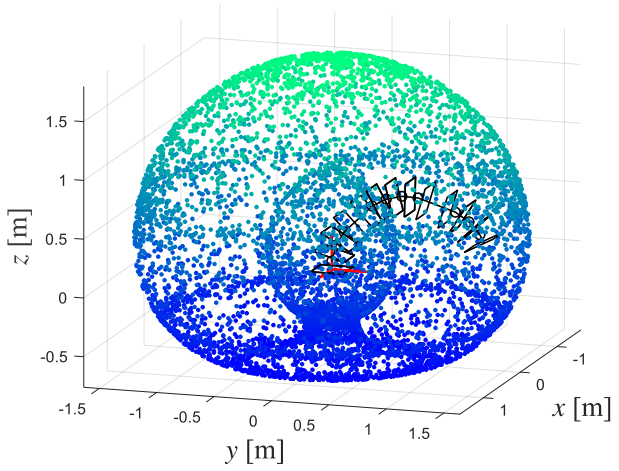
Firstly, this section presents and discusses the results obtained using the kinematically optimized ray-based method. Then, its performance is compared with the results obtained using pseudo-inverse Jacobian ray-based and Monte Carlo methods.

5.4.1 Proposed optimized ray-based method results

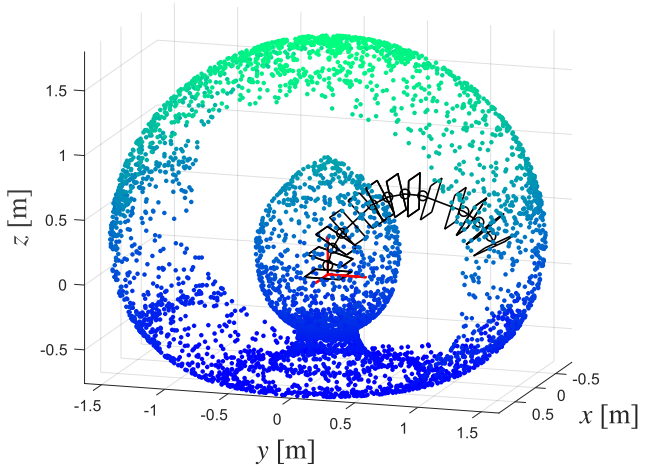
The total number of generated points to determine each workspace is 8120. Figure 5.7 shows three views of the NB-R1 workspace. The colors are assigned to the points on the base of their height along axis z and help the visual identification of the workspace. The NB-R1 workspace boundaries are uniformly and continuously identified and can be easily visualized. This workspace is symmetric around axis z. Figure 5.7a shows the vertical section, defined by the yz-plane, of the NB-R1 workspace. Figure 5.7b shows a 3D view of the complete NB-R1 workspace and Fig. 5.7c presents a 3D view of the inner boundaries of the NB-R1 workspace. Then, Fig. 5.8 shows three views of the NB-R2 workspace. Figure 5.8a shows the vertical section, defined by the yz-plane, of the NB-R2 workspace. Figure 5.8b shows a 3D view of the complete NB-R2 workspace and Fig. 5.8c presents a 3D view of the inner boundaries of the NB-R2 workspace. Again, the NB-R2 workspace is uniformly and continuously detected and symmetric around axis z. In this case, the total workspace is smaller than the NB-R1 case. This is due to a lower number of NB-modules and the absence of the two links l_1 and l_2 . Finally, Fig. 5.9 shows three views of the NB-R3 workspace. Figure 5.9a shows the vertical section, defined by the yz-plane, of the NB-R3 workspace. Figure 5.9b shows a 3D view of the complete NB-R3 workspace and Fig. 5.9c presents a 3D view of the inner boundaries of the NB-R3 workspace. The NB-R3 workspace has a similar volume of the NB-R1 one since both robots have the same number of module and link lengths. However, the NB-R3 workspace is not symmetric as in the two previous cases because of the two offsets β_1 and β_2. The workspace distribution is moved along the positive side of axis y since β_1 and β_2 are rotated about the axis x of a negative value. The workspace maintains its symmetry with respect to the yz-plane since there is no offset about the axis y. The NB-R3 workspace boundaries are, in general, uniformly and continuously identified. In fact, the workspace shape can be easily visualized. Nevertheless, the inner boundaries are rougher and not perfectly described in this case. This happens because, despite the presence of the kinematic optimization tasks, the robot reaches singular configurations inside the workspace, not reaching



(a) Vertical section in yz-plane

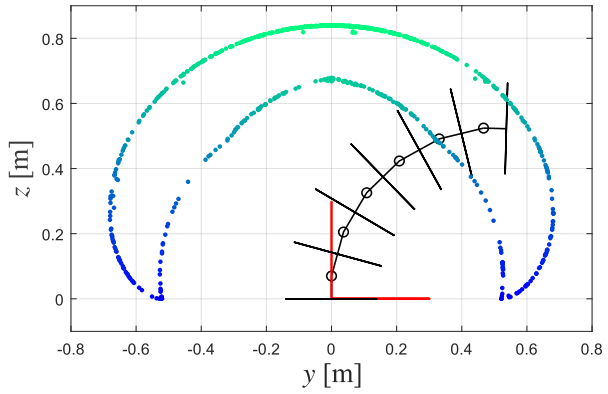


(b) Complete workspace view

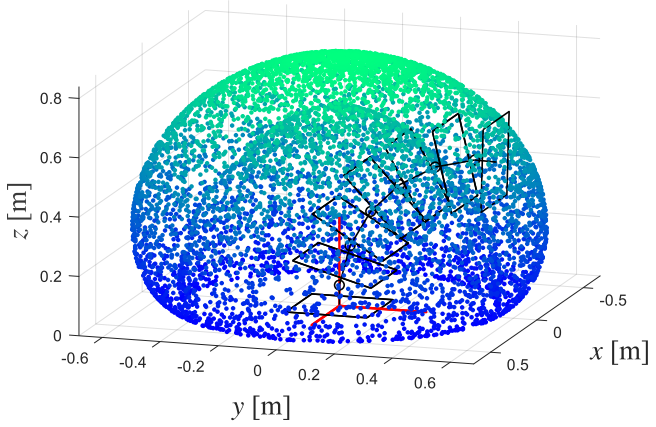


(c) Internal workspace view

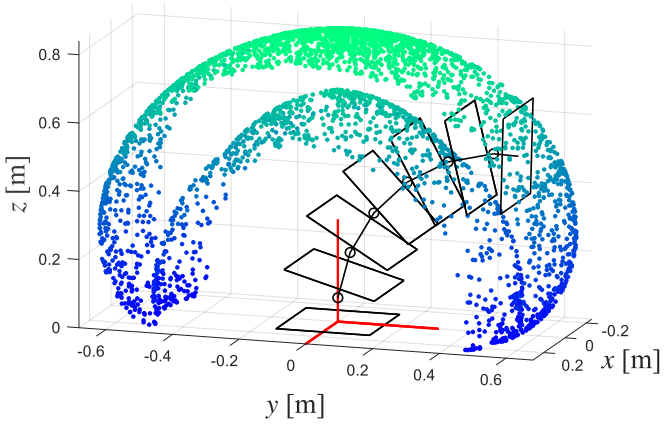
Figure 5.7: Generated points describing the NB-R1 workspace using the proposed optimized ray-based method



(a) Vertical section in yz-plane

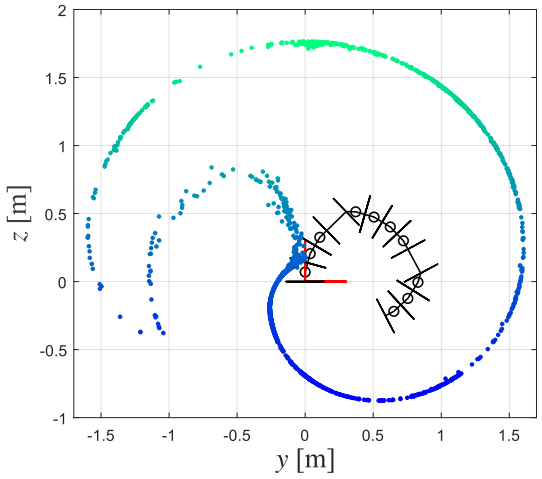


(b) Complete workspace view

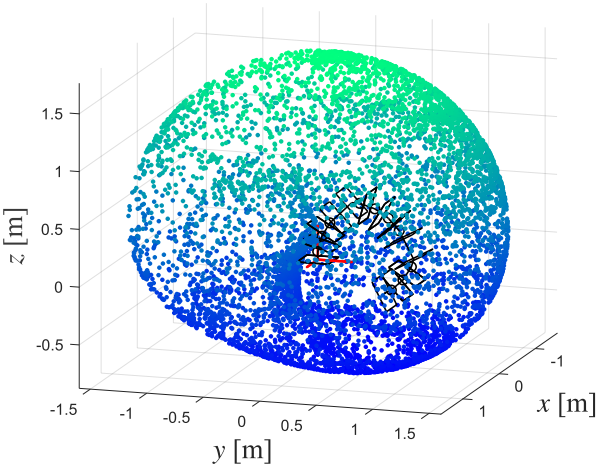


(c) Internal workspace view

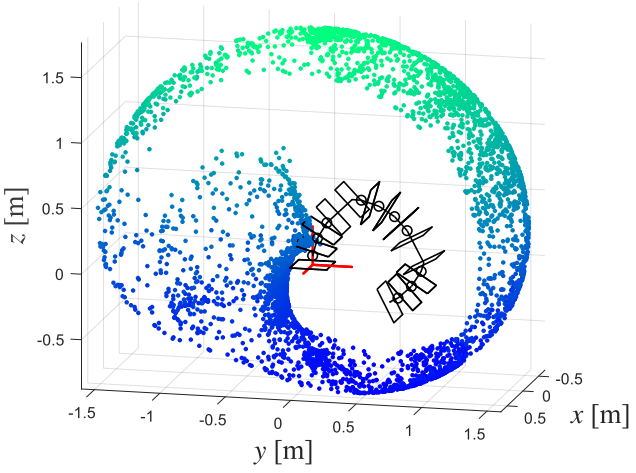
Figure 5.8: Generated points describing the NB-R2 workspace using the proposed optimized ray-based method



(a) Vertical section in yz-plane



(b) Complete workspace view



(c) Internal workspace view

Figure 5.9: Generated points describing the NB-R3 workspace using the proposed optimized ray-based method

Table 5.4: Total computational time to determine the 8120 points that describe the robot workspace through the ray-based method

Robot name	Total computational time
NB-R1	6.4 min
NB-R2	0.8 min
NB-R3	4.5 min

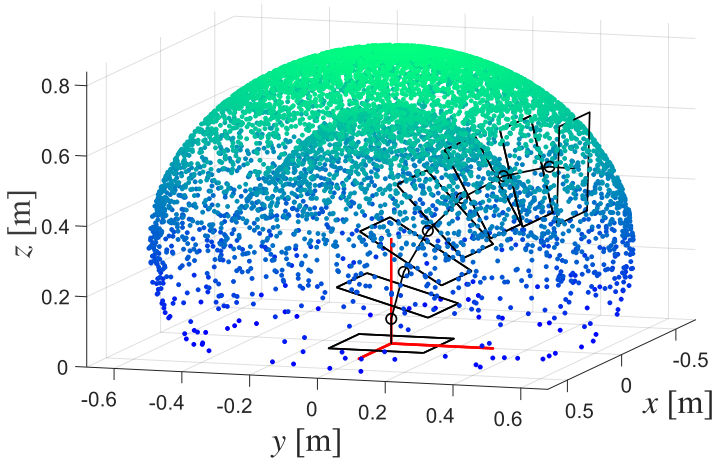


Figure 5.10: Points describing NB-R2 workspace obtained starting robot joints with all different random values

the real boundary. This behavior does not appear with the other robots and it is probably due to the presence of β_1 and β_2. More analysis will be done on this point in the future. Table 5.4 provides the total time to determine the workspace of the three analyzed robots.

As previously described, the robots are initialized in a specific way to obtain a uniform distribution of points around the entire workspace. Figure 5.10 shows the workspace of the NB-R2 when the robot starting configurations are generated initializing all the joints with different random values. As a result, few points describe the workspace lower part, being more concentrated in the upper part. This phenomenon proves the utility of initializing the robot configuration in the proposed way.

Figure 5.11 shows the NB-R1 robot configuration in black for a specific position on its workspace boundary and the linear Jacobian matrix singular vectors in magenta. The magnitude of the singular vectors tangent to the workspace boundary differs from zero, while the magnitude of the singular vector aligned to the robot

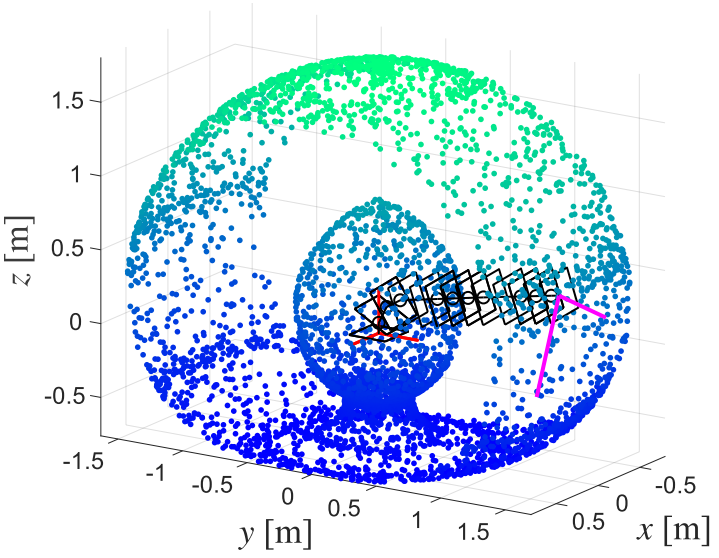


Figure 5.11: Robot configuration, black lines, and linear Jacobian matrix singular vectors, magenta lines, for the NB-R1 in one point on its workspace boundaries

configuration is zero. This means the robot has lost one degree of freedom in the linear task space, reaching a singular configuration. The end-effector can no longer move in the zero singular vector direction since it has reached the workspace boundary. The magnitude of at least one singular vector is always equal to zero on the points that compose the workspace boundaries of NB-R1, NB-R2 and NB-R3.

5.4.2 Comparison with the pseudo-inverse Jacobian ray-based method results

As explained, the TPIK algorithm employs two kinematic optimization tasks to improve the robot configuration and avoid getting stuck in a singularity while moving the robot end-effector. Here, the optimization tasks are not used to compare with the optimized ray-based method. Since the optimization tasks are disabled, the remaining task is related to the end-effector velocity based on the pseudo-inverse kinematic Jacobian matrix. So, the non-optimized ray-based method is named pseudo-inverse Jacobian ray-based. Figure 5.12 shows the generated points for the NB-R1 workspace without the optimization tasks. Some of the generated points did not reach the workspace boundary and are resulted inside the workspace. The majority of these points are close to the upper area of the internal boundaries. Here, the robot almost reached the workspace boundary but stopped in a singular configuration. Similar behavior can be seen for the NB-R2 using the pseudo-inverse Jacobian ray-based method, as shown in Fig. 5.13. Figure 5.14 shows the workspace bound-

aries of the NB-R3 for the pseudo-inverse Jacobian ray-based method. Some final end-effector positions were already returned by the algorithm inside the workspace for the NB-R3 optimized test as shown in Fig. 5.9c. However, in the case of no optimization, more final positions are returned inside the workspace and further from the boundaries.

The only advantage of the pseudo-inverse Jacobian ray-based method is that it takes less time to run. Table 5.5 shows the time comparison between the optimized and pseudo-inverse Jacobian ray-based ray-based tests. However, the time difference is slight and also affected by stopping the process earlier for some configurations. In the end, it is better to employ the optimization tasks to avoid getting the points

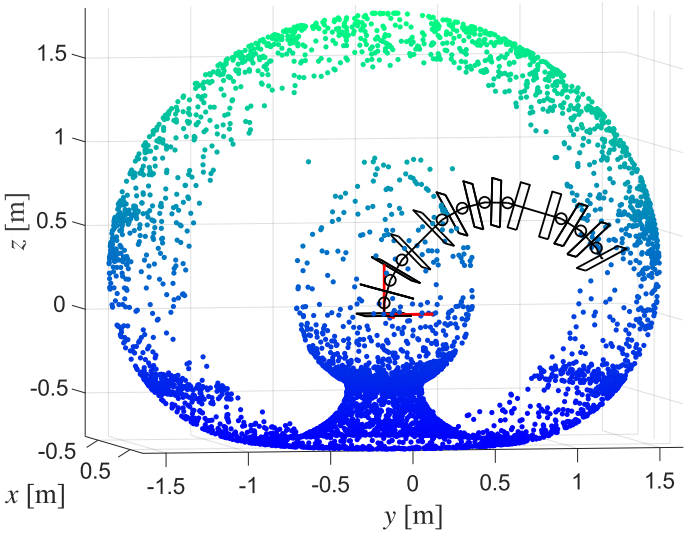


Figure 5.12: Internal boundary view of points describing NB-R1 workspace obtained with pseudo-inverse Jacobian ray-based method

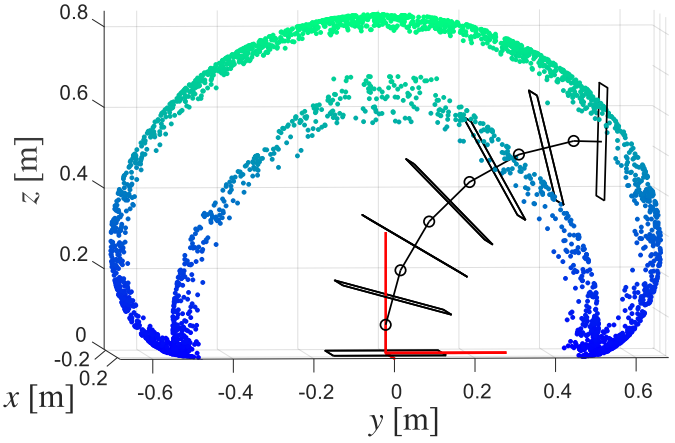


Figure 5.13: Internal boundary view of points describing NB-R2 workspace obtained with pseudo-inverse Jacobian ray-based method

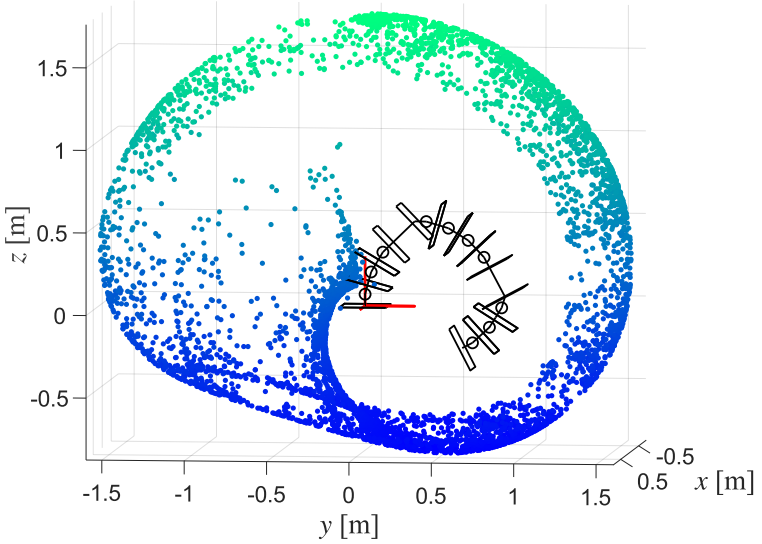


Figure 5.14: Internal boundary view of points describing NB-R3 workspace obtained with pseudo-inverse Jacobian ray-based method

Table 5.5: Comparison of kinematically optimized and pseudo-inverse Jacobian ray-based methods in workspace determination

Robot	Method	Number of points	Time	Boundary comments
NB-R1	Optimized ray-based	8120	6.4 min	Uniform and continuous
	Pseudo-inverse Jacobian ray-based	8120	4.7 min	Internal partially wrong
NB-R2	Optimized ray-based	8120	0.8 min	Uniform and continuous
	Pseudo-inverse Jacobian ray-based	8120	0.7 min	Internal partially wrong
NB-R3	Optimized ray-based	8120	4.5 min	Uniform and continuous
	Pseudo-inverse Jacobian ray-based	8120	3.7 min	Internal partially wrong

inside the workspace.

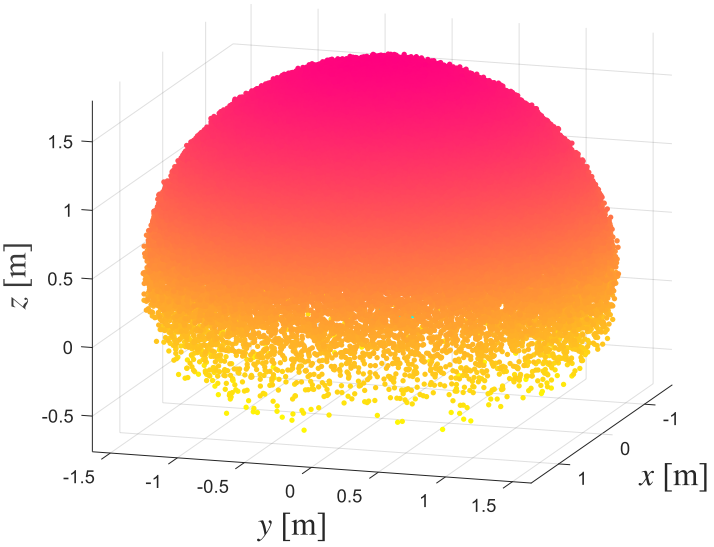
5.4.3 Comparison with Monte Carlo method results

Here, the optimized ray-based method performance is compared to the Monte Carlo one. In the Monte Carlo case, 500000 random configurations were generated. This high number is chosen to obtain the vastest workspace area possible for each robot. Higher numbers were used for the Monte Carlo test and similar results were always obtained. So, the random configuration number was set to 500000 to reduce the computational time. Table 5.6 compares the results of optimized ray-based and Monte Carlo methods for each robot. In all cases, the optimized ray-based perfor-

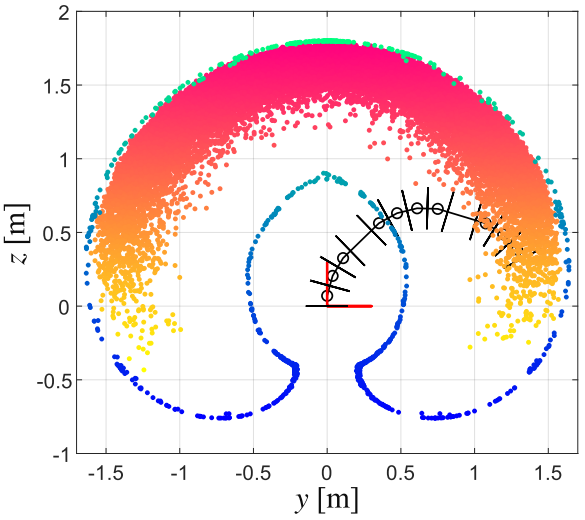
mance is better. The computational time of the Monte Carlo method for the three Nimbl'Bot robots is more or less the same since the process only requires generating random robot configurations and saving the end-effector positions. Figure 5.15 shows a 3D and section views of the NB-R1 workspace generated using the Monte Carlo method. Figure 5.15b compares the NB-R1 workspace section generated using the Monte Carlo method, red-yellow points, together with the workspace section generated using the optimized ray-based method, green-blue points. The upper external boundaries were correctly identified. However, the lower and internal boundaries are entirely ignored and the real robot workspace can not be reconstructed. On the contrary, the proposed optimized ray-based workspace determination algorithm can identify the correct NB-R1 boundaries in much less time using fewer points. Figure 5.16 shows a 3D and section views of the NB-R2 workspace generated using the Monte Carlo method. Figure 5.17 shows a 3D and section views of the NB-R3 workspace generated using the Monte Carlo method. The same comparison with the optimized ray-based method is shown for these other two robots and the same considerations can be made. The Monte Carlo method can not generally be used for the workspace determination of the Nimbl'Bot robot. In fact, some groups of points are more likely to be selected than others. This behavior results in a wrong identification of some boundaries.

Table 5.6: Comparison of optimized ray-based and Monte Carlo methods in workspace determination

Robot	Method	Number of points	Time	Boundary comments
NB-R1	Optimized ray-based	8120	6.4 min	Uniform and continuous
	Monte Carlo	500000	26 min	Lower and internal missing
NB-R2	Optimized ray-based	8120	0.8 min	Uniform and continuous
	Monte Carlo	500000	26 min	Lower and internal missing
NB-R3	Optimized ray-based	8120	4.5 min	Uniform and continuous
	Monte Carlo	500000	26 min	Lower and internal missing

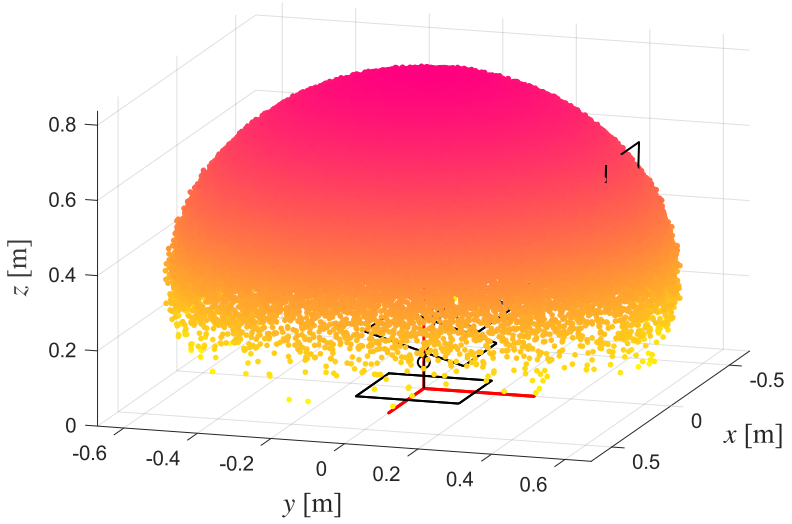


(a) Complete workspace view generated with Monte Carlo method

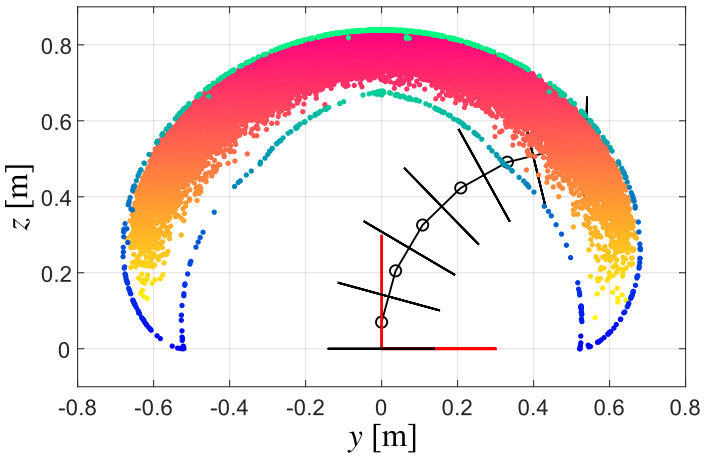


(b) Vertical section in yz-plane with comparison between optimized ray-based and Monte Carlo workspaces

Figure 5.15: Workspace points generated with Monte Carlo method describing NB-R1 workspace and comparison with optimized ray-based process results. Red-yellow points are generated with the Monte Carlo method. Blue-green points are generated with the optimized ray-based method.

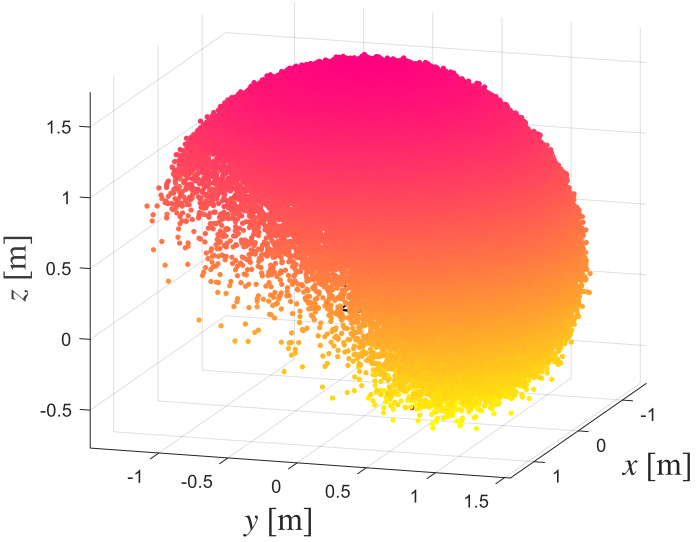


(a) Complete workspace view generated with Monte Carlo method

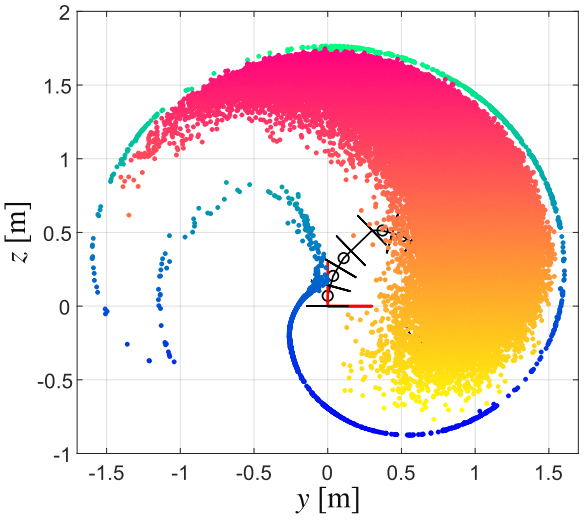


(b) Vertical section in yz-plane with comparison between optimized ray-based and Monte Carlo workspaces

Figure 5.16: Workspace points generated with Monte Carlo method describing NB-R2 workspace and comparison with optimized ray-based process results. Red-yellow points are generated with the Monte Carlo method. Blue-green points are generated with the optimized ray-based method.



(a) Complete workspace view generated with Monte Carlo method



(b) Vertical section in yz-plane with comparison between optimized ray-based and Monte Carlo workspaces

Figure 5.17: Workspace points generated with Monte Carlo method describing NB-R3 workspace and comparison with optimized ray-based process results. Red-yellow points are generated with the Monte Carlo method. Blue-green points are generated with the optimized ray-based method.

5.5 Proposed workspace surface plot

After computing the points on the workspace boundaries, it is necessary to interpolate them to generate the workspace surface. This section describes the algorithm to interpolate all the points and plot the workspace surface. This procedure makes

it possible later to compute other workspace features like its surface area or volume. Moreover, an automatic process to determine the reachability of a 3D point could be implemented. However, the interpolation process presented here applies only to symmetrical workspaces. This means that only the workspace surfaces of the NB-R1 and NB-R2 can be computed. The NB-R3 workspace has a too complex non-symmetrical shape and can not be interpolated in a surface at the moment. Firstly, the point interpolation and surface generation method is explained. Then, the NB-R1 and NB-R2 workspace surfaces are shown.

This surface plot process is developed using the program MATLAB. Starting from the workspace point cloud, a cubic grid that contains all the points is created in the 3D space using the function `meshgrid`. This grid comprises several equidistant nodes spread along all the axes x, y and z that will be used for the first part of the process. A smaller step between each node leads to more nodes and a denser grid. Then, the generated points are substituted with the closest node on the previously defined 3D grid. More than one point can be substituted with the same node on the grid. This means there will be at most as many nodes on the workspace boundary as the generated points. So, a denser grid will lead to a more accurate surface plot. Figure 5.18 shows the selected nodes closer to the determined workspace boundary points. The set of these nodes is called A and the coordinates are named (x_A, y_A, z_A).

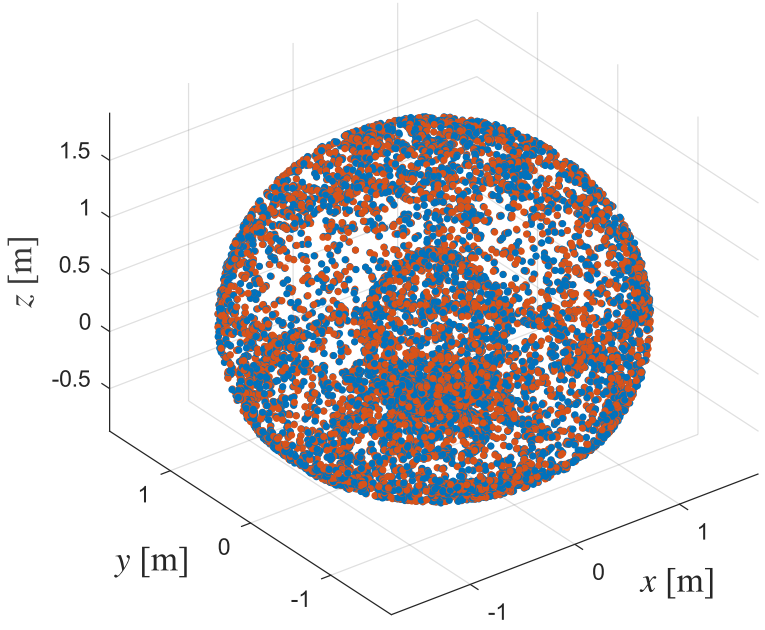
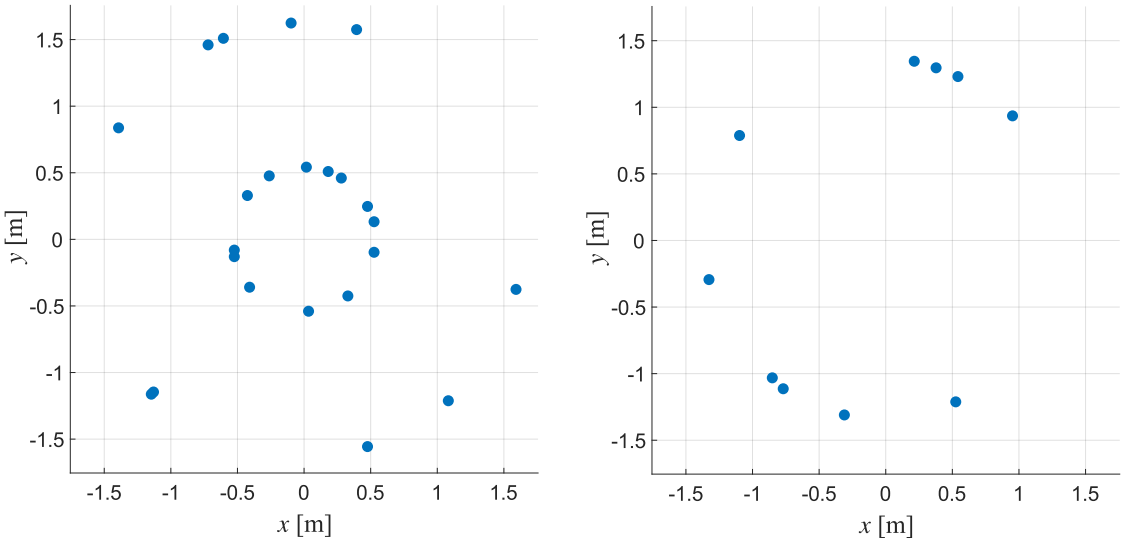


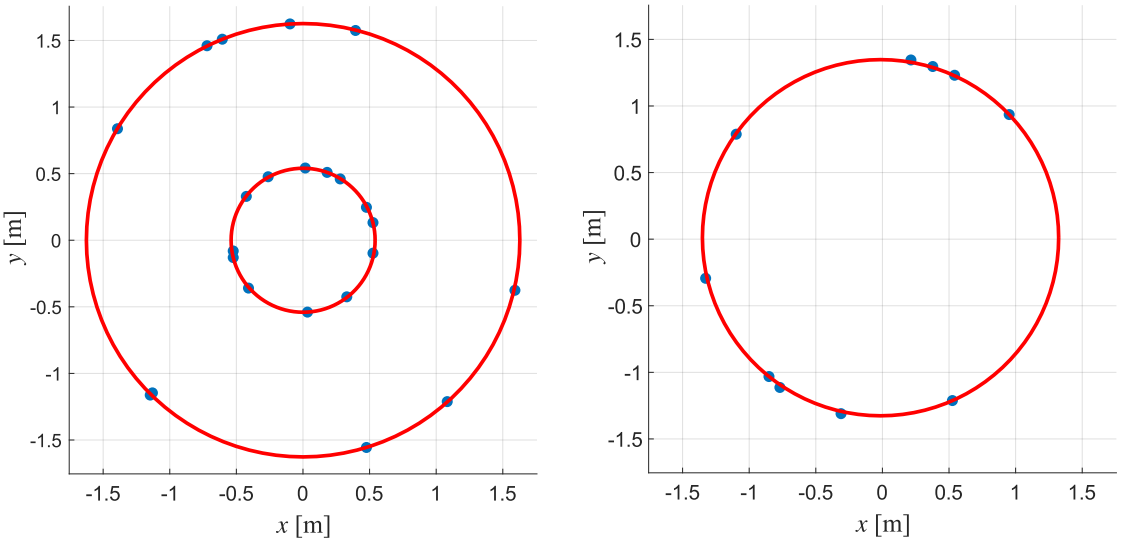
Figure 5.18: NB-R1 workspace boundary points in red and selected nodes on grid in blue

The interpolation process starts sectioning the grid along axis z. As shown in Fig. 5.19, a xy section of Fig. 5.18 shows one or two circumferences for the symmet-



(a) Selected grid nodes at height $z_A = 0.11$ m (b) Selected grid nodes at height $z_A = 1.13$ m

Figure 5.19: NB-R1 selected nodes on grid for some horizontal sections in xy-plane for grid value z_A along z



(a) Selected grid nodes at height $z_A = 0.11$ m (b) Selected grid nodes at height $z_A = 1.13$ m

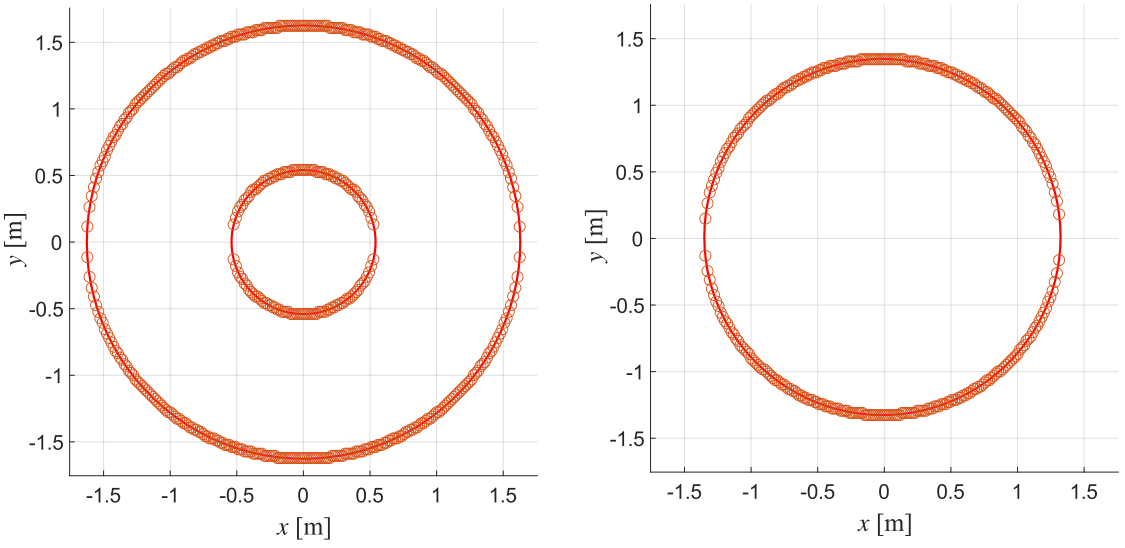
Figure 5.20: Reconstructed circumferences with center (x_c, y_c) and radius R_1 and R_2 starting from the NB-R1 selected nodes on grid for some horizontal sections in xy-plane for grid value z_A along z

rical workspaces based on the height along z. These nodes are identified by $A(z_A)$ where z_A is a grid value along z. They are interpolated to identify the coefficients of two circumferences that pass through the nodes. Only one circumference is considered when the identified circumferences are too close because the nodes form just one circle, as shown in Fig. 5.19b. A minimization problem is solved to obtain the coefficients of the circumferences.

$$\begin{aligned} \min_{\substack{x_c, y_c, \\ R_1, R_2}} \quad & \text{avg}\{((x_A - x_c)^2 + (y_A - y_c)^2 - R_1)((x_A - x_c)^2 + (y_A - y_c)^2 - R_2)\} \\ & \forall (x_A, y_A) \in A(z_A) \end{aligned} \tag{5.3}$$

The Eq. 5.3 minimize the average distance of each point from the two circumferences identified by the same center (x_c, y_c) and radius R_1 and R_2. If the difference between R_1 and R_2 is under a certain threshold, only one circumference is considered. This process is repeated $\forall z_A \in A$. The reconstructed circumferences starting from the selected nodes are plotted in Fig. 5.20.

At this point, one or two circumferences have been identified on each grid height. The second part of the interpolation process starts and the previous 3D grid is deleted. A new grid is initialized equal to the previous one. All the nodes that compose this new grid are collected in a set called B with coordinates named (x_B, y_B, z_B).



(a) Selected grid nodes at height $z_C = 0.11$ m (b) Selected grid nodes at height $z_C = 1.13$ m

Figure 5.21: New selected nodes on grid in coincidence with the obtained circumferences for the NB-R1 workspace for some horizontal sections in xy-plane for grid value z_C along z

Algorithm 5.2 Grid node coefficient assignment

Require: Grid around the desired generated workspace, called B, and nodes on grid with coordinates named (x_B, y_B, z_B). Selected nodes on the workspace boundary C with coordinates named (x_C, y_C, z_C).

Initialize variable sign $s = -1$.
for $\forall z_B \in B$ **do**
    for $\forall x_B \in B$ **do**
        for $\forall y_B \in B$ **do**
            Compute minimum distance d between y_B and closest y_C.
            Assign value $s \doteq d$ to node (x_B, y_B, z_B).
            if $d = 0$ **then**
                Switch sign $s = -s$ because passed node (x_C, y_C, z_C) on boundary.
            end if
        end for
    end for
end for

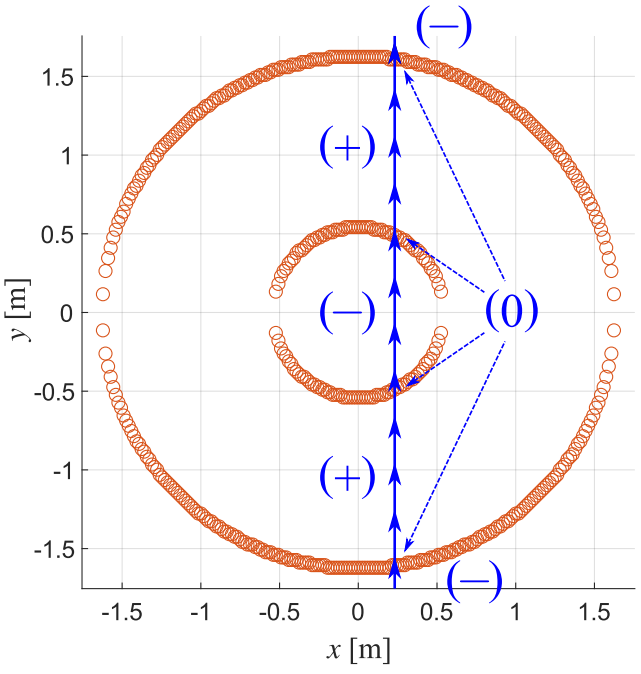
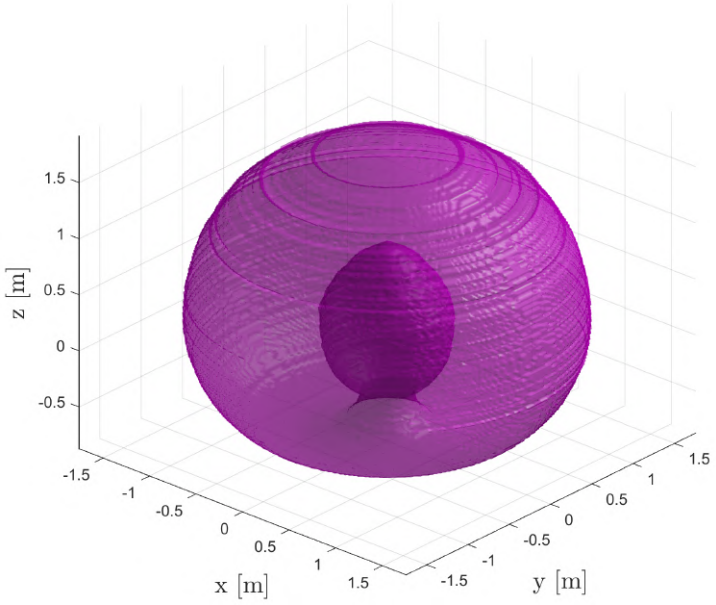


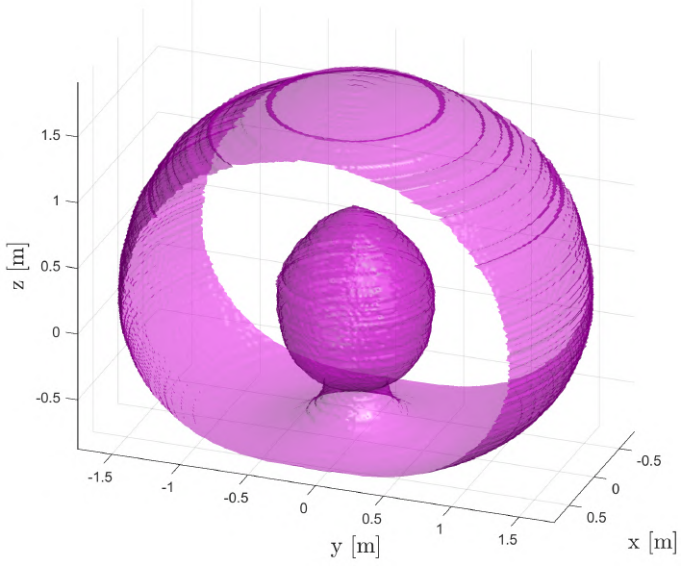
Figure 5.22: Processes of coefficient value assignment to each node in the grid built on the NB-R1 workspace for some horizontal sections in xy-plane for grid value z_B along z. Full procedure explained in Algorithm 5.2.

The nodes that coincide with the computed circumferences are also collected in another set called C with coordinates (x_C, y_C, z_C). So, the set C is composed of the

nodes on the workspace boundaries. Figure. 5.21 shows the selected nodes that coincide with the circumferences. After this step, a positive or negative coefficient is assigned to each node of the set B. This coefficient is equal to the minimum distance d between y_B and closest y_C for each couple (x_B, z_B). A positive coefficient is assigned when the node is inside the workspace and becomes negative for the nodes outside. When y_B and y_C are equal, a 0 value is assigned since that node is on the workspace boundary. Algorithm 5.2 and Fig. 5.22 summarizes this procedure.

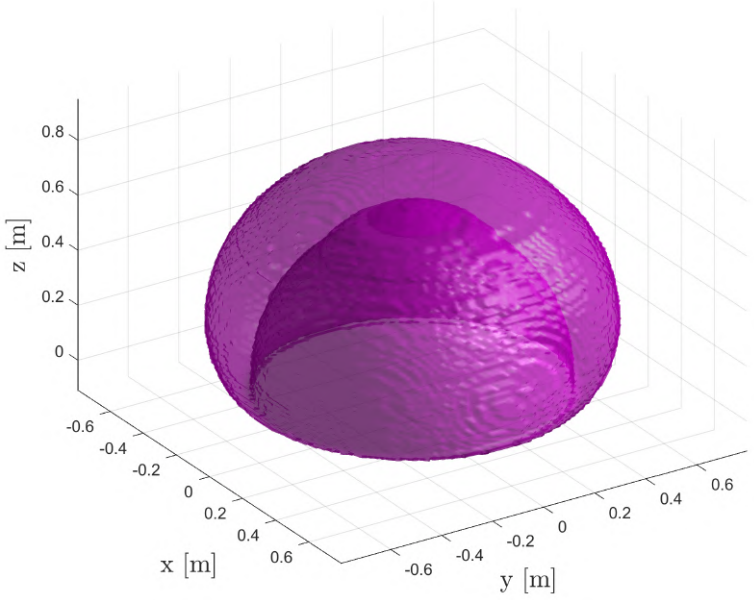


(a) Complete workspace view

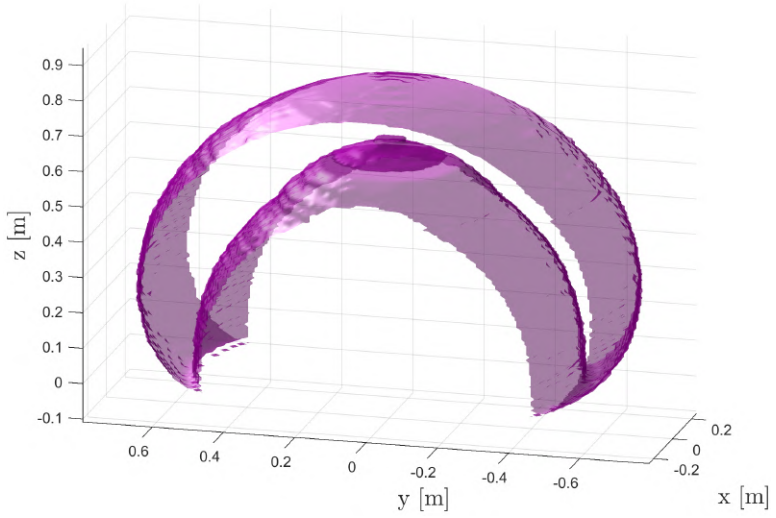


(b) Internal workspace view

Figure 5.23: Surface of the NB-R1 workspace generated interpolating the points obtained with the workspace determination algorithm



(a) Complete workspace view



(b) Internal workspace view

Figure 5.24: Surface of the NB-R2 workspace generated interpolating the points obtained with the workspace determination algorithm

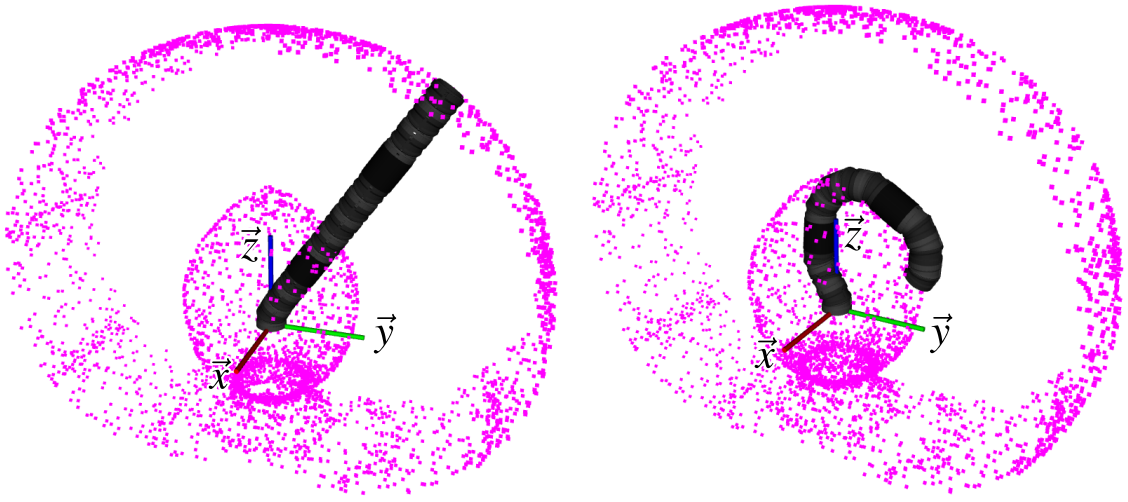
Once, a coefficient is assigned to each node, it is possible to use the `isosurface` function for generating and plotting the workspace boundary surface. Figure 5.23 shows the surface of the NB-R1 workspace, both 3D and inner views. In the NB-R1 case, the grid had a step of 0.03 m. Figure 5.24 shows the surface of the NB-R2 workspace, both 3D and inner views. In the NB-R2 case, the grid had a step of 0.01 m. The surface has been fully reconstructed and is easy understandable. This method can be applied only to the NB-R1 and NB-R2 robots because they are symmetrical and the circumferences can be used to interpolate the point. The

NB-R3 workspace boundaries have too complex shapes for this interpolation process.

5.6 Discussion about the proposed workspace determination algorithm

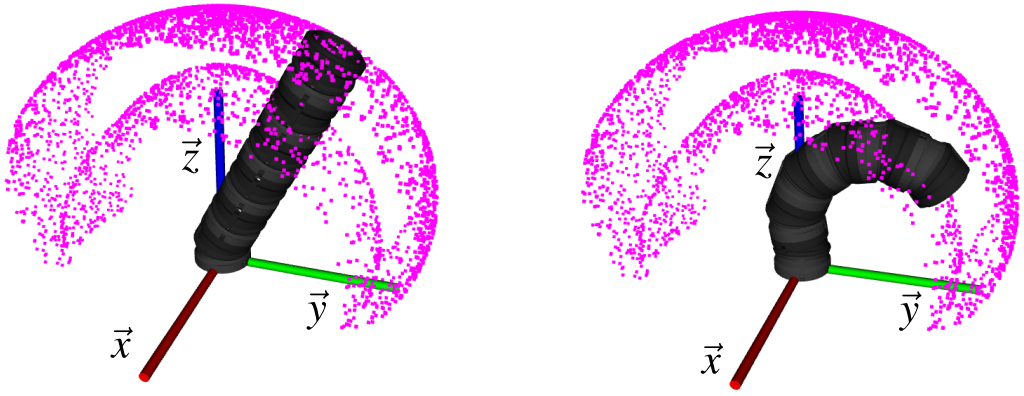
This section highlights some limitations of the proposed workspace determination algorithm. The first major limitation is the missing prior knowledge about the analyzed manipulator workspaces. So, there is no way to ensure the correctness of the result obtained by this algorithm. However, one way to check the result correctness is by plotting some robot configuration for each workspace. Figure 5.25 shows the NB-R1 robot CAD model plotted using ROS RViz tool in two different configurations with the end-effector on the detected boundaries. The obtained workspace points are shown in magenta. Trying to move the robot further, reaching a position outside this workspace is impossible. So, the correctness of the workspace detection is demonstrated by testing. Figures 5.26 and 5.27 show the NB-R2 and NB-R3 robot CAD models, respectively, in two different configurations with the end-effector on the detected boundaries. The obtained workspace points are shown in magenta. ROS RViz tool is again employed to plot the robots and their workspaces. The same test can be performed to prove the result fidelity.

The proposed algorithm correctly identified the inner boundaries of the previous



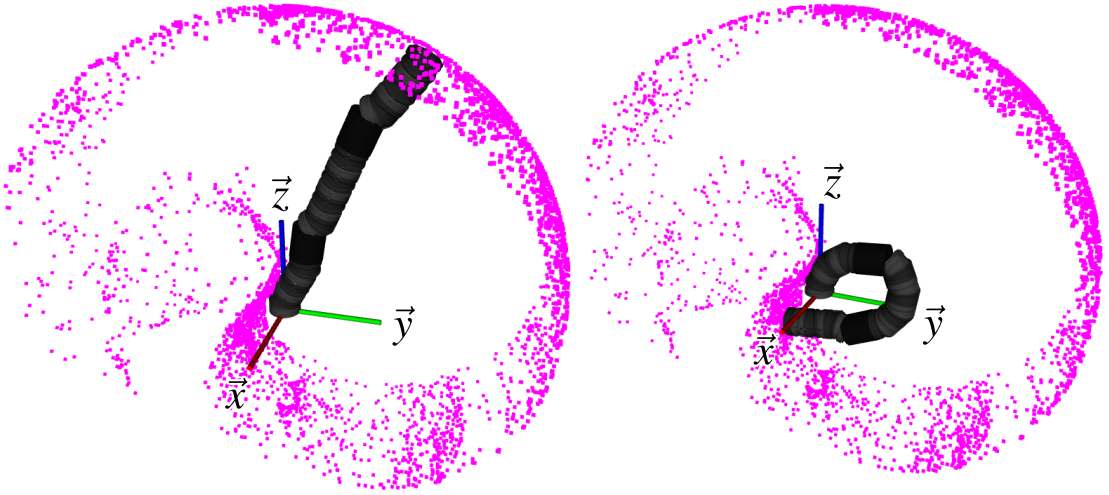
(a) Robot on external boundaries (b) Robot on internal boundaries

Figure 5.25: NB-R1 robot configuration on two collected points by the workspace determination algorithm. Obtained workspace points highlighted in magenta.



(a) Robot on external boundaries (b) Robot on internal boundaries

Figure 5.26: NB-R2 robot configuration on two collected points by the workspace determination algorithm. Obtained workspace points highlighted in magenta.



(a) Robot on external boundaries (b) Robot on internal boundaries

Figure 5.27: NB-R3 robot configuration on two collected points by the workspace determination algorithm. Obtained workspace points highlighted in magenta.

robot workspaces. However, the inner boundary detection can lead to errors when the number of NB-modules increases. Figures 5.28 and 5.29 show two Nimbl'Bot robot composed of 12 and 14 NB-modules, respectively. No link or offset is inserted between the NB-modules. These robots can touch their base. Figure 5.30 shows the vertical sections in yz-plane of the generated workspace points for the 12 NB-module robots. The workspace of the 12 NB-module robot has some internal boundaries that are bounded between $y \simeq [-0.2, 0.2]$ m and $z \simeq [0, 0.5]$ m, shown in Fig. 5.30. However, some other points are collected above and below the internal boundaries inside the workspace. These points are not part of the workspace boundaries. However, the robot NB-modules reached their limits when moving towards

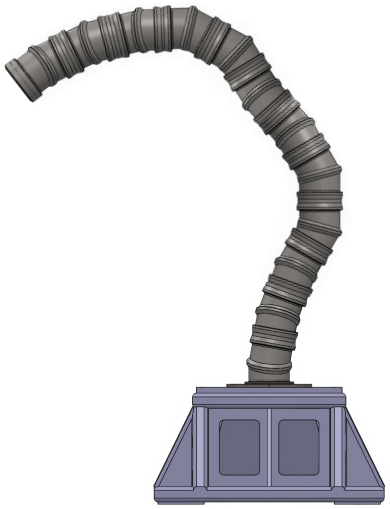


Figure 5.28: Nimbl'Bot robot composed of 12 NB-modules

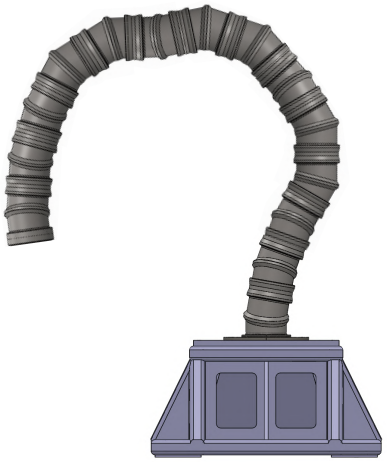


Figure 5.29: Nimbl'Bot robot composed of 14 NB-modules

these points. So, the robot was in a configuration that could not be escaped and the end-effector positions were collected as part of the boundaries. A clearer view of this behavior is shown in Fig. 5.31 for the 14 NB-module robot. It has no inner boundaries. However, several points were collected by the algorithm.

Figure 5.32 shows the 12 NB-module robot configuration to reach one of the points collected inside the workspace that are not part of the actual boundaries. The first modules of the robot reach the maximum allowed tilt and the robot can no longer move along the desired direction or escape from the singular configuration. However, the robot could reach the same end-effector linear position with another configuration and move toward it. The same behavior can be noticed for the 14 NB-module robot, as shown in Fig. 5.33. Similarly, the robot is stuck on an internal position that does not belong to the workspace boundaries. The limitation explained

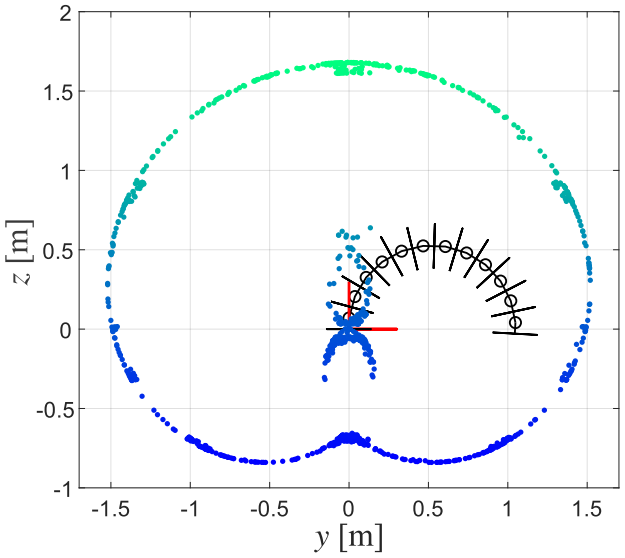


Figure 5.30: Vertical section in yz-plane of points describing workspace of 12 NB-module robot

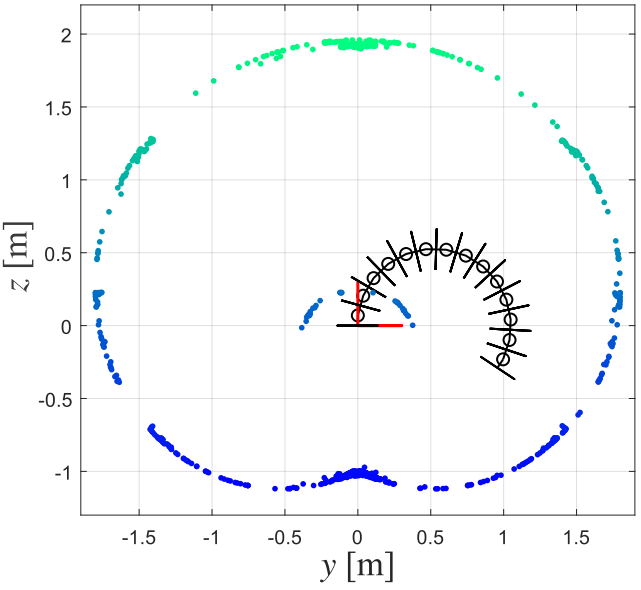


Figure 5.31: Vertical section in yz-plane of points describing workspace of 14 NB-module robot

in this section can generate problems in the correct identification of the manipulator workspaces. Further studies on this limitation should be done to improve the workspace determination process.

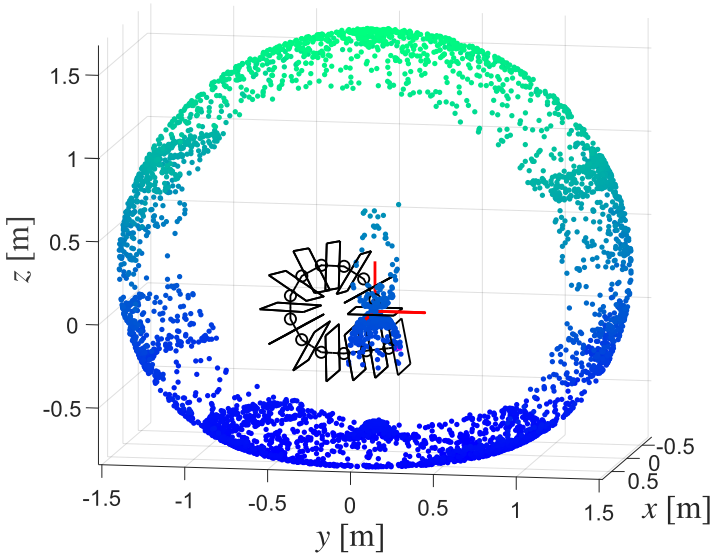


Figure 5.32: Configuration of 12 NB-module robot on point inside the workspace which does not belong to the boundaries

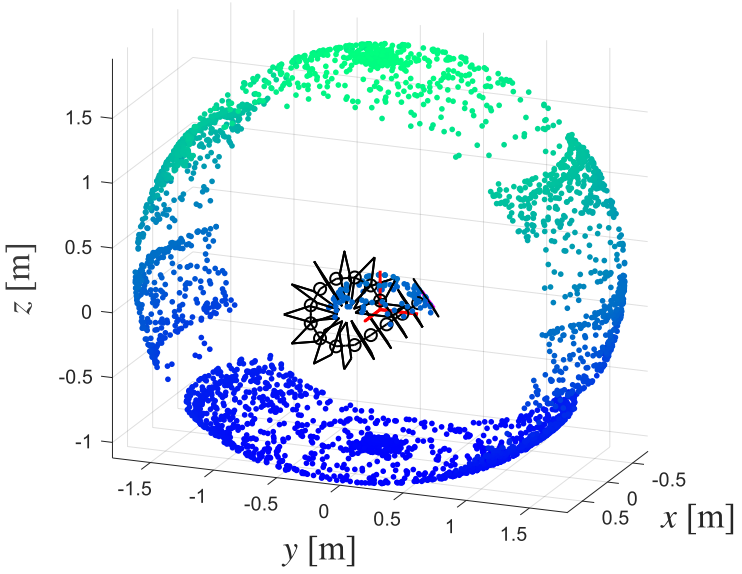


Figure 5.33: Configuration of 14 NB-module robot on point inside the workspace which does not belong to the boundaries

5.7 Workspace determination conclusions

This chapter presented a new algorithm for the workspace determination of robotic manipulators. The workspace determination process was evaluated on three kinematic redundant robots. However, it can also be applied to non-redundant manipu-

lators. This process employs the TPIK algorithm and kinematic optimization tasks, namely dexterity and manipulability, for the workspace determination. It is not affected by computational redundancy, like the Monte Carlo based methods, and identifies only the workspace boundaries. The performed tests emphasized the process ability to detect the complete workspace boundaries in a small amount of time. It always took less than ten minutes to produce one workspace. The process lasted less than one minute for the smaller robot, e.g. NB-R2. Moreover, using the kinematic optimization tasks allowed for maintaining better kinematic configurations while moving and prevented the manipulator from ending in singular configurations inside the workspace. In the NB-R3 robot case, the generated map identifies the workspace inner boundaries with less accuracy. However, the workspace shape is perfectly identifiable from the obtained results. The optimized ray-based method results are compared with the ones obtained through pseudo-inverse Jacobian ray-based and Monte Carlo methods. The proposed optimized ray-based workspace determination algorithm presents more accurate results than the pseudo-inverse Jacobian ray-based method. It is also faster, more accurate and requires fewer points than the Monte Carlo one. Then, a method for interpolating the points of the NB-R1 and NB-R2 workspaces was proposed. The method can generate the surface for symmetrical workspaces. However, this process can not plot the surface for the NB-R3 workspace. The surface interpolation will be helpful to develop an algorithm to compute the reachability of a 3D point automatically. In fact, this is a complex task to automatize, although it is visually easy. Moreover, the surface generation will allow the computation of the area and volume of the workspace. Finally, two main limitation of the proposed algorithm are discussed. First, there is no prior knowledge about the analyzed robot workspaces. So, it is not possible to determine the result correctness. However, a process was explained to check the obtained results. The second limitation appears when the number of NB-modules that compose the design grows. Over 12 NB-modules, the workspace inner boundaries presents some errors because the robot remained stuck in some internal configurations. Part of the work presented in this chapter was published in [GCSL23b].

Chapter 6

Conclusions

Kinematic redundant manipulators are increasingly used in manufacturing industry, providing important advantages. In fact, there can be multiple possible solutions for the same task since the robot is kinematically redundant. They can work in cluttered environments with many obstacles or collaborate with humans and other robots. Moreover, redundancy can be used to solve tasks simultaneously, such as optimizing different performance indices while performing the main task. However, it is important to recognize that this kinematic redundancy also introduces complexities. The main one is the identification of an algorithm to solve the inverse kinematic model problem. Moreover, analyzing the robot features, like the workspace, and optimizing its design become more complex tasks in the presence of kinematic redundancy. This chapter gives a synopsis of the manuscript content, summarizing the main topics treated during the research, and proposes the future work for each subject.

6.1 Thesis synopsis

The work done in this thesis involved three main topics that are linked by two common threads. These topics are related to the problems that arise with kinematic redundant robots, especially when applied to machining applications. So, the first common thread is the use of kinematic redundant manipulators in trajectory-tracking applications. The literature on these robots is revised in the introduction and their main complexities are highlighted. In the manuscript, the kinematic redundant robot composed of the actuation mechanism developed by Nimbl'Bot is employed as example for the performed tests. This actuation mechanism is described and its geometric and kinematic models are exposed. Then, a deeper analysis of the

relationship between the mechanism design parameter and its kinematic performance is proposed. The second common thread is the use of an algorithm for the kinematic redundancy resolution to identify the robot configuration for a specific task while optimizing some performance indices. In this research, a task priority based kinematic redundancy resolution algorithm was employed to kinematically control the Nimbl'Bot manipulator. It is called task priority inverse kinematic (TPIK) and exploits the robot redundancy to solve a list of prioritized tasks simultaneously.

The three main subjects that make up the thesis are briefly described below. First, the TPIK algorithm is used to kinematically control the redundant Nimbl'Bot robot following a set of desired trajectories. Since this kinematic control algorithm takes advantage of the redundancy solving simultaneous tasks, three new tasks are introduced. These tasks are based on the kinetostatic performance of the robot, namely dexterity, manipulability and robot transmission ratio (RTR). These tasks are used for the kinetostatic optimization of the robot configuration. In the conducted tests, the robot tracks different given trajectories with and without the optimization tasks. These tests are repeated several times to avoid the attractions to local maxima. The results show a better kinetostatic behavior when the optimization tasks are active. Moreover, the time difference between using or not the optimization tasks is negligible. A new index ϵ is defined as a linear combination of the kinetostatic indices and used to rate the test performance. The value of ϵ is on average 50% higher when reaching the trajectory starting poses activating the optimization tasks. The mean value of ϵ is on average 22% higher along the trajectories when the optimization tasks are active. In general, the kinetostatic performance directly relates to the trajectory placement and the velocity and force vector orientations. Finally, some studies on the linear kinematic Jacobian matrix singular vectors show that the movement effectiveness along the direction $\vec{y}$ is reduced compared to the other axes.

The second topic proposes a method for the design optimization of kinematic redundant robots. Here, the TPIK algorithm is employed to create this novel approach and optimize the design with respect to the desired applications and the kinetostatic indices dexterity, manipulability and RTR. This process considers the optimized design parameters as controllable virtual joints and updates their values while moving the robot. Initially, a simpler case is used to validate the new approach. Only two design variables and two paths are used and the results are compared with a discretized method. The two methods converged to equivalent results, obtaining the same performance, but the new process proposed here requires six time less the total computational time. In fact, the proposed optimization process can identify the best

solution for a specific application without testing all the possible design parameter combinations. Later, the proposed method is applied to a case study with more design parameters and trajectories. Several possible Nimbl'Bot robot designs are identified with high kinetostatic performance in the desired workspace area. The algorithm is tested with and without a task based on the center of mass to compare the ending robot dimension difference. When the center of mass task is not used, the resulting designs have high kinetostatic performance but too large dimensions for a real application. When the center of mass task is employed, the obtained designs are smaller, generating some realizable manipulators, but the kinetostatic performance is lower than in the previous case. In conclusion, this design optimization algorithm gives the guidelines for building a performant redundant robot based on a desired application and some performance metrics. Of course, some rate parameters can be antagonistic or not comparable to each other, like the kinetostatic indices and the center of mass distance. So, the user will have to use the guidelines to choose the most suitable result, doing a trade-off between all the application requirements.

The third topic is related to the kinematic redundant robot workspace determination and analysis. The workspace determination process is developed and evaluated on some kinematic redundant designs and employs a ray-based concept. However, it can also be applied to non-redundant manipulators. This process employs the TPIK algorithm and kinematic optimization tasks, namely dexterity and manipulability, for the workspace determination. The goal is to identify the workspace boundaries in a shorter period of time, avoiding the problems that Monte Carlo based methods have facing with. The tests are performed on three Nimbl'Bot designs and the results give a clear map of the workspaces. Each test takes less than ten minutes to produce one workspace and less than one minute for the smaller robot. In the case of robots with non-symmetrical workspaces, the workspace inner boundary profiles are less accurate than in the cases of symmetrical workspaces. Nevertheless, the resulting workspace shape can still be accurately determined based on the outcomes. The optimized ray-based method is compared two other methods, pseudo-inverse Jacobian ray-based and Monte Carlo. The proposed algorithm presents more accurate results than the pseudo-inverse Jacobian ray-based method. It is also faster, more accurate and requires fewer points than the Monte Carlo one. Then, a new point interpolating method for the symmetrical workspaces is introduced to plot the workspace surfaces. The surface generation is helpful to understand more features of the workspace, like calculating its volume or computing the reachability of a 3D point. Two main limitation of the proposed algorithm are discussed. First, there is no prior knowledge about the analyzed robot workspaces. So, it is not possible to determine the result

correctness. However, a process was explained to check the obtained results. The second limitation appears when the number of NB-modules that compose the design grows. Over 12 NB-modules, the workspace inner boundaries presents some errors because the robot remained stuck in some internal configurations.

All the proposed algorithms were tested on several different Nimbl'Bot robotic designs built through a serial connection of the NB-modules. However, these methods can be applied to any type of kinematic redundant manipulator. All the proposed algorithms give promising results and could be further fine-tuned to improve the resulting quality.

6.2 Future work

The work done in this thesis spreads in different fields and opens other subjects. This section presents the future work and analysis for each topic treated in this research.

1. **NB-module Modeling and Analysis:** Starting from Chapter 2, the NB-module geometric and kinematic models were described and analyzed. However, its dynamic model was not treated in this research. Similarly, the stiffness of the NB-module was not modeled. Future analyses should address these topics, which are relevant to demonstrate the abilities and limits of the robots actuated by NB-modules.

2. **Task Priority Kinematic Control:** The robot kinetostatic performance was tested by tracking some trajectories in Chapter 3. However, the proposed study uses only four squared trajectories with constant velocities and forces to test the robot performance. So, it is impossible to identify the best placement and orientation for the workpiece. This concept is the starting point for future analysis. Then, the performance of other existing manipulators should be compared with the Nimbl'Bot design one on the same tasks. This comparison should be made later on to present the Nimbl'Bot design potentials with respect to the counterparts. Moreover, the simulation tests could include some obstacles and self-collision avoidance tasks. These tasks allow the robot to work in more complex environments or with other robots. Finally, all the tests were performed in simulation. The TPIK algorithm should be tested on a real prototype of the Nimbl'Bot robot to demonstrate its abilities in real scenarios.

3. **Design Optimization Process:** In Chapter 4, the proposed design optimization process provided guidelines to build kinetostatic optimized Nimbl'Bot designs with respect to the desired machining application. Future work could involve further analysis to identify if more general guidelines exist for building a Nimbl'Bot robot design with high kinetostatic performance. For example, this could be done by giving more importance to one index than others or searching for specific relationships between the design parameters. Other design parameters could be included to study how they affect the kinetostatic performance. Moreover, obstacle and self-collision avoidance could be added to the optimization process to consider a more complex environment.

4. **Workspace Determination Algorithm:** Regarding the workspace determination algorithm proposed in Chapter 5, the process needs to be revised to improve the non-symmetrical workspace identification. Future work could also address the workspace determination for a specific end-effector orientation. This is important when planning the workpiece placement inside the robot reachable area. Then, a general process to interpolate the workspace points for any robot should be developed, starting from the one already proposed for symmetrical workspaces. So, it should be possible to plot the workspace surface for any Nimbl'Bot design and automatically compute whether a point is inside the workspace. Finally, some additional steps should be added to the workspace determination process to escape the inner boundary identification problem.

Bibliography

[ABR08] Gabriella Acaccia, Luca Bruzzone, and Roberto Razzoli. A modular robotic system for industrial applications. *Assembly Automation*, 28(2):151–162, 2008.

[ADJM13] Hatem Al-Dois, AK Jha, and RB Mishra. Task-based design optimization of serial robot manipulators. *Engineering Optimization*, 45(6):647–658, 2013.

[AEL19] Ghasem Abbasnejad, Jonathan Eden, and Darwin Lau. Generalized ray-based lattice generation and graph representation of wrench-closure workspace for arbitrary cable-driven robots. *IEEE Transactions on Robotics*, 35(1):147–161, 2019.

[ALC92] Jorge Angeles and Carlos S López-Cajún. Kinematic isotropy and the conditioning index of serial robotic manipulators. *The International Journal of Robotics Research*, 11(6):560–571, 1992.

[AM15] Hossein Ahmadzadeh and Ellips Masehian. Modular robotic systems: Methods and algorithms for abstraction, planning, control, and synchronization. *Artificial Intelligence*, 223:27–64, 2015.

[Ang92] Jorge Angeles. The design of isotropic manipulator architectures in the presence of redundancies. *The International Journal of Robotics Research*, 11(3):196–201, 1992.

[Ang03] Jorge Angeles. *Fundamentals of robotic mechanical systems: theory, methods, and algorithms*. Springer, 2003.

[APS19] Reem J Alattas, Sarosh Patel, and Tarek M Sobh. Evolutionary modular robotics: Survey and analysis. *Journal of Intelligent & Robotic Systems*, 95:815–828, 2019.

[BKRC15] Jessica Burgner-Kahrs, D Caleb Rucker, and Howie Choset. Continuum robots for medical applications: A survey. *IEEE Transactions on Robotics*, 31(6):1261–1280, 2015.

[BMR12] Oriol Bohigas, Montserrat Manubens, and Lluís Ros. A complete method for workspace boundary determination on general structure manipulators. *IEEE Transactions on Robotics*, 28(5):993–1006, 2012.

[Bra16] Mathias Brandstötter. *Adaptable serial manipulators in modular design.* PhD thesis, UMIT, Institute of Automation andControl Engineering, Hall in Tirol, Austria, 2016.

[BRS+17] Alberto Brunete, Avinash Ranganath, Sergio Segovia, Javier Perez De Frutos, Miguel Hernando, and Ernesto Gambao. Current trends in reconfigurable modular robots design. *International Journal of Advanced Robotic Systems*, 14(3):1729881417710457, 2017.

[BZG02] IA Bonev, D Zlatanov, and CM Gosselin. Advantages of the modified Euler angles in the design and control of PKMs. In *Proceedings of Parallel Kinematic Machines International Conference*, pages 171–188, Quebec, Canada, September 29–October 2 2002. Citeseer.

[BZL89] Beno Benhabib, G Zak, and MG Lipton. A generalized kinematic modeling method for modular robots. *Journal of robotic systems*, 6(5):545–571, 1989.

[Car09] Marco Carricato. Decoupled and homokinetic transmission of rotational motion via constant-velocity joints in closed-chain orientational manipulators. *Journal of Mechanisms and Robotics*, 1(4):041008, 2009.

[CB92] Gregory S Chirikjian and Joel W Burdick. A geometric approach to hyper-redundant manipulator obstacle avoidance. *Journal of Mechanical Design*, 114(4):580–585, 1992.

[CB93] Gregory S Chirikjian and Joel W Burdick. Design and experiments with a 30 dof robot. In *Proceedings of International Conference on Robotics and Automation*, pages 113–119, Atlanta, GA, May 02–06 1993. IEEE.

[CB94] Gregory S Chirikjian and Joel W Burdick. A hyper-redundant manipulator. *IEEE Robotics & Automation Magazine*, 1(4):22–29, 1994.

[CB95] Gregory S Chirikjian and Joel W Burdick. Kinematically optimal hyper-redundant manipulator configurations. *IEEE transactions on Robotics and Automation*, 11(6):794–806, 1995.

[CBH08] Alexandre Campos, Christoph Budde, and Jürgen Hesselbach. A type synthesis method for hybrid robot structures. *Mechanism and Machine Theory*, 43(8):984–995, 2008.

[CCRC18] Marco Ceccarelli, Daniele Cafolla, Matteo Russo, and Giuseppe Carbone. Heritagebot platform for service in cultural heritage frames. *International Journal of Advanced Robotic Systems*, 15(4):1729881418790692, 2018.

[CDGF13] Stéphane Caro, Claire Dumas, Sébastien Garnier, and Benoît Furet. Workpiece placement optimization for machining operations with a KUKA KR270-2 robot. In *Proceedings of International Conference on Robotics and Automation*, pages 2921–2926, Karlsruhe, Germany, May 6-10 2013. IEEE.

[CGSBK18] Mohamed Taha Chikhaoui, Josephine Granna, Julia Starke, and Jessica Burgner-Kahrs. Toward motion coordination control and design optimization for dual-arm concentric tube continuum robots. *IEEE Robotics and Automation Letters*, 3(3):1793–1800, 2018.

[Che08] Kai Cheng. *Machining dynamics: fundamentals, applications and practices*. Springer Science & Business Media, 2008.

[CL22] Hung Hon Cheng and Darwin Lau. Ray-based cable and obstacle interference-free workspace for cable-driven parallel robots. *Mechanism and Machine Theory*, 172:104782, 2022.

[CLV06] S-H Cha, TA Lasky, and SA Velinsky. Kinematic redundancy resolution for serial-parallel manipulators via local optimization including joint constraints. *Mechanics based design of structures and machines*, 34(2):213–239, 2006.

[COW08] Stefano Chiaverini, Giuseppe Oriolo, and Ian D. Walker. *Kinematically Redundant Manipulators*, pages 245–268. Springer Berlin Heidelberg, 2008.

[CWCC16] Daniele Cafolla, Mingfeng Wang, Giuseppe Carbone, and Marco Ceccarelli. Larmbot: a new humanoid robot with parallel mechanisms. In

ROMANSY 21-Robot Design, Dynamics and Control: Proceedings of the 21st CISM-IFToMM Symposium, pages 275–283. Springer, 2016.

[CY96] I-Ming Chen and Guilin Yang. Configuration independent kinematics for modular robots. In *Proceedings of International Conference on Robotics and Automation*, volume 2, pages 1440–1445, Minneapolis, MN, April 22–28 1996. IEEE.

[DBG+19] Dhaivat Dholakiya, Shounak Bhattacharya, Ajay Gunalan, Abhik Singla, Shalabh Bhatnagar, Bharadwaj Amrutur, Ashitava Ghosal, and Shishir Kolathaya. Design, development and experimental realization of a quadrupedal research platform: Stoch. In *Proceedings of 5th International Conference on Control, Automation and Robotics*, pages 229–234, Beijing, China, April 19–22 2019. IEEE.

[DCGF12] Claire Dumas, Stéphane Caro, Stéphane Garnier, and Benoît Furet. Workpiece Placement Optimization of Six-Revolute Industrial Serial Robots for Machining Operations. *Proceedings of 11th Biennial Conference on Engineering Systems Design and Analysis*, 2:419–428, July 2–4 2012.

[DDC13] Hui Dong, Zhijiang Du, and Gregory S Chirikjian. Workspace density and inverse kinematics for planar serial revolute manipulators. *Mechanism and Machine Theory*, 70:508–522, 2013.

[DDN19] Ayan Dutta, Prithviraj Dasgupta, and Carl Nelson. Distributed configuration formation with modular robots using (sub) graph isomorphism-based approach. *Autonomous Robots*, 43:837–857, 2019.

[DFDC15] Hui Dong, Taosha Fan, Zhijiang Du, and Gregory S Chirikjian. Inverse kinematics of discretely actuated ball-joint manipulators using workspace density. In *Proceedings of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 57144, page V05CT08A039, Boston, MA, August 2–5 2015. American Society of Mechanical Engineers.

[DK04] Etienne Dombre and Wisama Khalil. *Modeling, identification and control of robots*. Butterworth-Heinemann, (2004).

[DLCA19] Paolo Di Lillo, Stefano Chiaverini, and Gianluca Antonelli. Handling robot constraints within a set-based multi-task priority inverse

kinematics framework. In *Proceedings of International Conference on Robotics and Automation*, pages 7477–7483, Montreal, Canada, May 20–24 2019. IEEE.

[DOXY20] Zhao-cai Du, Guang-Yao Ouyang, Jun Xue, and Yan-bin Yao. A review on kinematic, workspace, trajectory planning and path planning of hyper-redundant manipulators. In *Proceedings of 10th Institute of Electrical and Electronics Engineers International Conference on Cyber Technology in Automation, Control, and Intelligent Systems*, pages 444–449, Xi'an, China, October 10–13 2020. IEEE.

[DP14] Vladimir Dukovski and Zoran Pandilov. Comparison of the characteristics between serial and parallel robots. *Acta Technica Corvininesis-Bulletin of Engineering*, 7(1), 2014.

[Duf21] Ludovic Dufau. Articulated robot arm. US patent 10,953,554 `https://uspto.report/patent/grant/10,953,554`, March 23, 2021. Online accessed 23 January 2023.

[EMW14] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028, 2014.

[EWO+14] Johannes Englsberger, Alexander Werner, Christian Ott, Bernd Henze, Maximo A Roa, Gianluca Garofalo, Robert Burger, Alexander Beyer, Oliver Eiberger, Korbinian Schmid, et al. Overview of the torque-controlled humanoid robot toro. In *Proceedings of International Conference on Humanoid Robots*, pages 916–923, Madrid, Spain, November 18–20 2014. IEEE-RAS.

[FDL13] Fabrizio Flacco and Alessandro De Luca. Optimal redundancy resolution with task scaling under hard bounds in the robot joint space. In *Proceedings of International Conference on Robotics and Automation*, pages 3969–3975, Karlsruhe, Germany, May 06–10 2013. IEEE.

[FDL14] Fabrizio Flacco and Alessandro De Luca. A reverse priority approach to multi-task control of redundant robots. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 2421–2427, Chicago, IL, September 14–18 2014. IEEE.

[FDLK12] Fabrizio Flacco, Alessandro De Luca, and Oussama Khatib. Prioritized multi-task motion control of redundant robots under hard joint constraints. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 3970–3977, Vilamoura-Algarve, Portugal, October 7–12 2012. IEEE.

[FK90] Toshio Fukuda and Yoshio Kawauchi. Cellular robotic system (cebot) as one of the realization of self-organizing intelligent universal manipulator. In *Proceedings of International Conference on Robotics and Automation*, pages 662–667, Cincinnati, OH, May 13–18 1990. IEEE.

[FN88] Toshio Fukuda and Seiya Nakagawa. Approach to the dynamically reconfigurable robotic system. *Journal of Intelligent and Robotic Systems*, 1:55–72, 1988.

[FSMVM+13] Alfonso Fernández-Sarría, Lucía Martínez, Borja Velázquez-Martí, Magdalena Sajdak, J Estornell, and JA Recio. Different methodologies for calculating crown volumes of platanus hispanica trees using terrestrial laser scanner and a comparison with classical dendrometric measurements. *Computers and electronics in agriculture*, 90:176–185, 2013.

[GA91] Clement Gosselin and Jorge Angeles. A global performance index for the kinematic optimization of robotic manipulators. *Journal of Mechanical Design*, 113(3):220–226, 1991.

[GCSL23a] Angelica Ginnante, Stéphane Caro, Enrico Simetti, and François Leborne. Kinetostatic optimization for kinematic redundancy planning of nimbl'bot robot. *Journal of Mechanisms and Robotics*, 16(3), 2023.

[GCSL23b] Angelica Ginnante, Stéphane Caro, Enrico Simetti, and François Leborne. Workspace determination of kinematic redundant manipulators using a ray-based method. In *Proceedings of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Boston, Massachusetts, August 20–23 2023. American Society of Mechanical Engineers.

[GKY18] Kevin G Gim, Joohyung Kim, and Katsu Yamane. Design and fabrication of a bipedal robot using serial-parallel hybrid leg mechanism.

In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 5095–5100, Madrid, Spain, October 1–5 2018. IEEE.

[GLC+21] Angelica Ginnante, François Leborne, Stéphane Caro, Enrico Simetti, and Giuseppe Casalino. Design and kinematic analysis of a novel 2-dof closed-loop mechanism for the actuation of machining robots. In *Proceedings of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 85444, Online, Virtual, August 17–19 2021. American Society of Mechanical Engineers.

[GP73] Gene H Golub and Victor Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on numerical analysis*, 10(2):413–432, 1973.

[GS18] Clément Gosselin and Louis-Thomas Schreiber. Redundancy in parallel mechanisms: A review. *Applied Mechanics Reviews*, 70(1):010802, 2018.

[GSCL23] Angelica Ginnante, Enrico Simetti, Stéphane Caro, and François Leborne. Task priority based design optimization of a kinematic redundant robot. *Mechanism and Machine Theory*, 187:105374, 2023.

[GST19] Bora Gonul, Omer Faruk Sapmaz, and Lutfi Taner Tunc. Improved stable conditions in robotic milling by kinematic redundancy. *Procedia CIRP*, 82:485–490, 2019.

[GW00] Ian A Gravagne and Ian D Walker. On the structure of minimum effort solutions with application to kinematic redundancy resolution. *IEEE Transactions on Robotics and Automation*, 16(6):855–863, 2000.

[GY06] Yisheng Guan and Kazuhito Yokoi. Reachable space generation of a humanoid robot using the monte carlo method. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 1984–1989, Beijing, China, October 09–15 2006. IEEE.

[HKC+17] Soonwoong Hwang, Hyeonguk Kim, Younsung Choi, Kyoosik Shin, and Changsoo Han. Design optimization method for 7 dof robot manipulator using performance indices. *International Journal of Precision Engineering and Manufacturing*, 18(3):293–299, 2017.

[HN91] Peter Hughes and Frank Naccarato. Inverse kinematics of variable geometry truss manipulators. *Journal of Robotic Systems*, 8(2):249–266, 1991.

[HPX+21] Yiheng Han, Jia Pan, Mengfei Xia, Long Zeng, and Yong-Jin Liu. Efficient se (3) reachability map generation via interplanar integration of intra-planar convolutions. In *Proceedings of International Conference on Robotics and Automation*, pages 1854–1860, Xi'an China, May 30–June 5 2021. IEEE.

[HW03] Michael W Hannan and Ian D Walker. Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots. *Journal of robotic systems*, 20(2):45–63, 2003.

[ISA15] I Iglesias, MA Sebastián, and JE Ares. Overview of the state of robotic machining: Current situation and future potential. *Procedia engineering*, 132:911–917, 2015.

[JLKH19] Dominic Jud, Philipp Leemann, Simon Kerscher, and Marco Hutter. Autonomous free-form trenching using a walking excavator. *IEEE Robotics and Automation Letters*, 4(4):3208–3215, 2019.

[JW19] Wei Ji and Lihui Wang. Industrial robotic machining: a review. *The International Journal of Advanced manufacturing Technology*, 103(1-4):1239–1255, 2019.

[JWT+11] Aaron Johnson, Cornell Wright, Matthew Tesch, Kevin Lipkin, and Howie Choset. A novel architecture for modular snake robots. *Citeseer, Tech. Rep.*, 2011.

[KA05] Waseem A Khan and Jorge Angeles. The kinetostatic optimization of robotic manipulators: The inverse and the direct problems. *Journal of Mechanical Design*, 128(1):168–178, 2005.

[KAW15] Suleman Khan, Kjell Andersson, and Jan Wikander. Jacobian matrix normalization - A comparison of different approaches in the context of multi-objective optimization of 6-dof haptic devices. *Journal of Intelligent & Robotic Systems*, 79(1):87–100, 2015.

[KK93] J-O Kim and Pradeep K Khosla. A formulation for task based design of robot manipulators. In *Proceedings of RSJ International Confer-*

ence on Intelligent Robots and Systems, volume 3, pages 2310–2317, Yokohama, Japan, July 26–30 1993. IEEE.

[KNH+19] Seong Hyeon Kim, Eunseok Nam, Tae In Ha, Soon-Hong Hwang, Jae Ho Lee, Soo-Hyun Park, and Byung-Kwon Min. Robotic machining: A review of recent progress. *International Journal of Precision Engineering and Manufacturing*, 20:1629–1642, 2019.

[KSP17] Sorada Khaengkam, Jiraphon Srisertpol, and Veerawuth Punlum. The application of double arms scara robot for deburring of pcb support plate. In *Proceedings of International Conference on Circuits, Devices and Systems*, pages 1–5, Chengdu, China, September 05–08 2017. IEEE.

[KSR+14] Virendra Kumar, Soumen Sen, Shibendu S Roy, Chandan Har, and SN Shome. Design optimization of serial link redundant manipulator: an approach using global performance metric. *Procedia Technology*, 14:43–50, 2014.

[KTV+90] James P Karlen, Jack M Thompson, Havard I Vold, James D Farrell, and Paul H Eismann. A dual-arm dexterous manipulator system with anthropomorphic kinematics. In *Proceedings of International Conference on Robotics and Automation*, pages 368–373, Cincinnati, OH, May 13–18 1990. IEEE.

[KWdGF+20] Shivesh Kumar, Hendrik Wöhrle, José de Gea Fernández, Andreas Müller, and Frank Kirchner. A survey on modularity and distributivity in series-parallel hybrid robots. *Mechatronics*, 68:102367, 2020.

[LBUP06] Sebastian Lohmeier, Thomas Buschmann, Heinz Ulbrich, and Friedrich Pfeiffer. Modular joint design for performance enhanced humanoid robot lola. In *Proceedings of International Conference on Robotics and Automation*, pages 88–93, Orlando, FL, May 15–19 2006. IEEE.

[LPSG13] Pål Liljebäck, Kristin Y Pettersen, Øyvind Stavdahl, and Jan Tommy Gravdahl. *Snake robots: modelling, mechatronics, and control*. Springer, 2013.

[LSFY18] Li Li, Junyun Shang, YL Feng, and Huai Yawen. Research of trajectory planning for articulated industrial robot: a review. *Computer engineering and applications*, 54(5):36–50, 2018.

[LUE90] Li Liu, BJ Ulrich, and Mohamed A Elbestawi. Robotic grinding force regulation: design, implementation and benefits. In *Proceedings of International Conference on Robotics and Automation*, pages 258–265, Cincinnati, OH, May 13–18 1990. IEEE.

[LXGC17] Yiming Liu, Hui Xu, Changxing Geng, and Guodong Chen. A modular manipulator for industrial applications: Design and implement. In *Proceedings of 2nd International Conference on Robotics and Automation Engineering*, pages 331–335, Shanghai, China, December 29–31 2017. IEEE.

[MD95] Hamid Reza Mohammadi Daniali. *Contributions to the kinematic synthesis of parallel manipulators*. PhD thesis, McGill University, Canada, 1995.

[MDA22] Omar W Maaroof, Mehmet İsmet Can Dede, and Levent Aydin. A robot arm design optimization method by using a kinematic redundancy resolution technique. *Robotics*, 11(1):1, 2022.

[Mer06] Jean-Pierre Merlet. *Parallel robots*, volume 128. Springer Science & Business Media, 2006.

[MKYC02] Giacomo Marani, Jinhyun Kim, Junku Yuh, and Wan Kyun Chung. A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators. In *Proceedings of International Conference on Robotics and Automation*, volume 2, pages 1973–1978, Washington, DC, May 11–15 2002. IEEE.

[MRG17] Midhun S Menon, VC Ravi, and Ashitava Ghosal. Trajectory planning and obstacle avoidance for hyper-redundant serial robots. *Journal of Mechanisms and Robotics*, 9(4), 2017.

[NCW18] Abhilash Nayak, Stéphane Caro, and Philippe Wenger. Comparison of 3-[pp] s parallel manipulators based on their singularity free orientation workspace, parasitic motions and complexity. *Mechanism and Machine Theory*, 129:293–315, 2018.

[NHY87] Yoshihiko Nakamura, Hideo Hanafusa, and Tsuneo Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(2):3–15, 1987.

[NPRRA02] J Norberto Pires, John Ramming, Stephen Rauch, and Ricardo Araújo. Force/torque sensing applied to industrial robotic deburring. *Sensor Review*, 22(3):232–241, 2002.

[ODAS15] Christian Ott, Alexander Dietrich, and Alin Albu-Schäffer. Prioritized multi-task compliance control of redundant manipulators. *Automatica*, 53:416–423, 2015.

[OHAH+20] Olatunji Mumini Omisore, Shipeng Han, Yousef Al-Handarish, Wenjing Du, Wenke Duan, Toluwanimi Oluwadara Akinyemi, and Lei Wang. Motion and trajectory constraints control modeling for flexible surgical robotic systems. *Micromachines*, 11(4):386, 2020.

[Par00] Jonghoon Park. Analysis and control of kinematically redundant manipulators: An approach based on kinematically decoupled joint space decomposition. *PhD Thesis, POSTECH*, 2000.

[PC06] Geoffrey Pond and Juan A Carretero. Formulating jacobian matrices for the dexterity analysis of parallel manipulators. *Mechanism and Machine Theory*, 41(12):1505–1519, 2006.

[PCY03] Joon-Young Park, Pyung-Hun Chang, and Jeong-Yean Yang. Task-oriented design of robot kinematics using the grid method. *Advanced robotics*, 17(9):879–907, 2003.

[PDSC11] John Pandremenos, Christos Doukas, Panagiotis Stavropoulos, and George Chryssolouris. Machining with robots: a critical review. pages 1–9, Athens, Greece, September 28–30 2011.

[Pet17] Kristin Y Pettersen. Snake robots. *Annual Reviews in Control*, 44:19–44, 2017.

[PK93] Christiaan JJ Paredis and Pradeep K Khosla. Kinematic design of serial link manipulators from task specifications. *The International Journal of Robotics Research*, 12(3):274–287, 1993.

[PRG+17] Adrián Peidró, Óscar Reinoso, Arturo Gil, José María Marín, and Luis Payá. An improved monte carlo method based on gaussian growth to calculate the workspace of robots. *Engineering Applications of Artificial Intelligence*, 64:197–207, 2017.

[PSVP13] Doina Pisla, Andras Szilaghyi, Calin Vaida, and Nicolae Plitea. Kinematics and workspace modeling of a new hybrid robot used in minimally invasive surgery. *Robotics and Computer-Integrated Manufacturing*, 29(2):463–474, 2013.

[RCC08] Novona Rakotomanga, Damien Chablat, and Stéphane Caro. Kinetostatic performance of a planar parallel mechanism with variable actuation. In *Advances in robot kinematics: Analysis and design*, pages 311–320. Springer, 2008.

[RMG16] Alexander Reiter, Andreas Müller, and Hubert Gattringer. Inverse kinematics in minimum-time trajectory planning for kinematically redundant manipulators. In *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 6873–6878, Florence, Italy, October 23–26 2016. IEEE.

[RSBT18] William S Rone, Wael Saab, and Pinhas Ben-Tzvi. Design, modeling, and integration of a flexible universal spatial robotic tail. *Journal of Mechanisms and Robotics*, 10(4):041001, 2018.

[RSH+15] Nicolaus A Radford, Philip Strawser, Kimberly Hambuchen, Joshua S Mehling, William K Verdeyen, A Stuart Donnan, James Holley, Jairo Sanchez, Vienny Nguyen, Lyndon Bridgwater, et al. Valkyrie: Nasa's first bipedal humanoid robot. *Journal of Field Robotics*, 32(3):397–419, 2015.

[SAM+17] Filippo Sanfilippo, Jon Azpiazu, Giancarlo Marafioti, Aksel A Transeth, Øyvind Stavdahl, and Pål Liljebäck. Perception-driven obstacle-aided locomotion for snake robots: the state of the art, challenges and possibilities. *Applied Sciences*, 7(4):336, 2017.

[SC16] Enrico Simetti and Giuseppe Casalino. A novel practical technique to integrate inequality control objectives and task transitions in priority based control. *Journal of Intelligent & Robotic Systems*, 84(1-4):877–902, 2016.

[SCWA18] Enrico Simetti, Giuseppe Casalino, Francesco Wanderlingh, and Michele Aicardi. Task priority control of underwater intervention systems: Theory and applications. *Ocean Engineering*, 164:40–54, 2018.

[SCWA19] Enrico Simetti, Giuseppe Casalino, Francesco Wanderlingh, and Michele Aicardi. A task priority approach to cooperative mobile manipulation: Theory and experiments. *Robotics and Autonomous Systems*, 122:103287, 2019.

[Sic90] Bruno Siciliano. Kinematic control of redundant robot manipulators: A tutorial. *Journal of intelligent and robotic systems*, 3(3):201–212, 1990.

[SK05] Luis Sentis and Oussama Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(04):505–518, 2005.

[SKK08] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Redundant robots*, volume 200. Springer, 2008.

[SNM+17] Steffen Schütz, Atabak Nejadfard, Krzysztof Mianowski, Patrick Vonwirth, and Karsten Berns. Carl—a compliant robotic leg featuring mono-and biarticular actuation. In *Proceedings of 17th International Conference on Humanoid Robotics*, pages 289–296, Birmingham, UK, November 15–17 2017. IEEE-RAS.

[SS91] B Siciliano and J-J E Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Proceeding of 5th International Conference on Advanced Robotics*, volume 2, pages 1211–1216, Seattle, WA, July 18–20 1991.

[SW95] Sanjeev Seereeram and John T Wen. A global approach to path planning for redundant manipulators. *IEEE Transactions on Robotics and Automation*, 11(1):152–160, 1995.

[SWBC03] Elie Shammas, Alon Wolf, H Ben Brown, and Howie Choset. New joint design for three-dimensional hyper redundant robots. In *Proceedings of RSJ International Conference on Intelligent Robots and Systems*, volume 4, pages 3594–3599, Las Vegas, NV, October 27–31 2003. IEEE.

[SWC06] Elie Shammas, Alon Wolf, and Howie Choset. Three degrees-of-freedom joint for spatial hyper-redundant robots. *Mechanism and machine theory*, 41(2):170–190, 2006.

[SY11] Lin Song and Suixian Yang. Research on modular design of perpendicular jointed industrial robots. In *Proceedings of Intelligent Robotics and Applications: 4th International Conference*, pages 63–72, Aachen, Germany, December 6–8 2011. Springer.

[Tan00] Tanio K Tanev. Kinematics of a hybrid (parallel–serial) robot manipulator. *Mechanism and Machine Theory*, 35(9):1183–1196, 2000.

[TGA93] Yoshimi Takeuchi, Dongfang Ge, and Naoki Asakawa. Automated polishing process with a human-like dexterous robot. In *Proceedings of International Conference on Robotics and Automation*, pages 950–956, Atlanta, GA, May 02–06 1993. IEEE.

[TGK99] HK Tönshoff, H Grendel, and R Kaak. Structure and characteristics of the hybrid manipulator georg v. In *Parallel Kinematic Machines: Theoretical Aspects and Industrial Requirements*, pages 365–376. Springer, 1999.

[WBB+12] Cornell Wright, Austin Buchan, Ben Brown, Jason Geist, Michael Schwerin, David Rollinson, Matthew Tesch, and Howie Choset. Design and architecture of the unified modular snake robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 4347–4354, Saint Paul, MN, çqy 14–18 2012. IEEE.

[WBC+03] Alon Wolf, H Benjamin Brown, Randy Casciola, Albert Costa, Michael Schwerin, E Shamas, and Howie Choset. A mobile hyper redundant mechanism for search and rescue tasks. In *Proceedings of RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2889–2895, Las Vegas, NV, October 27–31 2003. IEEE.

[WC04] Yunfeng Wang and Gregory S Chirikjian. Workspace generation of hyper-redundant manipulators as a diffusion process on se (n). *IEEE Transactions on Robotics and Automation*, 20(3):399–408, 2004.

[WC18] Julian Whitman and Howie Choset. Task-specific manipulator design and trajectory synthesis. *IEEE Robotics and Automation Letters*, 4(2):301–308, 2018.

[WJP+07] Cornell Wright, Aaron Johnson, Aaron Peck, Zachary McCord, Allison Naaktgeboren, Philip Gianfortoni, Manuel Gonzalez-Rivero, Ross

Hatton, and Howie Choset. Design of a modular snake robot. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 2609–2614, San Diego, CA, October 29–November 02 2007. IEEE.

[WWS+17] Patrick M Wensing, Albert Wang, Sangok Seok, David Otten, Jeffrey Lang, and Sangbae Kim. Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots. *Ieee transactions on robotics*, 33(3):509–522, 2017.

[YMM11] Samer Yahya, Mahmoud Moghavvemi, and Haider AF Mohamed. Geometrical approach of planar hyper-redundant manipulators: Inverse kinematics, path planning and workspace. *Simulation Modelling Practice and Theory*, 19(1):406–422, 2011.

[Yos85] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985.

[ZCL20] Zeqing Zhang, Hung Hon Cheng, and Darwin Lau. Efficient wrench-closure and interference-free conditions verification for cable-driven parallel robot trajectories using a ray-based method. *IEEE Robotics and Automation Letters*, 5(1):8–15, 2020.

[ZHZ+18] Zhiyuan Zhao, Shuai He, Yaping Zhao, Ce Xu, Qingwen Wu, and Zhenbang Xu. Workspace analysis for a 9-dof hyper-redundant manipulator based on an improved monte carlo method and voxel algorithm. In *Proceedings of International Conference on Mechatronics and Automation*, pages 637–642, Changchun, China, August 05–08 2018. IEEE.

[ZKA12] SHH Zargarbashi, Waseem Khan, and Jorge Angeles. Posture optimization in robot-assisted machining operations. *Mechanism and Machine Theory*, 51:74–86, 2012.

Journal and Conference Publications

- **Title:** Design and Kinematic Analysis of a Novel 2-DoF Closed-Loop Mechanism for the Actuation of Machining Robots
- **Conference:** ASME/IDETC-CIE 2021
- **Date:** August 17 – 20, 2021
- **Abstract:** The essential characteristics of machining robots are their stiffness and their accuracy. For machining tasks, serial robots have many advantages, like a larger workspace, but they lack the necessary stiffness to accomplish high machining effort tasks. One way to increase the stiffness of serial manipulators is to make their joints using closed-loop or parallel mechanisms instead of using the classical prismatic and revolute joints. This increases the accuracy of a manipulator without reducing its workspace. This paper introduces an innovative two degrees of freedom closed-loop mechanism and shows how this mechanism can be used to build high stiffness and large workspace serial robots. The design of this mechanism is described through its geometric and kinematic models. Then, the kinematic performance of the mechanism is analyzed, and a serial arrangement of several such mechanisms is proposed to obtain a potential design of a machining robot.
- **Reference:** Angelica Ginnante, François Leborne, Stéphane Caro, Enrico Simetti, and Giuseppe Casalino. Design and kinematic analysis of a novel 2-dof closed-loop mechanism for the actuation of machining robots. In *Proceedings of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 85444, Online, Virtual, August 17–19 2021. American Society of Mechanical.

- **Title:** Kinetostatic Optimization for Kinematic Redundancy Planning of Nimbl'Bot Robot
- **Journal:** Journal of Mechanism and Robotics (ASME)
- **Date:** March 27, 2023
- **Abstract:** In manufacturing industry, Computer Numerical Control (CNC) machines are often preferred over Industrial Serial Robots (ISR) for machining tasks. Indeed, CNC machines offer high positioning accuracy, which leads to slight dimensional deviation on the final product. However, these machines have a restricted workspace generating limitations in the machining work. Conversely, ISR are typically characterized by a larger workspace. ISR have already shown satisfactory performance in tasks like polishing, grinding and deburring. This paper proposes a kinematic redundant robot composed of a novel two degrees-of-freedom mechanism with a closed kinematic chain. After describing a task priority inverse kinematic control framework used for joint trajectory planning exploiting the robot kinematic redundancy, the paper analyses the kinetostatic performance of this robot depending on the considered control tasks. Moreover, two kinetostatic tasks are introduced and employed to improve the robot performance. Simulation results show how the robot better performs when the optimization tasks are active.
- **Reference:** Angelica Ginnante, Stéphane Caro, Enrico Simetti, and François Leborne. Kinetostatic optimization for kinematic redundancy planning of nimbl'bot robot. *Journal of Mechanisms and Robotics*, 16(3), 2023.

- **Title:** Task Priority Based Design Optimization of a Kinematic Redundant Robot
- **Journal:** Mechanism and Machine Theory
- **Date:** May 9, 2023
- **Abstract:** This paper presents and defines a new design optimization method for kinematic redundant robot manipulators based on their applications. Kinematic redundant manipulators can reach a pose with an infinite number of postures. So, identifying the best robot design and configuration for a set of desired tasks is a highly complex non-linear problem. This approach employs a task priority control algorithm to perform a task oriented robot design optimization. The design parameters are replaced by controllable prismatic or revolute virtual joints and controlled by the algorithm to accomplish the desired tasks. Therefore, this new method finds an optimal robot design for a set of tasks taking advantage of the robot kinematic redundancy. This method is evaluated on a highly kinematic redundant manipulator, which tracks a set of paths with its end-effector while maintaining good kinetostatic performance.
- **Reference:** Angelica Ginnante, Enrico Simetti, Stéphane Caro, and François Leborne. Task priority based design optimization of a kinematic redundant robot. *Mechanism and Machine Theory*, 187:105374, 2023.

- **Title:** Workspace Determination of Kinematic Redundant Manipulators Using a Ray-Based Method
- **Conference:** ASME/IDETC-CIE 2023
- **Date:** August 20 – 23, 2023
- **Abstract:** Determining the workspace of a robotic manipulator is extremely significant for knowing its abilities and planning the robot application. There exist several techniques for the robot workspace determination. However, these methods usually are affected by computational redundancy, like in the case of Monte Carlo based methods, or their implementation is difficult. Moreover, the workspace analysis of kinematic redundant manipulators is even more complex. This paper introduces a ray-based workspace determination algorithm, easy to implement and not affected by computational redundancy. The proposed method can be applied to any type of serial robot, but it is tested only on spatial kinematic redundant robots. The results show how the approach can clearly determine the boundary of the robot workspace in a short period of time. Finally the time and quality performance of the ray-based method results are compared to the Monte Carlo one demonstrating the improvement of the proposed method.
- **Reference:** Angelica Ginnante, Stéphane Caro, Enrico Simetti, and François Leborne. Workspace determination of kinematic redundant manipulators using a ray-based method. In *Proceedings of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Conference, Boston, Massachusetts, August 20–23 2023. American Society of Mechanical Engineers.

Titolo: Progettazione, analisi e controllo cinematico ottimizzati di bracci robotici seriali altamente ridondanti

Parole chiave: ridondanza cinematica, priorità dei task, prestazioni cinetostatiche, ottimizzazione del design, analisi dello spazio di lavoro

Riassunto: L'uso di manipolatori robotici nell'industria è cresciuto negli ultimi decenni per migliorare e velocizzare i processi industriali. I manipolatori industriali hanno iniziato a essere studiati per le attività di lavorazione, poiché possono coprire spazi di lavoro più ampi, aumentando la gamma di operazioni realizzabili e migliorando la flessibilità. L'azienda Nimbl'Bot ha sviluppato un nuovo meccanismo, o modulo, per costruire robot modulari seriali più rigidi e flessibili per applicazioni di lavorazione. Questo manipolatore è un robot cinematico ridondante con 21 gradi di libertà. Questa tesi analizza a fondo le caratteristiche del robot Nimbl'Bot ed è suddivisa in tre argomenti principali. Il primo argomento riguarda l'utilizzo di un algoritmo di risoluzione della ridondanza cinematica a priorità di compito per la traiettoria di inseguimento del robot Nimbl'Bot, ottimizzando le sue prestazioni cinetostatiche. Il secondo argomento è l'ottimizzazione del design del robot con ridondanza cinematica rispetto a un'applicazione desiderata e alle sue prestazioni cinetostatiche. Per il terzo argomento, viene proposto un nuovo algoritmo di determinazione dello spazio di lavoro per manipolatori cinematici ridondanti. Per ogni argomento vengono proposti e testati diversi esperimenti in simulazione con alcuni design di robot Nimbl'Bot.

Titre: Conception, analyse et contrôle cinématique optimisées de bras robotiques sériel hautement redondants

Mots clés: redondance cinématique, priorité des tâches, performance cinétostatique, optimisation de la conception, analyse de l'espace de travail

Résumé: L'utilisation de manipulateurs robotiques dans l'industrie s'est développée au cours des dernières décennies afin d'améliorer et d'accélérer les processus industriels. Les manipulateurs industriels ont commencé à être étudiés pour les tâches d'usinage car ils peuvent couvrir de plus grands espaces de travail, ce qui augmente la gamme d'opérations réalisables et améliore la flexibilité. La société Nimbl'Bot a mis au point un nouveau mécanisme, ou module, pour construire des robots modulaires en série plus rigides et plus flexibles pour les applications d'usinage. Ce manipulateur est un robot redondant cinématique à 21 degrés de liberté. Cette thèse analyse en profondeur les caractéristiques du robot Nimbl'Bot et est divisée en trois sujets principaux. Le premier sujet concerne l'utilisation d'un algorithme de résolution de redondance cinématique prioritaire pour la trajectoire de suivi du robot Nimbl'Bot tout en optimisant ses performances cinétostatiques. Le deuxième sujet est l'optimisation de la conception d'un robot à redondance cinématique en fonction d'une application souhaitée et de ses performances cinétostatiques. Pour le troisième sujet, un nouvel algorithme de détermination de l'espace de travail est proposé pour les manipulateurs redondants cinématiques. Plusieurs tests de simulation sont proposés et testés sur quelques conceptions de robots Nimbl'Bot pour chaque sujet.

Title: Design, analysis and kinematic control of highly redundant serial robotic arms

Keywords: kinematic redundancy, task priority, kinetostatic performance, design optimization, workspace analysis

Abstract: The use of robotic manipulators in industry has grown in the last decades to improve and speed up industrial processes. Industrial manipulators started to be investigated for machining tasks since they can cover larger workspaces, increasing the range of achievable operations and improving flexibility. The company Nimbl'Bot developed a new mechanism, or module, to build stiffer flexible serial modular robots for machining applications. This manipulator is a kinematic redundant robot with 21 degrees of freedom. This thesis thoroughly analysis the Nimbl'Bot robot features and is divided into three main topics. The first topic regards using a task priority kinematic redundancy resolution algorithm for the Nimbl'Bot robot tracking trajectory while optimizing its kinetostatic performances. The second topic is the kinematic redundant robot design optimization with respect to a desired application and its kinetostatic performance. For the third topic, a new workspace determination algorithm is proposed for kinematic redundant manipulators. Several simulation tests are proposed and tested on some Nimbl'Bot robot designs for each subjects.