Optimized Ray-Based Method for Workspace Determination of Kinematic Redundant Manipulators

Angelica Ginnante, Stéphane Caro, Enrico Simetti, François Leborne

▶ **To cite this version:**

Angelica Ginnante, Stéphane Caro, Enrico Simetti, François Leborne. Optimized Ray-Based Method for Workspace Determination of Kinematic Redundant Manipulators. Journal of Mechanisms and Robotics, 2024, 16 (11), pp.111002. 10.1115/1.4065071 . hal-04670923

HAL Id: hal-04670923
https://hal.science/hal-04670923v1

Submitted on 13 Aug 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Optimized Ray-Based Method for Workspace Determination of Kinematic Redundant Manipulators

Angelica Ginnante[1,2,3], **Stéphane Caro**[1], **Enrico Simetti**[2],
François Leborne[3]
[1]Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, 1 Rue de la Noë, 44321 Nantes, France
[2]University of Genova, DIBRIS, 16145 Genova, Italy
[3]Nimbl'Bot, 7 Avenue de Guitayne, 33610 Canéjan, France
Email: aginnante@nimbl-bot.com, Stephane.Caro@ls2n.fr, Enrico.Simetti@unige.it, fleborne@nimbl-bot.com

ABSTRACT

Determining the workspace of a robotic manipulator is highly significant for knowing its abilities and planning the robot application. Several techniques exist for robot workspace determination. However, these methods are usually affected by computational redundancy, like in the Monte Carlo based method case, and their implementation can be complex. The workspace analysis of kinematic redundant manipulators is even more complex. This paper proposes a kinematically optimized ray-based workspace determination algorithm based on a simple idea and not affected by computational redundancy. The proposed method can be applied to any serial robot but is tested only on spatial kinematic redundant robots. The results show how the approach can correctly determine the robot workspace boundaries in a short time. Then, the correctness and computational time of the proposed optimized ray-based method are compared to pseudo-inverse Jacobian ray-based and Monte Carlo methods. The comparison demonstrates that the proposed method has better results in a shorter time. Finally, some limitations of the proposed algorithm are discussed.

1 Introduction

The workspace of a manipulator is the set of positions and orientations reachable by the robot end-effector [1]. 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 [2]. Three main types of methods exist for the workspace determination of kinematic redundant robots, as described in [2]. The first one is the geometric type, mainly applied for the workspace identification of kinematic redundant planar robots [3]. 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 [4]. 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 approach can be too complex when applied to kinematic redundant robots [5]. The third approach is the numerical one, which identifies the workspace boundaries using the robot forward kinematics. This method can be applied to any robot, redundant or not, and its solution is easily understandable [2]. 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 [6]. The Monte Carlo method generates many random robot configurations to determine the robot workspace. However, this method has several drawbacks [2, 5]. The generated workspace can be inaccurate, especially on its boundaries [2], and the real workspace can be different from the one obtained [5]. 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 [7]. 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 point areas usually corresponds to the boundaries of the workspace [7]. On the contrary, workspace inner areas are characterized by a high density of points, leading to high computational time waste.

In [5], the authors employed an improved version of the Monte Carlo method to calculate the robot workspace, called Gaussian Growth. The method consists of generating a number 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. The algorithm takes the robot configurations for one position in the inaccurate area to populate poorly defined workspace regions. Then, the process slightly modifies this configuration through a multivariate normal distribution to identify new configurations. So, new workspace points are identified and the poorly defined regions are improved. In [8], the authors presented another Monte Carlo method combined with the Gaussian distribution plus a Voxel algorithm [9] 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 workspace generation algorithms employ the workspace density and the N-dimensional Euclidean motion group $SE(N)$. In [10], 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 for planar serial arms with revolute joints, as shown in [11]. 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 [12]. The approach described in [13] 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.

There exist other interesting methods based on a point-cloud representation of robot workspaces. The point-cloud representation is useful to construct network graphs, leveraging them for real-time motion planning, particularly in instances involving non-convex workspace shapes. In [14], the authors presents a new sampling strategy, Workspace Importance Sampling (WIS), to tackle narrow passages in probabilistic roadmap (PRM) planning. By tetrahedralizing the workspace and using geometric information as heuristic values, WIS guides sampling in the configuration space. The goal is to increase the likelihood of sampling free configurations in narrow passages, vital for capturing free space connectivity, while reducing sampling in open collision-free areas. WIS draws inspiration from importance sampling in Monte Carlo integration. Another work [15] presents a novel path planning method that integrates numerical algebraic geometry to compute geometric features and distances. This deterministic planner offers stronger guarantees than resolution completeness, automatically determining the appropriate resolution for a given input space. The output is a weighted graph representing the configuration space, with nodes as points and edges weighted by ambient space distance. Despite being computationally demanding for a global graph-based model, its rapid feature size and distance computations can be integrated with other proposals. The resulting graph models efficiently compute accurate shortest paths, supporting repeated calculations. However, this paper does not deal with the problematic of motion planning in robot workspaces. The goal behind the proposed method is to show the reachable workspace for a robot end-effector simplifying the computations compared to what is required for motion planning. Moreover, these techniques are not applied on complex kinematic redundant design.

The workspace determination algorithm described here is an optimized ray-based method. The ray-based method idea is presented in [16, 17, 18] 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 [18]. Moreover, ray-based methods decrease the computational time with respect to the other approaches [16]. The proposed workspace determination method considers more complex robotic architectures, particularly 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 quickly with continuous boundaries. In fact, the goal of the proposed method is to provide as output the workspace shape fully described by its 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. For the moment, the proposed method is developed to identify the workspace reachable by robot the end-effector for any orientation. The robots that test the proposed workspace determination algorithm are highly kinematic redundant [19]. They are perfect candidates as describing their workspace is a complex problem [2].

The paper is outlined as follows. Section 2 introduces the background of the algorithm, namely the kinematic control algorithm used to control the robot and the employed optimization tasks. Section 3 describes the workspace determination method proposed in this paper. Section 4 presents the family of kinematic redundant robots employed to test the new method. Section 5 introduces the preliminary steps performed to initialize the tests. Section 6 analyzes the obtained results testing the proposed workspace determination algorithm and compares the performance with other methods. Section 7 highlights and discusses some limitations of the proposed approach. Conclusions and future work are given in section 8.

2 Background

This section describes the kinematic control algorithm and optimization tasks employed by the workspace determination algorithm. The chosen kinematic control algorithm is developed to exploit the redundancy of robotic machines and solve several tasks simultaneously. Firstly, the kinematic control algorithm is introduced. Then, two kinematic optimization tasks are described. These tasks prevent the robot from reaching singular configurations where the end-effector is not at the boundary of the workspace.

2.1 Kinematic Control Algorithm

The employed kinematic control algorithm is called Task Priority Inverse Kinematic (TPIK) and is presented in [20, 21, 22, 23]. It can find the robot configuration that solves a set of tasks with different priority levels. Moreover, it can activate and deactivate one or more tasks without generating algorithmic discontinuities [20] to avoid over-constraining the robotic system.

Here, some general definitions are given to describe the TPIK algorithm briefly. This paper deals with serial robotic manipulators, so the vector named $\mathbf{q} \in \mathbb{R}^n$ represents the joint position vector that describes the arm configuration. The variable n is the number of joints contained in the robot. The joint velocities are grouped in the vector $\dot{\mathbf{q}} \in \mathbb{R}^n$. A control objective is defined to represent one of the robot goals and the associated task state. It corresponds to a scalar variable $x(\mathbf{q})$ computed as a function of the robot configuration $\mathbf{q}$. Two types of control objectives exist equality and inequality. More details are given in [23]. Each control objective is associated with a feedback reference rate $\dot{\bar{x}}$ that drives $x(\mathbf{q})$ to the desired point x^* with the linked rate $\dot{x}^*$. Each control objective $x(\mathbf{q})$ has an activation function $a^i(x) \in [0,1]$, which states if the control objective is applicable or not at a specific time instant. Finally, a priority level is assigned to each task based on its control objective importance. The tasks with the highest priorities are solved first by employing the necessary robot degrees of freedom. Then, lower priority tasks are solved if enough degrees of freedom remain. These concepts were fully described in [23].

With the previous definitions, the control action $\mathscr{A}$ taken as input by the TPIK algorithm can be defined as a series of priority levels composed of [23]:

$\dot{\bar{\mathbf{x}}}_k = [\dot{\bar{x}}_{1,k},\ \dots\ ,\ \dot{\bar{x}}_{m_k,k}]^\top$ is the vector of all the scalar control objective reference rates, where m_k is the number of control objectives for the k^{th} priority level.
$\mathbf{J}_k$ is the Jacobian matrix associated with the vector $[\dot{x}_{1,k},\ \dots\ ,\ \dot{x}_{m_k,k}]^\top$ with respect to the joint velocities $\dot{\mathbf{q}}$ for the k^{th} priority level.
$\mathbf{A}_k = \text{diag}(a_{1,k},\ \dots\ ,\ a_{m_k,k})$ is the activation function diagonal matrix for the k^{th} priority level.

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

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

where S_{k-1} is the manifold of all the previous priority level solutions. The solution to the k^{th} priority level is searched in the manifold of all the previous priority level solutions. The notation $\text{R}-\min$ highlights that each minimization is performed through specific regularized space projections to implement priorities among the tasks defined in [20]. Moreover, the TPIK algorithm uses regularized space projection to implement priorities among all the tasks. One of the main advantages of the TPIK algorithm is the use of the activation functions for inequality control objectives to avoid blocking degrees of freedom that lower priority level tasks could use. The complete explanation of the TPIK algorithm solution mechanism can be found in [20].

2.2 Optimization tasks

The TPIK algorithm employs a task related to the robot end-effector velocity to perform the optimized ray-based workspace determination. However, the TPIK algorithm also includes two kinematic optimization tasks to avoid blocking in singular configurations. These tasks are based on the kinematic indices: dexterity and manipulability. These indices are related to the kinematic Jacobian matrix $\mathbf{J}_e$ for the robot end-effector velocity

$$\mathbf{t} = \begin{bmatrix}\dot{\mathbf{p}}\\ \boldsymbol{\omega}\end{bmatrix} = \mathbf{J}_e(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix}\mathbf{J}_l(\mathbf{q})\\ \mathbf{J}_a(\mathbf{q})\end{bmatrix}\dot{\mathbf{q}}, \tag{2}$$

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, i.e., linear and angular ones, it must be weighted before computing the kinematic performance indices. This weighing uses the characteristic length L introduced in [24] to solve the absence of dimensional homogeneity. The computation of L is proposed in [25]. Then, the rows associated with the linear velocities of the end-effector in $\mathbf{J}_e$ are divided by L for the revolute joints. The weighted kinematic Jacobian matrix is written as $\mathbf{J}_w$.

Dexterity

The dexterity $\eta(\mathbf{J}_w)$ characterizes the kinematic performance of a manipulator in a given configuration [26]. In [23], the analytical expression of η is computed using the Frobenius norm of $\mathbf{J}_w$

$$\eta(\mathbf{J}_w) = \frac{m}{\gamma_1(\mathbf{J}_w)\,\gamma_2(\mathbf{J}_w)}, \tag{3}$$

where m, which represents the number of rows of $\mathbf{J}_w$, is the dimension of the task space and

$$\gamma_1 = \sqrt{\text{trace}(\mathbf{J}_w\mathbf{J}_w^\top)} \text{ and } \gamma_2 = \sqrt{\text{trace}[(\mathbf{J}_w\mathbf{J}_w^\top)^{-1}]}. \tag{4}$$

The index η is bounded between 0 and 1. The higher η, the better the robot dexterity. The robot reaches an isotropic posture when $\eta = 1$. The smaller η, the worse the robot dexterity and the closer to a singularity. Moreover, η can be defined as the ratio between the smallest and the highest singular values of $\mathbf{J}_w$ indicating how close the manipulability hyper-ellipsoid is to being a hyper-sphere [27]. The derivative of the skill as a function of the joint variables is described in [23],

$$\frac{\partial \eta}{\partial q_i} = -\eta \left(\frac{\partial \gamma_1}{\partial q_i} \frac{1}{\gamma_1} + \frac{1}{\gamma_2} \frac{\partial \gamma_2}{\partial q_i} \right), \tag{5}$$

where

$$\begin{cases} \frac{\partial \gamma_1}{\partial q_i} = \frac{1}{\gamma_1} \text{trace} \left\{ \mathbf{J}_w \frac{\partial \mathbf{J}_w^\top}{\partial q_i} \right\}, \\ \frac{\partial \gamma_2}{\partial q_i} = \frac{1}{\gamma_2} \text{trace} \left\{ -\mathbf{J}_w \frac{\partial \mathbf{J}_w^\top}{\partial q_i} (\mathbf{J}_w \mathbf{J}_w^\top)^2 \right\}. \end{cases} \tag{6}$$

The dexterity Jacobian matrix $\mathbf{J}_\eta$ as a function of the joint variables is

$$\mathbf{J}_\eta = \left[\frac{\partial \eta}{\partial q_1} \ \cdots \ \frac{\partial \eta}{\partial q_n} \right]. \tag{7}$$

Manipulability

The manipulability is an index that measures the kinematic abilities of the robotic system through its weighted Jacobian matrix $\mathbf{J}_w$ [28]. The manipulability of a manipulator is defined as

$$\mu = \sqrt{\det(\mathbf{J}_w \mathbf{J}_w^\top)}, \tag{8}$$

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 [29]. It should be noted that the manipulator reaches a kinematic singularity when μ reaches zero. The derivative of the manipulability as a function of the joint variables is explained in [30] and used in [31]:

$$\frac{\partial \mu}{\partial q_i} = \mu \, \text{trace} \left\{ \frac{\partial \mathbf{J}_w}{\partial q_i} \mathbf{J}_w^+ \right\}. \tag{9}$$

The manipulability Jacobian matrix $\mathbf{J}_\mu$ as a function of the joint variables is

$$\mathbf{J}_\mu = \left[\frac{\partial \mu}{\partial q_1} \ \cdots \ \frac{\partial \mu}{\partial q_n} \right]. \tag{10}$$

3 Proposed workspace Determination Method

This section describes the workspace determination procedure proposed in this paper. 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

Table 1: Details about task name, category, type, and hierarchy level in end-effector velocity action $\mathscr{A}$, symbol (E) for equality control objective and (I) for inequality

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

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 $\varepsilon_{\text{kinematic}}$, 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 $\varepsilon_{\text{kinematic}}$ and the 2-norm of the end-effector velocity $||\mathbf{t}||_2$ goes under a selected threshold $\varepsilon_{\text{velocity}}$, it means that the end-effector has reached the workspace boundary and its position is saved. Afterwards, the robot is again set to the starting configuration and the end-effector is moved along another vector. After the end-effector has been moved along all the displacement 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 1 collects the tasks employed by the TPIK algorithm for the workspace generation and their hierarchical priority levels. The algorithm is developed in C++. The method presented in this section was developed to determine the robot workspace reachable by the end-effector for any orientation.

4 Family of Robots Under Study

This section describes the three Nimbl'Bot robots employed for testing the proposed workspace determination algorithm. First, the mechanism that composes and actuates the robots is described. Then, some initialization steps necessary to run the process are presented.

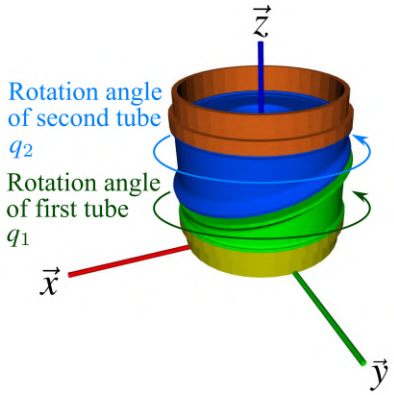
4.1 Mechanism description

The mechanism, called NB-module, developed and patented by the company Nimbl'Bot [32] is used as a case study in this paper. Here, the NB-module is presented. It comprises two closed kinematic chains, one internal and one external, actuated by two motors. So, it is a two degrees of freedom mechanism. The internal kinematic chain

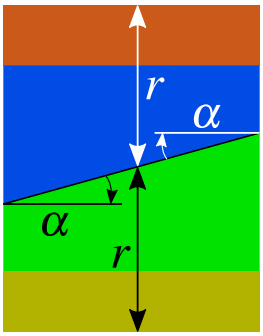
Algorithm 1 Workspace determination algorithm

Require: Several starting configurations and a set of displacement vectors along different directions. The thresholds $\varepsilon_{\text{kinematic}}$ and $\varepsilon_{\text{velocity}}$ 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 > ε_kinematic do
            while End-effector twist ||t||_2 > ε_velocity do
                Move end-effector along k^th vector with t_t.
            end while
        end while
        Save end-effector position.
    end for
end for
```



(a) 3D view and rotation angles



(b) Front view and design

Fig. 1: NB-module representation in the home pose with rotation angles and design parameters [23]

is passive and ensures the strength of the entire design. The external kinematic chain is the active one, actuated by the two motors and shown Fig. 1a. It comprises the yellow and orange platforms and the blue and green hollow cylinders, both cut by an oblique plane. The green tube rotates around the vertical axis of the yellow platform when the first motor actuates it. This corresponds to the first joint and its angle is called q_1. The blue tube rotates around the vertical axis of the orange platform when the second motor actuates it. This corresponds to the second joint and its angle is called q_2. The rotation of these two tubes is entirely independent and continuous. Figure 1b shows the design parameters of the NB-module. The variable r indicates half the height of the NB-module, set to 0.07 m for the rest of this paper. The variable α represents the slope of the oblique plane that cuts the two tubes, set to 15° for the rest of this paper. For more details, the NB-modules are described in [23, 33].

4.2 Description of the robots under analysis

Several NB-modules can be attached serially to generate different kinematic redundant manipulators. In this paper, the workspace of three different robots is analyzed. The first robot, called NB-R1, is shown in Fig. 2. It comprises six NB-modules serially attached and has twelve degrees of freedom. The second robot, shown in Fig. 3, is called NB-R2. It has ten NB-modules and can be divided into three main regions: the shoulder (three NB-modules), the elbow (four NB-modules) and the wrist (three NB-modules). In total, it has twenty degrees

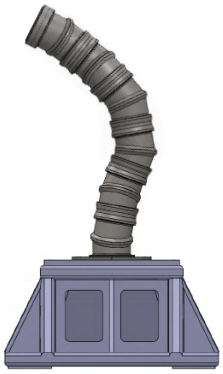


Fig. 2: Robot NB-R1 formed of six NB-modules

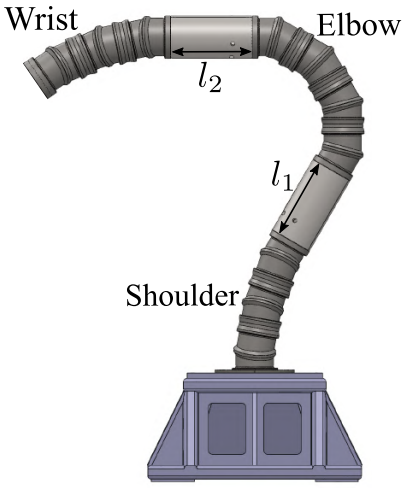


Fig. 3: Robot NB-R2 formed of ten NB-modules plus two links l_1 and l_2

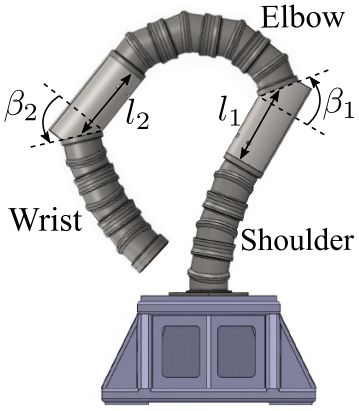


Fig. 4: Robot NB-R3 formed of ten NB-modules plus two links l_1 and l_2 and two offsets β_1 and β_2

Table 2: Robot design details

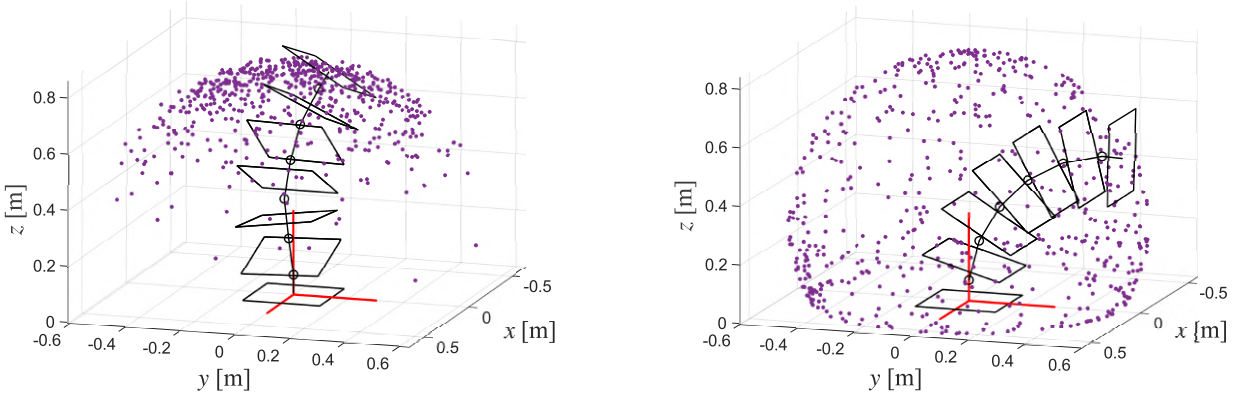
NB-module half height r	0.07 m
NB-module slope α	15°
Link lengths l_1 and l_2	0.2 m
Offsets β_1 and β_2	-45°

of freedom. These three regions are connected by two links l_1 and l_2 of length equal to 0.2 m. The third robot, represented in Fig. 4, is organized as the second one, i.e., with ten NB-modules and two links. It is called NB-R3. So, it has twenty 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 NB-module. The length of both the links l_1 and l_2 is equal to 0.2 m and the offsets β_1 and β_2 equal to -45°. Table 2 summarizes the robot dimensions.

5 Test description

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 shows the end-effector position distribution for several starting configurations using the random or proposed initialization. The randomly generated points are located in the upper area of the workspace, while the proposed initialization is distributed uniformly at the starting point.

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



(a) Random initialization

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

Fig. 5: Comparison between random and proposed initialization for robot starting configurations. The purple points are the starting end-effector position generated through the two methods.

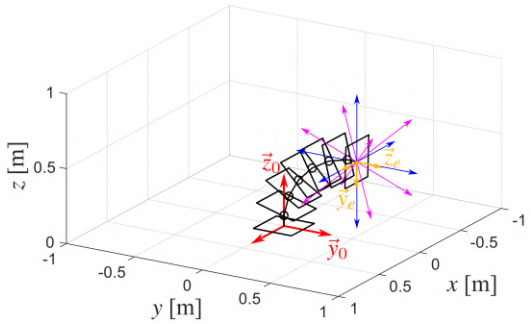


Fig. 6: Representation of the NB-R2 and displacement vectors applied to the robot end-effector position. $\mathcal{F}_0$ is the base frame and $\mathcal{F}_e$ is the end-effector frame.

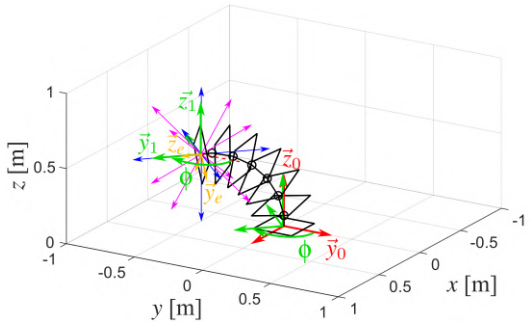


Fig. 7: Representation of the NB-R2 with $\phi = 135°$ and displacement vectors applied to the end-effector position rotated of the same ϕ. $\mathcal{F}_0$ is the base frame, $\mathcal{F}_1$ is the base frame rotated of ϕ around $z_0 \equiv z_1$ and $\mathcal{F}_e$ is the end-effector frame.

respectively,

$$\vec{v}_{1,2} = \begin{bmatrix} \pm 1 \\ 0 \\ 0 \end{bmatrix}, \ \vec{v}_{3,4} = \begin{bmatrix} 0 \\ \pm 1 \\ 0 \end{bmatrix}, \ \vec{v}_{5,6} = \begin{bmatrix} 0 \\ 0 \\ \pm 1 \end{bmatrix}. \tag{11}$$

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

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

where c is equal to $1/\sqrt{3}$. The displacement vectors applied to the end-effector are shown in Fig. 6. These 14 vectors are chosen for the simulation to move the robot uniformly in all directions. In the future, more analysis could be done to identify if more or fewer displacement vectors give 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. 7. The rotation angle equals the azimuth angle ϕ of the transformation matrix between the base and end-effector frames. The azimuth angle ϕ can be obtained using the notation presented in [33]. Table 3 provides the test implementation details and values.

Table 3: Implementation details

$\varepsilon_{\text{kinematic}}$	10^{-6}
$\varepsilon_{\text{velocity}}$	10^{-3} m/s
$\lvert \mathbf{t}_t \rvert$	0.14 m/s

Table 4: Machine details

CPUs number	4
CPU model	Intel Core i7 10th Gen, 1.30GHz
Language	C++

Table 4 gives the machine details used for the simulations.

In this case, 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. With fewer repetitions, there were not enough points to correctly identify the boundaries for the analyzed robots. In the 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 the smaller NB-R2 and the bigger NB-R1/3 workspaces. There is no rule to identify the best repetition number, 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.

6 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.

6.1 Proposed optimized ray-based method results

Figure 8 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 graph on the left shows the vertical section, defined by the yz-plane, of the NB-R1 workspace. Then, on the right, the upper graph shows a 3D view of the complete NB-R1 workspace and the lower one presents a 3D view of the inner boundaries. The NB-R1 workspace boundaries are uniformly and continuously identified and can be easily visualized. This workspace is symmetric around the z axis. Figure 9 shows three views of the NB-R2 workspace. The graph on the left shows the vertical section, defined by the yz-plane, of the NB-R2 workspace. Then, on the right, the upper graph shows

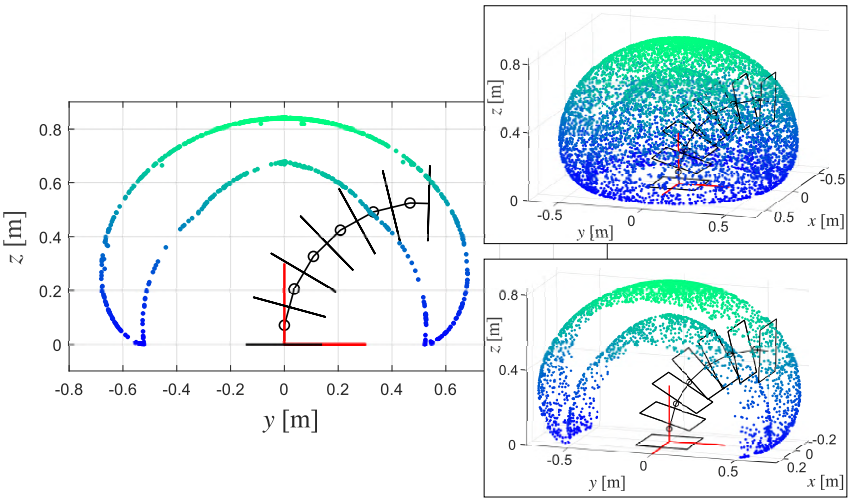


Fig. 8: Generated points describing robot NB-R1 workspace using the proposed optimized ray-based method. The graph on the left shows a vertical section of the workspace in yz-plane. The graph on the upper right shows the complete workspace. The graph on the lower right shows the internal boundary view.

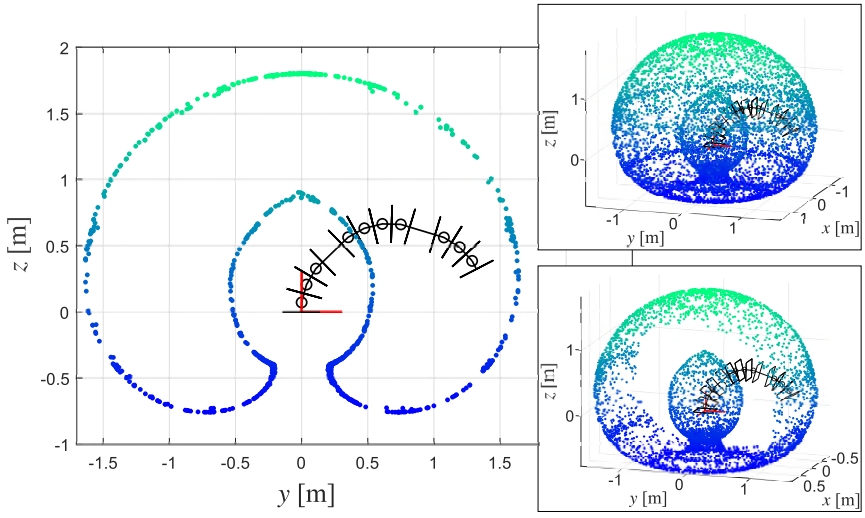


Fig. 9: Generated points describing robot NB-R2 workspace using the proposed optimized ray-based method. The graph on the left shows a vertical section of the workspace in *yz*-plane. The graph on the upper right shows the complete workspace. The graph on the lower right shows the internal boundary view.

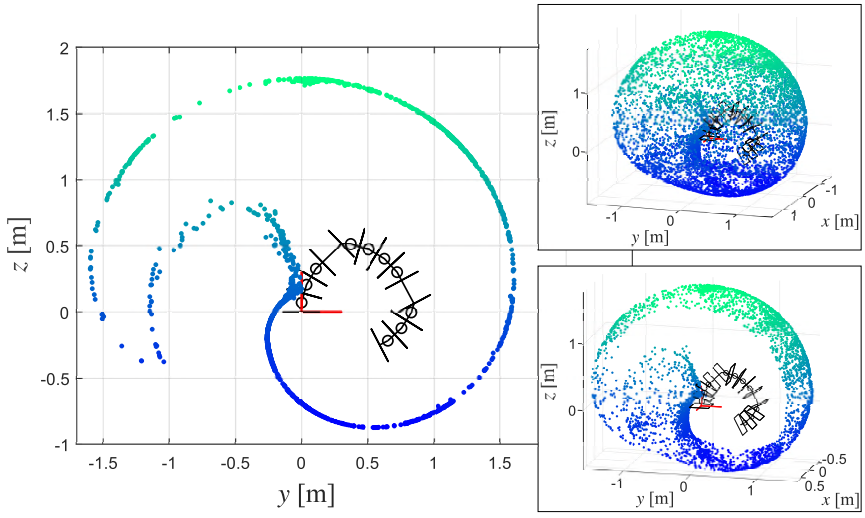


Fig. 10: Generated points describing robot NB-R3 workspace using the proposed optimized ray-based method. The graph on the left shows a vertical section of the workspace in *yz*-plane. The graph on the upper right shows the complete workspace. The graph on the lower right shows the internal boundary view.

a 3D view of the complete NB-R2 workspace and the lower one presents a 3D view of the inner boundaries. Again, the NB-R2 workspace is uniformly, continuously detected, and symmetric around the z axis. In this case, the total workspace is bigger than for the NB-R1, thanks to the higher number of NB-modules and the presence of two links l_1 and l_2. Finally, Fig. 10 shows three views of the NB-R3 workspace. The graph on the left shows the vertical section, defined by the *yz*-plane, of the NB-R3 workspace. Then, on the right, the upper graph shows a 3D view of the complete NB-R3 workspace, and the lower one presents a 3D view of the inner boundaries of the workspace. The NB-R3 workspace has a similar volume to the NB-R2 one since both robots have the same number of modules 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

Table 5: Comparison of optimized ray-based, pseudo-inverse Jacobian 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
	Pseudo-inverse Jacobian ray-based	8120	4.7 min	Internal partially wrong
	Monte Carlo	500000	26 min	Lower and internal missing
NB-R2	Optimized ray-based	8120	0.8 min	Uniform and continuous
	Pseudo-inverse Jacobian ray-based	8120	0.7 min	Internal partially wrong
	Monte Carlo	500000	26 min	Lower and internal missing
NB-R3	Optimized ray-based	8120	4.5 min	Uniform and continuous
	Pseudo-inverse Jacobian ray-based	8120	3.7 min	Internal partially wrong
	Monte Carlo	500000	26 min	Lower and internal missing

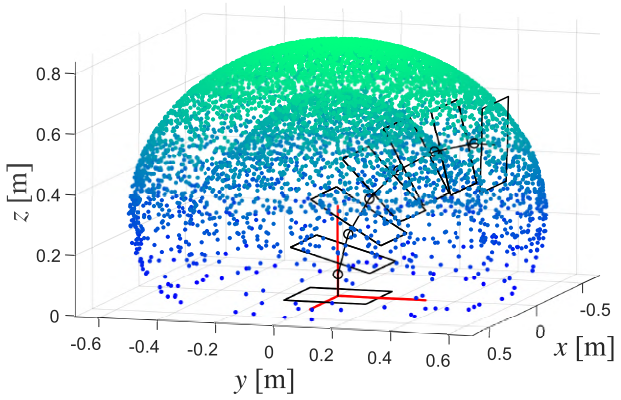


Fig. 11: Points describing NB-R1 workspace obtained starting robot joints with all different random values

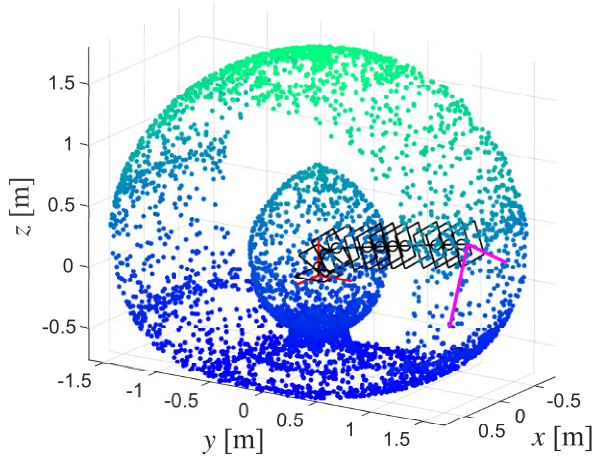


Fig. 12: Robot configuration, black lines, and linear Jacobian matrix singular vectors, magenta lines, for the NB-R2 in one point on its workspace boundaries

inner boundaries are rougher and not perfectly described. This happens because, despite the presence of the kinematic optimization tasks, the robot reaches singular configurations inside the workspace, not reaching the real boundary. This behavior does not appear with the other robots and 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 provides the total time to determine the three robot workspaces.

As previously described, the robots are initialized in a specific way to obtain a uniform distribution of points around the entire workspace. Figure 11 shows the workspace of the NB-R1 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 12 shows the NB-R2 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 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.

6.2 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 13 shows the generated points for the NB-R1 workspace without the optimization tasks. Some of the generated points did not reach the workspace boundary, resulting inside the workspace. Most 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. 14. Figure 15 shows the workspace boundaries 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. 10. However, in the case of no optimization, more final positions are returned inside the workspace and further from the boundaries.

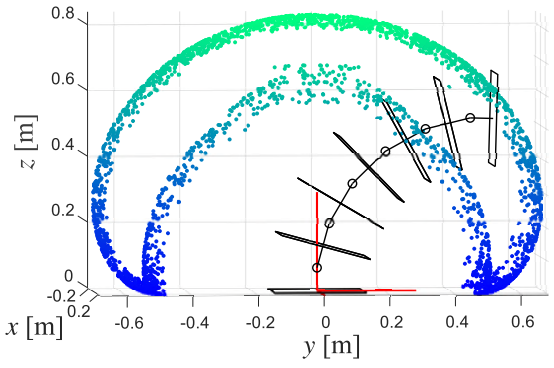


Fig. 13: Internal boundary view of points describing NB-R1 workspace obtained with pseudo-inverse Jacobian ray-based

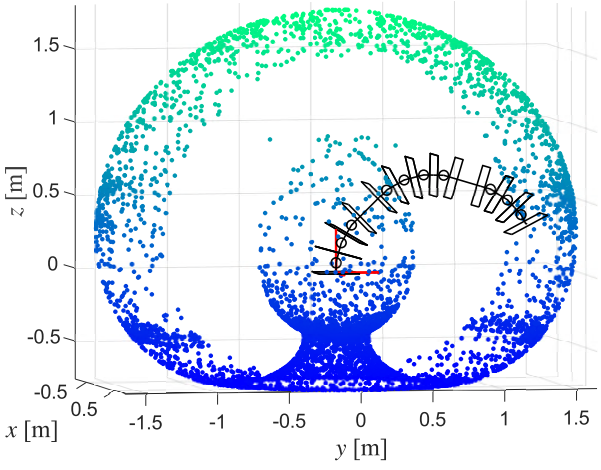


Fig. 14: Internal boundary view of points describing NB-R2 workspace obtained with pseudo-inverse Jacobian ray-based

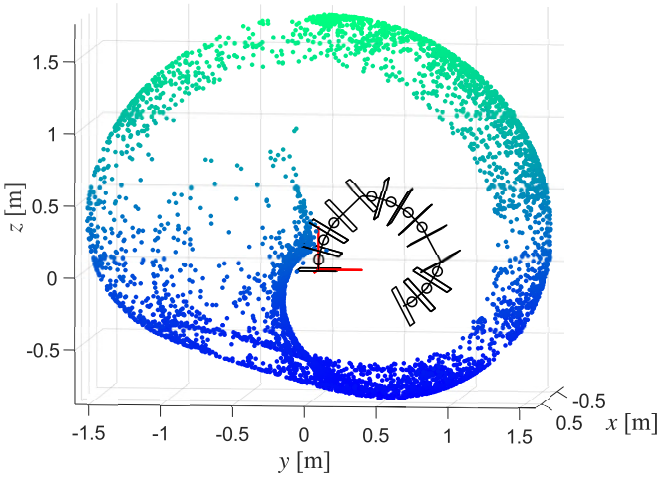


Fig. 15: Internal boundary view of points describing NB-R3 workspace obtained with pseudo-inverse Jacobian ray-based

The only advantage of the pseudo-inverse Jacobian ray-based method is that it takes less time to run. Table 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 inside the workspace.

6.3 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 compares the results of optimized ray-based and Monte Carlo methods for each robot. In all cases, the optimized ray-based performance is better. The computational time of the Monte Carlo method for the three Nimbl'Bot robots is the same since the process only requires generating random robot configurations and saving the end-effector po-

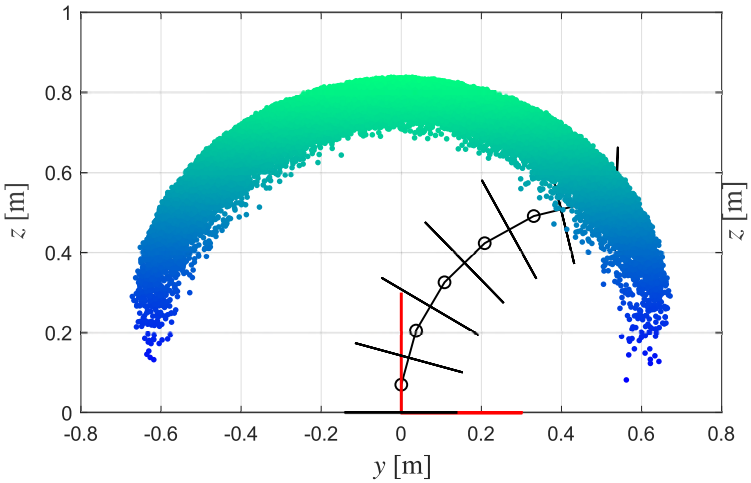


Fig. 16: Comparison between points describing NB-R1 workspace generated with optimized ray-based and Monte Carlo methods. Blue-green points are generated with the optimized ray-based method. Red-yellow points are generated with the Monte Carlo method.

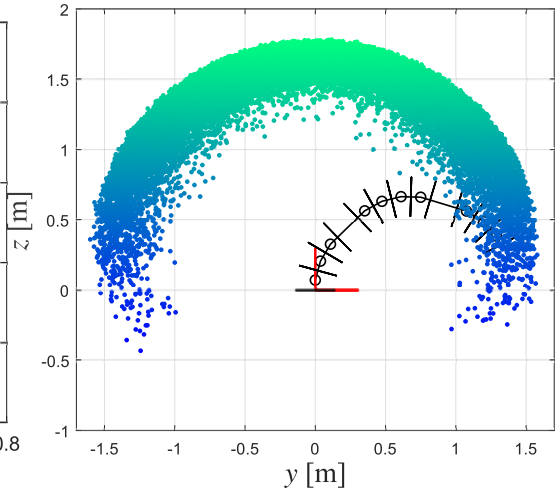


Fig. 17: Comparison between points describing NB-R1 workspace generated with optimized ray-based and Monte Carlo methods. Blue-green points are generated with the optimized ray-based method. Red-yellow points are generated with the Monte Carlo method.

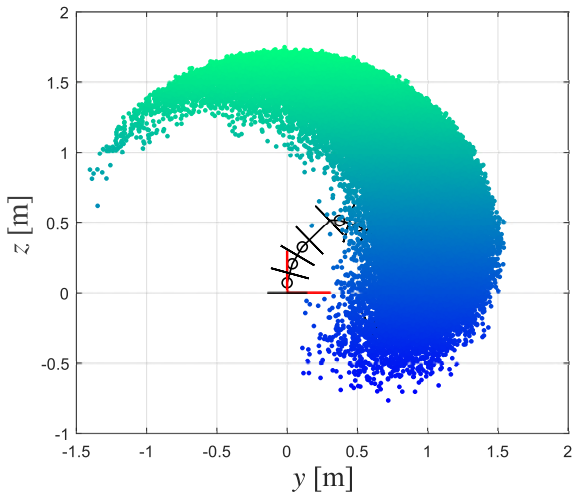


Fig. 18: Comparison between points describing NB-R1 workspace generated with optimized ray-based and Monte Carlo methods. Blue-green points are generated with the optimized ray-based method. Red-yellow points are generated with the Monte Carlo method.

sitions. Figure 16 shows a section of the NB-R1 workspace generated using the Monte Carlo method, red-yellow points, together with the workspace generated using the optimized ray-based method, green-blue points. The point colors are related to the height along axis z. 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 less than ten minutes using only 8120 points. Figure 17 shows the NB-R2 workspace generated using the Monte Carlo method. Figure 18 shows the NB-R3 workspace generated using the Monte Carlo method. The same considerations made for the NB-R1 can be done in these other two cases. The Monte Carlo method can not generally be used for workspace determination. In fact, some groups of points are more likely to be selected than others. This behavior results in a wrong identification of some boundaries.

The comparison presented here clearly shows how the proposed workspace determination algorithm has better performance than a simple Monte Carlo method. However, it is known that random sampling in joint space, translated to task coordinates, can yield a subpar representation. So, the Monte Carlo method could be improve through the use of adaptive sampling techniques. One example of these techniques is the one presented in [34] that takes advantage of the algebraic nature of the point cloud, defining the workspace area. Future works will address this point to improve the methods comparison.

7 Discussion about the proposed workspace determination algorithm

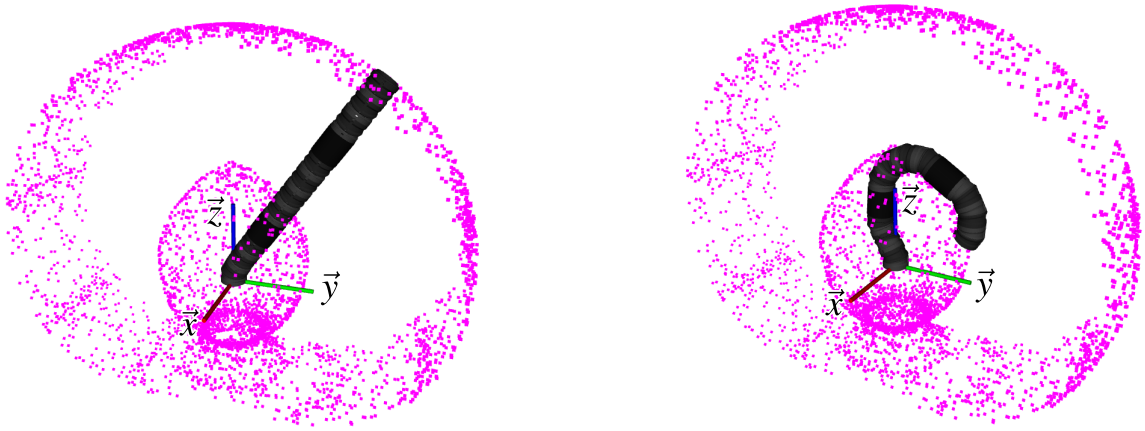
This section highlights some limitations of the proposed workspace determination algorithm. The first major limitation is the need for prior knowledge about the analyzed manipulator workspaces. So, there is no way to ensure the correctness of the result obtained by this algorithm. However, one way to check the result correctness is by plotting some robot configuration for each workspace. Figure 19 shows the NB-R1 robot CAD model plotted using the 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 20 and 21 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 robot workspaces. However, the inner boundary detection can lead to errors when the number of NB-modules increases. Figure 22 shows a Nimbl'Bot robot composed of 12 NB-modules. No link or offset is inserted between the NB-modules. This robot

(a) Robot on external boundaries

(b) Robot on internal boundaries

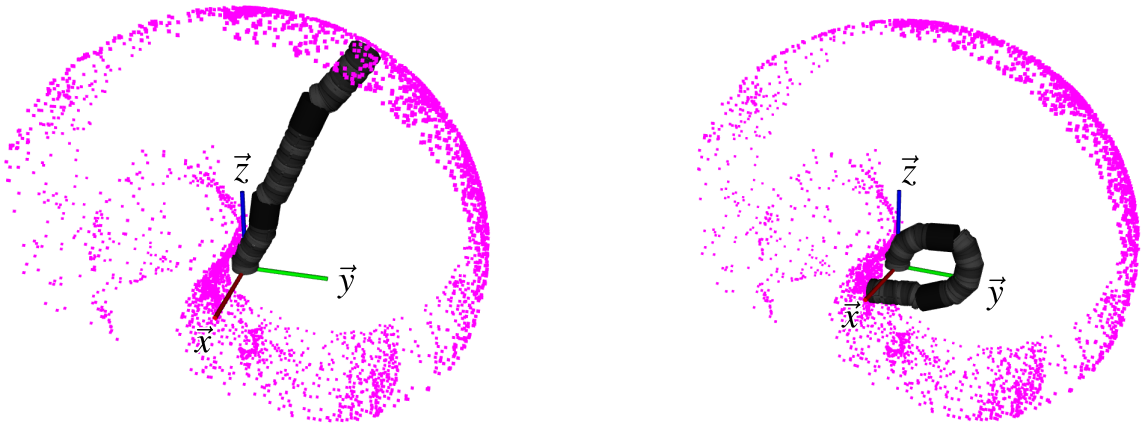
Fig. 19: NB-R1 robot configuration on two collected points by the workspace determination algorithm. Obtained workspace points highlighted in magenta.



(a) Robot on external boundaries

(b) Robot on internal boundaries

Fig. 20: 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

Fig. 21: NB-R3 robot configuration on two collected points by the workspace determination algorithm. Obtained workspace points highlighted in magenta.

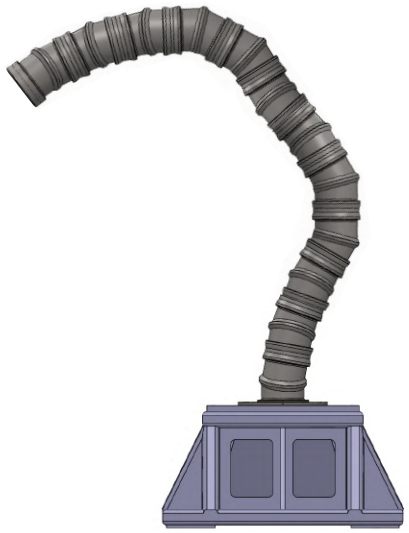


Fig. 22: Nimbl'Bot robot composed of 12 NB-modules

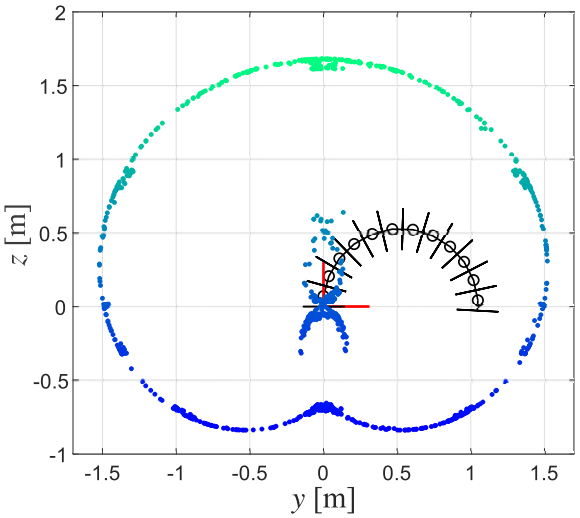


Fig. 23: Vertical section in *yz*-plane of points describing workspace of 12 NB-module robot

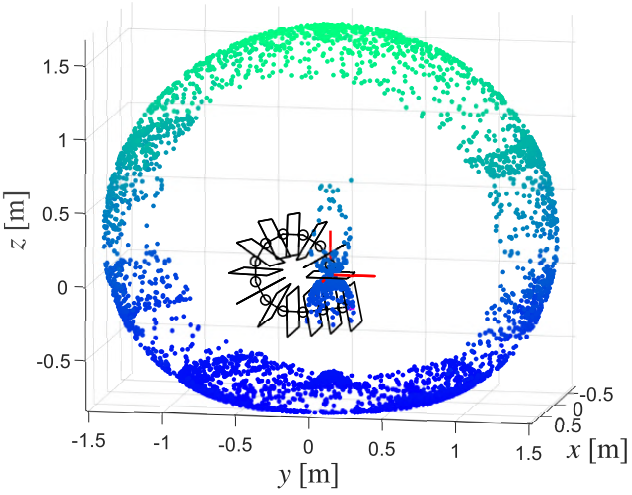


Fig. 24: Configuration of 12 NB-module robot on point inside the workspace which does not belong to the boundaries

can touch its base. Figure 23 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. 23. 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 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. Figure 24 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. This algorithm limitation can generate problems in correctly identifying the manipulator workspaces. Further studies on this problem will be done to improve the workspace determination process.

8 Conclusions

This paper 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 manipulators. 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 smallest 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 and Monte Carlo methods. The proposed optimized ray-based workspace determination algorithm is faster, more accurate and requires fewer points than the Monte Carlo one. Finally, two main limitations 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 present some errors because the robot remained stuck in some internal configurations.

Future work will address the workspace determination for a specific end-effector orientation. This is important when planning the workpiece placement inside the robot reachable area. Then, additional steps should be added to the workspace determination process to escape the inner boundary identification limitation. The point distribution obtained from the Monte Carlo method will be improved developing some adaptive sampling techniques to compare the improved results with the method proposed in this paper. Finally, a point interpolation and surface generation method will be developed to plot the identified workspace surfaces.

Acknowledgements

This work was supported by the ANRT (Association Nationale de la Recherche et de la Technologie) grant CIFRE n° 2020/1051 and Nimbl'bot (`https://nimbl-bot.com/`).

References

[1] Dombre, E., and Khalil, W., (2004). *Modeling, identification and control of robots*. Butterworth-Heinemann.

[2] Du, Z.-c., Ouyang, G.-Y., Xue, J., and Yao, Y.-b., 2020. "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, IEEE, pp. 444–449.

[3] Yahya, S., Moghavvemi, M., and Mohamed, H. A., 2011. "Geometrical approach of planar hyper-redundant manipulators: Inverse kinematics, path planning and workspace". *Simulation Modelling Practice and Theory,* **19**(1), pp. 406–422.

[4] Bohigas, O., Manubens, M., and Ros, L., 2012. "A complete method for workspace boundary determination on general structure manipulators". *IEEE Transactions on Robotics,* **28**(5), pp. 993–1006.

[5] Peidró, A., Reinoso, Ó., Gil, A., Marín, J. M., and Payá, L., 2017. "An improved monte carlo method based on gaussian growth to calculate the workspace of robots". *Engineering Applications of Artificial Intelligence,* **64**, pp. 197–207.

[6] Guan, Y., and Yokoi, K., 2006. "Reachable space generation of a humanoid robot using the monte carlo method". In Proceedings of International Conference on Intelligent Robots and Systems, IEEE, pp. 1984–1989.

[7] Li, L., Shang, J., Feng, Y., and Yawen, H., 2018. "Research of trajectory planning for articulated industrial robot: a review". *Computer engineering and applications,* **54**(5), pp. 36–50.

[8] Zhao, Z., He, S., Zhao, Y., Xu, C., Wu, Q., and Xu, Z., 2018. "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, IEEE, pp. 637–642.

[9] Fernández-Sarría, A., Martínez, L., Velázquez-Martí, B., Sajdak, M., Estornell, J., and Recio, J., 2013. "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**, pp. 176–185.

[10] Wang, Y., and Chirikjian, G. S., 2004. "Workspace generation of hyper-redundant manipulators as a diffusion process on se (n)". *IEEE Transactions on Robotics and Automation,* **20**(3), pp. 399–408.

[11] Dong, H., Du, Z., and Chirikjian, G. S., 2013. "Workspace density and inverse kinematics for planar serial revolute manipulators". *Mechanism and Machine Theory,* **70**, pp. 508–522.

[12] Dong, H., Fan, T., Du, Z., and Chirikjian, G. S., 2015. "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, Vol. 57144, American Society of Mechanical Engineers, p. V05CT08A039.

[13] Han, Y., Pan, J., Xia, M., Zeng, L., and Liu, Y.-J., 2021. "Efficient se (3) reachability map generation via interplanar integration of intra-planar convolutions". In Proceedings of International Conference on Robotics and Automation, IEEE, pp. 1854–1860.

[14] Kurniawati, H., and Hsu, D., 2004. "Workspace importance sampling for probabilistic roadmap planning". In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), Vol. 2, IEEE, pp. 1618–1623.

[15] Edwards, P. B., Baskar, A., Hills, C., Plecnik, M., and Hauenstein, J. D., 2023. "Output mode switching for parallel five-bar manipulators using a graph-based path planner". In 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 9735–9741.

[16] Abbasnejad, G., Eden, J., and Lau, D., 2019. "Generalized ray-based lattice generation and graph representation of wrench-closure workspace for arbitrary cable-driven robots". *IEEE Transactions on Robotics,* **35**(1), pp. 147–161.

[17] Zhang, Z., Cheng, H. H., and Lau, D., 2020. "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), pp. 8–15.

[18] Cheng, H. H., and Lau, D., 2022. "Ray-based cable and obstacle interference-free workspace for cable-driven parallel robots". *Mechanism and Machine Theory,* **172**, p. 104782.

[19] Chirikjian, G. S., and Burdick, J. W., 1994. "A hyper-redundant manipulator". *IEEE Robotics & Automation Magazine,* **1**(4), pp. 22–29.

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

[21] Simetti, E., Casalino, G., Aicardi, M., and Wanderlingh, F., 2018. "Task priority control of underwater intervention systems: Theory and applications". *Ocean Engineering,* **164**, pp. 40–54.

[22] Simetti, E., Casalino, G., Aicardi, M., and Wanderlingh, F., 2019. "A task priority approach to cooperative mobile manipulation: Theory and experiments". *Robotics and Autonomous Systems,* **122**, p. 103287.

[23] Ginnante, A., Caro, S., Simetti, E., and Leborne, F., 2023. "Kinetostatic optimization for kinematic redundancy planning of nimbl'bot robot". *Journal of Mechanisms and Robotics*, pp. 1–20.

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

[25] Khan, W. A., and Angeles, J., 2005. "The kinetostatic optimization of robotic manipulators: The inverse and the direct problems". *Journal of Mechanical Design,* **128**(1), pp. 168–178.

[26] Angeles, J., and López-Cajún, C. S., 1992. "Kinematic isotropy and the conditioning index of serial robotic manipulators". *The International Journal of Robotics Research,* **11**(6), pp. 560–571.

[27] Pond, G., and Carretero, J. A., 2006. "Formulating jacobian matrices for the dexterity analysis of parallel manipulators". *Mechanism and Machine Theory,* **41**(12), pp. 1505–1519.

[28] Yoshikawa, T., 1985. "Manipulability of robotic mechanisms". *The international journal of Robotics Re-*

search, **4**(2), pp. 3–9.

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

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

[31] Marani, G., Kim, J., Yuh, J., and Chung, W. K., 2002. "A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators". In Proceedings of International Conference on Robotics and Automation, Vol. 2, IEEE, pp. 1973–1978.

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

[33] Ginnante, A., Leborne, F., Caro, S., Simetti, E., and Casalino, G., 2021. "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, Vol. 85444, American Society of Mechanical Engineers.

[34] Di Rocco, S., Eklund, D., and Gäfvert, O., 2022. "Sampling and homology via bottlenecks". *Mathematics of Computation,* **91**(338), pp. 2969–2995.