Modeling the Reachability Space of Robotic Manipulators through Ellipsoid Equations

Journal of Intelligent & Robotic Systems (2025) 111:90
https://doi.org/10.1007/s10846-025-02294-5

REGULAR PAPER

Modeling the Reachability Space of Robotic Manipulators through Ellipsoid Equations

Rosario Francesco Cavelli[1] · Pangcheng David Cen Cheng[1] · Marina Indri[1]

Received: 19 December 2024 / Accepted: 1 July 2025

Abstract
Accurate modeling of the reachability space of robotic manipulators is crucial for tasks such as robot positioning, trajectory planning, and human-robot collaboration. Traditional methods based on reachability and capability maps often rely on the workspace discretization, which can be computationally expensive and less adaptable to real-time applications. To address these limitations, this paper introduces a new approach to estimate and model the reachability space of manipulators using a single ellipsoid equation. By generating a point cloud from the robot kinematic model, the proposed method avoids the complexity of forward and inverse kinematics calculations to generate the set of reachable points. The ellipsoid parameters are computed by exploiting two techniques: an optimization-based process and a machine learning approach that leverages the PointNet model. Different optimization algorithms and variants of the PointNet model are tested and compared in terms of computational efficiency and accuracy. Experimental results demonstrate the effectiveness of the proposed method in capturing and modeling an accurate representation of the reaching capabilities of a robotic manipulator.

Keywords Reachability space · Robotic manipulators · Ellipsoid model · Kinematic optimization · Point cloud

1 Introduction

The study of the manipulator's capability to reach a point in the 3D space is a fundamental task in robotics. An accurate representation and description of the Reachability Space (RS), i.e., the set of points that are reachable with at least one end-effector (EE) pose [1], are critical for various applications, including robot positioning for workspace optimization [2, 3], path and trajectory planning [4, 5], collision avoidance [6], and to ensure human safety during human-robot collaboration (HRC) [7, 8]. In addition, the increasing application of mobile manipulators in warehouses and facilities has led to reconsider the workspace of a manipulator. The additional degrees of freedom (DOF) introduced by the mobile base significantly increase the workspace of the robot. Indeed, unlike fixed manipulators, whose reachable points are uniquely determined by their positioning, for mobile manipulators currently unreachable poses can become reachable as a result of the robot repositioning. In such a case, modeling the manipulator's reachable space helps to find a suitable robot's positioning for the execution of the desired task.

One of the main methods used to represent the RS of a robotic arm is through a Reachability Map (RM) [9]. An RM for a robotic manipulator describes the region in the workspace that the EE of the robot can reach. Usually, it is constructed as a data structure made of 3D points, representing directly those belonging to the RS; otherwise, voxels or spheres are used to reduce the complexity of the RM by clustering several points in a single element [9, 10]. They are constructed by discretizing the robot's workspace, sampling reachable poses, and storing information about these poses in a structured manner. The use of an RM can be further expanded by associating to each element a measure representing how many poses a point can be reached with, and this parameter is often called *reachability measure*. From this combination, the Capability Map (CM) for a robotic manipulator can be determined [11]. It is possible to use RMs to optimize the positioning of manipulators to ensure that the points or the poses needed to complete a given task (or a

✉ Pangcheng David Cen Cheng
pangcheng.cencheng@polito.it

Rosario Francesco Cavelli
rosario.cavelli@polito.it

Marina Indri
marina.indri@polito.it

[1] Dip. di Elettronica e Telecomunicazioni, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy

Published online: 05 August 2025

set of them) not only belong to the RS, but are also part of the regions with the highest value of the chosen reachability measure.

As an alternative to the RMs, it is possible to use an analytical representation of the RS for a robotic manipulator. In this case, the manipulator's reachable workspace is mathematically modelled, achieving a detailed and continuous description of the space the manipulator can reach [12]. Unlike RMs, which are often discrete, the analytical method provides a comprehensive understanding of the spatial limits. Results in this domain frequently employ geometric constructs, such as ellipsoids or convex polytopes, to approximate the RS. These shapes are balanced between simplicity and accuracy, making them suitable for various applications in robotics, where quick and efficient calculations are necessary. Moreover, it is possible to construct a geometric structure that describes the space regions where the manipulator reaches the maximum values for a given physical quantity, e.g., the EE speed [13].

In this paper, we propose a novel method to compute and represent an estimation of the RS of a robotic arm using the equation of an ellipsoid. Our approach, which is suitable for a wide range of robotic manipulators (with fixed or mobile bases), combines the analytical representation of the RS with the capability to include different manipulability metrics. In this way, we can obtain a workspace description similar to reachability and capability maps, but enclosed in a single ellipsoid equation that results more compact and faster to use. The ellipsoid parameters are obtained through an optimization process, based on a point cloud that reliably estimates the RS. Such a set of 3D points is generated using only the robot kinematic model, thereby avoiding the need to solve the Forward Kinematic (FK) and the Inverse Kinematic (IK) problems. The optimization is tackled using two different approaches: one formulated as a proper optimization problem, and the other leveraging the PointNet Machine Learning (ML) model.

The remainder of the paper is organized as follows: Section 2 reviews the state of the art and highlights the shortcomings of existing methods, while Section 3 details the theoretical foundation on which the proposed method is based. Section 4 outlines the method proposed to generate the point cloud representing the RS of a manipulator, and the strategies used to compute the corresponding ellipsoid equation. The experimental validation of the proposed methodologies is presented and discussed in Section 5. Finally, Section 6 investigates potential applications and future research directions.

2 Related Works

This section briefly reviews the most relevant works related to the representation of RS. The first part focuses on various approaches where RSs have been utilized for different applications, while in the second part, the main methodologies to obtain a representation of such spaces are outlined along with their limitations.

2.1 Reachability Space Representation and its Applications

Reachability and capability maps are commonly used to represent robotic manipulation abilities in various applications. Zacharias et al. in [9] presented an early representation of a manipulator's RS, by mapping the reachable workspace through joint-space sampling, and calculating positions of the Tool Center Point (TCP) using FK equations. Clustering TCP coordinates into spheres and uniformly sampling them to consider different orientations, they introduced a reachability index based on the number of solutions obtained from the IK problem, creating a 3D map of the robot's manipulation capabilities. A similar approach in [14] uses Monte Carlo sampling to map the RS boundaries of a humanoid robot.

Alternative RS models include the voxel-based representation developed by Anderson-Sprecher et al. in [15]. The construction of the CM is achieved by means of an enlargement of the swept volume obtained from the movement of a manipulator link, which is then combined with the motion of the preceding ones in the kinematic chain. Rather than dexterity, the authors utilized a metric based on the time required to reach voxel centers from a predefined pose, prioritizing regions with faster reachability. The Orientation-Based Reachability (OBR) map introduced in [16] constructs RS layers for different end-effector orientations, adaptable across various end-effectors without recalculating each orientation.

The RSs demonstrated the possibility of enhancing the efficiency of motion planning and manipulation tasks [17]. Vahrenkamp et al. introduced the IK-Map in [18], a voxel-based map that stores approximated IK solutions for each EE pose. This map speeds up the online computation of the IK by providing an initial estimate of the solution. In [19], the impact of obstacles on RS is analyzed, resulting in the proposal of two approaches to model the partially reduced RM: a collision-sensitive RM and an ML-based RM that adapts to obstacles. The latter one was the fastest, but it required to retrain the model each time a new kinematic structure is used.

RS representations have also enhanced mobile manipulators and humanoids repositioning to optimize task execution. In [20, 21], RMs are used for mobile manipulator placement for task execution based on reachability and pattern recognition, while the work in [22] proposed optimization-based repositioning to minimize base placements. Dual-arm manipulation tasks also leverage RS. Authors in [23] used scores coming from the processing of CMs to decide on dual-arm placement in humanoid robots. Burget et al. in [24] inverted

the voxel-based representation of an RM and obtained an Inverse Reachability Map (IRM) to optimize the placement of humanoid robots based on the EE target. A similar IRM-based approach is used in [25], which is presented as an open-source library for reachability analysis and optimized robot base placement.

The Oriented Surface Reachability Map (OSRM) developed in [26] enabled humanoid robot positioning in unstructured environments by combining planar segmentation with an IRM for inclined surfaces. Integration of ML with a representation of the manipulator's RS was explored by Jauhri et al. [27], who used RS knowledge to guide reinforcement learning methods, improving efficiency in mobile manipulator tasks and shortening training time. The use of RS representations has also enhanced the collaboration between humans and robots. The authors in [28] demonstrated how the optimal placement of mobile manipulators based on CM eased human interaction and object manipulation, thereby accelerating robots' response to human actions.

Yoshikawa et al. in [29] introduced a mathematical approach to RS, which illustrated manipulator dexterity via Jacobian-based ellipsoids. These ellipsoids are an effective model for characterizing the ease of motion and force application along different directions. In [30], convex optimization was used to approximate the RS of fixed-base manipulators, with the workspace represented as convex hulls. Haviland et al. in [31] designed a reactive controller using manipulability ellipsoids to maximize manipulability. Chiacchio et al. in [32] further expanded RS modeling for redundant manipulators by defining polytopes to reflect force capabilities. Their work was later extended by [33] to over-approximate RS using zonotope techniques. A polytope-based RS for humanoid arm manipulability presented in [13] provides an algorithm to adapt velocity-polytopes for obstacle avoidance in complex environments. Skuric et al. in [34] used convex polytopes to approximate RS under actuation and kinematic constraints. Mathematical RS models have also applications in human-robot interaction. In [35], capsules were used to approximate the workspace of a human operator, which is essential for safe HRC. Similarly, in [36], a RS of human limbs was modelled via a convex hull, which facilitates collaboration and ensures a safe environment.

Overall, RS representations support task planning, obstacle avoidance, and workspace optimization, particularly through RM-based spatial distributions. However, they are computationally demanding and also require significant memory storage.

2.2 Algorithms for Reachability Space Computation

RS computation methods primarily involve either 3D space or joint-space sampling. Space sampling techniques [25, 37, 38] discretize a robot's 3D workspace, assuming that it is shaped like a cube or a sphere based on extended manipulator configurations [9, 39]. The sampling size impacts the accuracy and memory use of the RM. Smaller samples increase voxel generation, but require additional time to solve the IK, due to the workspace overestimation. Moreover, not all the generated voxels provide valid IK solutions, leading to redundant and useless calculations.

On the other hand, joint-space sampling techniques [9, 24] reduces computational burden by using FK to build RS models. Nevertheless, joint redundancy introduces inefficiency, since similar poses may arise from various joint values, increasing the computational load to resolve joint combinations.

In this paper, we present an easy-to-use method to generate a mathematical model of the RS. Our approach not only simplifies the modeling process, but also allows to obtain a practical ellipsoid equation that encloses the maximum values of the desired manipulability measure. The main goal of the proposed method is to provide a practical solution that bridges the gap between detailed spatial representation and computational efficiency, making it accessible for a wide range of applications for robotic manipulators.

3 Preliminaries

3.1 Definition of the Reference Frames

The kinematic model of a manipulator is based on the reference frames that define the poses of the robot links and joints. By using the kinematic model of the robot, the end-effector coordinates can be determined from the values of the joint variables. In this work, the definition of the reference frames is based on a set of rules that differs from the conventional approach, namely the Denavit-Hartenberg (DH) convention. This results also in a different mathematical notation. The selection of this approach stems from the way the kinematic model is used to generate the RS of a manipulator, as further deepened in the following sections. Indeed, it is necessary to have the invariance of reference frame $\mathcal{R}_i$ to the motion of joint j_i to compute all possible reachable points recursively, going from the last joint of the arm to the base. This requirement cannot be fulfilled using the DH convention, since the pose of a reference frame $\mathcal{R}_i$ changes as a result of the motion of the corresponding joint j_i. Furthermore, the rules adopted in this work align with those used in the URDF (Unified Robot Description Format) files of ROS, facilitating the implementation of the code for generating the point cloud, and making it more accessible and practical for the research community.

First, an additional naming is adopted for the links connected by a joint. As usual, the *i-th* joint j_i connects link l_{i-1}

and link l_i. Such links are also referred to as *parent* and *child* links of joint j_i, respectively.

Each reference frame $\mathcal{R}_i$ is fixed to the corresponding joint j_i, with its origin located at the beginning of the joint's child link. The only constraint for the orientation of its axes is that one of them must be coincident with the motion axis of the corresponding joint. The choice of the other ones is arbitrary and can be done in the most convenient way. The positioning of the origin also holds for the reference frames that are not representative of a joint, e.g., the origin of the fixed frame $\mathcal{R}_0$ is located at the beginning of link l_0, with arbitrary orientation of the axes.

Another important assumption has been made about the position and orientation of the reference frames during the motion of each joint. Indeed, as a result of the motion of the $i\text{-}th$ joint, only the reference frames following it along the kinematic chain, i.e., frames $\mathcal{R}_j$ with $j > i$, are affected by its motion. As a consequence, the reference frame $\mathcal{R}_i$ remains fixed, but any point defined in it is influenced by the joint motion. The homogeneous coordinates of a point defined in $\mathcal{R}_i$ after the motion of joint j_i can be computed as:

$$\tilde{\mathbf{p}}^i_{mov} = \mathbf{M}^i_i(\delta_i) \cdot \tilde{\mathbf{p}}^i, \tag{1}$$

where $\tilde{\mathbf{p}}^i \in \mathbb{R}^{4\times1}$ is the vector containing the homogeneous coordinates of the point in $\mathcal{R}_i$, $\mathbf{M}^i_i(\delta_i) \in \mathbb{R}^{4\times4}$ is the homogeneous transformation matrix defining the movement of joint j_i of an arbitrary quantity δ_i, and $\tilde{\mathbf{p}}^i_{mov} \in \mathbb{R}^{4\times1}$ is the vector representing the homogeneous coordinates in $\mathcal{R}_i$ of the moved point. The definition of matrix $\mathbf{M}^i_i$ depends on the type of the joint j_i: it is a pure-rotation homogeneous matrix if j_i is a revolute joint (and in this case δ_i is the rotation angle), whereas it is a pure-translation one if the joint is prismatic (and hence δ_i indicates the amount of the translation along the motion axis of the joint). In addition, $\mathbf{M}^i_i$ is defined taking into account that the axis of movement of the joint j_i coincides with one of the axes of the reference system $\mathcal{R}_i$, according to the choice made.

The relationship between two following reference frames $\mathcal{R}_{i-1}$ and $\mathcal{R}_i$ can be described using a homogeneous transformation matrix $\mathbf{T}^{i-1}_i \in \mathbb{R}^{4\times4}$, defined as:

$$\mathbf{T}^{i-1}_i = \left(\begin{array}{ccc|c} & \mathbf{R}_{RPY} & & \mathbf{t}^{i-1}_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right), \tag{2}$$

where $\mathbf{R}_{RPY} \in \mathbb{R}^{3\times3}$ is the roll-pitch-yaw (RPY) rotation matrix defining the orientation of the axes of $\mathcal{R}_i$ with respect to $\mathcal{R}_{i-1}$, while $\mathbf{t}^{i-1}_i \in \mathbb{R}^{3\times1}$ is the vector representing the origin of $\mathcal{R}_i$ in $\mathcal{R}_{i-1}$. The RPY representation is used because such angles are defined with respect to the reference frame $\mathcal{R}^{i-1}$. It must be noticed that all these homogeneous matrices do not include any variable representing the joint's motion. Indeed, they are defined starting from the geometric properties of the manipulator, setting all the joint variables equal to zero. Moreover, they are kept constant for all the computations. These matrices encode possible offsets and rotations that depend only on the definition of the reference frames.

Combining Eqs. 1 and 2, it is possible to compute the coordinates of a point in $\mathcal{R}_i$ with respect to $\mathcal{R}_{i-1}$ after the motion of joint j_i of the quantity δ_i. The resulting equation is:

$$\tilde{\mathbf{p}}^{i-1} = \mathbf{T}^{i-1}_i \cdot \mathbf{M}^i_i(\delta_i) \cdot \tilde{\mathbf{p}}^i, \tag{3}$$

where $\tilde{\mathbf{p}}^i \in \mathbb{R}^{4\times1}$ is the coordinates vector in $\mathcal{R}_i$ before the motion of joint j_i, $\tilde{\mathbf{p}}^{i-1} \in \mathbb{R}^{4\times1}$ is the coordinates vector in $\mathcal{R}_{i-1}$ after the motion, and the homogeneous matrices $\mathbf{M}^i_i(\delta_i)$ and $\mathbf{T}^{i-1}_i$ are as in Eqs. 1 and 2, respectively. An example of application of the rules illustrated in this section is provided in Example 1.

Example 1 Consider the RRP manipulator (R: Revolute joint, P: Prismatic joint) sketched in Fig. 1, together with a possible choice of the reference frames consistent with the rules previously illustrated. As an example, the coordinates of a point defined in $\mathcal{R}_2$ are computed with respect to $\mathcal{R}_1$ after the motion of joint j_2.

Since joint j_2 is a revolute joint rotating about the z-axis of $\mathcal{R}_2$, the homogeneous transformation matrix $\mathbf{M}^2_2(\delta_2)$ in

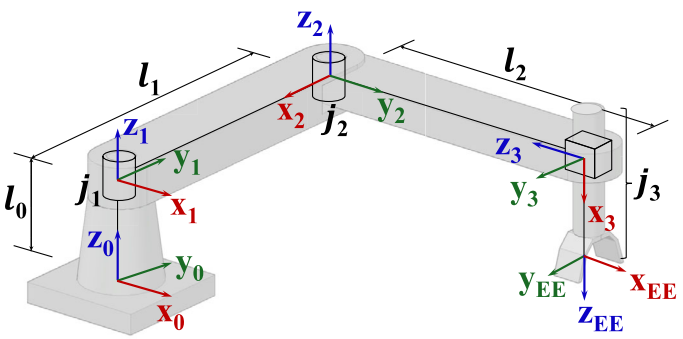


Fig. 1 Definition of the reference frames for a RRP robotic arm

Eq. 1 is given by:

$$\mathbf{M}_2^2(\delta_2) = \left(\begin{array}{ccc|c} cos(\delta_2) & -sin(\delta_2) & 0 & 0 \\ sin(\delta_2) & cos(\delta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right),$$

whereas the transformation matrix $\mathbf{T}_2^1$ in Eq. 2 is defined as:

$$\mathbf{T}_2^1 = \left(\begin{array}{ccc|c} cos(\frac{\pi}{2}) & -sin(\frac{\pi}{2}) & 0 & 0 \\ sin(\frac{\pi}{2}) & cos(\frac{\pi}{2}) & 0 & l_1 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right).$$

Let $\tilde{\mathbf{p}}^2$ be the homogeneous coordinate vector of the point of interest in $\mathcal{R}_2$. Its homogeneous coordinate vector in $\mathcal{R}_1$, after the motion of the joint j_2 of a quantity δ_2, can be computed as:

$$\tilde{\mathbf{p}}^1 = \mathbf{T}_2^1 \cdot \mathbf{M}_2^2(\delta_2) \cdot \tilde{\mathbf{p}}^2 = \left(\begin{array}{ccc|c} cos(\frac{\pi}{2}) & -sin(\frac{\pi}{2}) & 0 & 0 \\ sin(\frac{\pi}{2}) & cos(\frac{\pi}{2}) & 0 & l_1 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \cdot \left(\begin{array}{ccc|c} cos(\delta_2) & -sin(\delta_2) & 0 & 0 \\ sin(\delta_2) & cos(\delta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \cdot \tilde{\mathbf{p}}^2.$$

Remark 1 It must be noticed that the rules described in this section for the definition of the reference frames generally lead to a different formulation of the FK problem, if compared with other common conventions, like the DH one. Hence, the solutions obtained starting from the same joint values can be different. This is essentially due to the freedom of selecting the axes orientation, and to possible additional offsets encoded in matrices $\mathbf{T}_i^{i-1}$. It is possible to prove that in specific conditions the convention used in this work is equivalent to DH, thus providing in this case the same coordinates of the EE for the same joint values. Such proof is out of the scope of this paper, and for this reason, is not provided. The key point for this paper is that the reachability of a point depends only on the manipulator's geometric properties. Using a different reference frame definition can lead to different joint configurations needed to reach the target, but it does not affect its reachability characteristics.

3.2 Selection and Meaning of the Representative Point

To generate the point cloud representing the RS of a manipulator, a representative point must be selected, and its coordinates must be expressed with respect to a reference frame. The main characteristic of such point is that it must correspond to a position that the manipulator can actually reach with at least one orientation.

In this paper, we have chosen a point representing the wrist. Specifically, it is the origin of the reference frame attached to the last joint of the wrist. Its coordinates, expressed with respect to the reference frame associated to the last joint of the arm, are computed considering all the wrist links aligned to each other and to the last link of the arm. Figure 2 shows the representative point of an anthropomorphic manipulator equipped with a spherical wrist. It must be noticed that the representative point selection is independent of the number of joints constituting the kinematic chain.

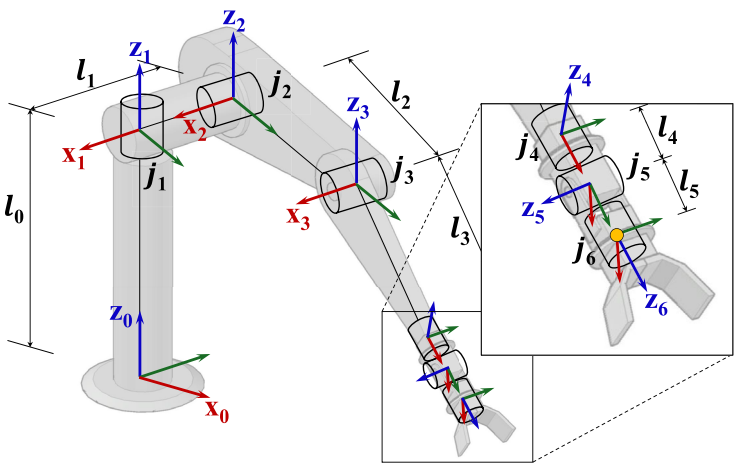


Fig. 2 Selection of the representative point (in yellow) for a manipulator constituted by an anthropomorphic arm and a spherical wrist. The definition of the reference frames follows the rules illustrated in Section 3.1

The choice made for the representative point is due to the connection of the EE to a manipulator's wrist. Indeed, the EE is typically attached to the child link of the last joint of the wrist, meaning that any representative point of the first will remain at a fixed distance from the latter. Such fixed distance can be considered as an offset, hence neglected while building a representation of the RS of the manipulator. Consequently, the origin of the reference frame attached to the last joint of the wrist can be chosen as the representative point, simplifying the analysis without affecting the accuracy.

The choice of considering the wrist links aligned, when defining the coordinates of the representative point, is related to the contribution given by the wrist joint motion. In a typical kinematic chain configuration, the motion of the wrist can significantly affect the position of the EE, and not only its orientation. However, it can only reduce the distance between the EE and the arm links with respect to the situation in which the wrist joints are aligned. By aligning the wrist links and moving only the arm joints, a conservative representation of the manipulator's RS can be created. A comparison of the point clouds generated with aligned wrist links and moving only the arm joints with those obtained moving all joints revealed that the outermost points of both clouds were aligned. This indicates that the coordinates of the representative point, when the motion of the wrist joints is considered during the generation of the point cloud, have a negligible impact on the calculation of the ellipsoid equation that represents the external boundary of the RS.

4 The Proposed Approach

The theoretical and implementation details of the proposed method are discussed in this section. First, a description of the procedure used to generate the point cloud is provided. Then, the two techniques used to solve the optimization problem to obtain the ellipsoid equation are illustrated: the first one is formulated as a proper optimization problem, while the second one is based on PointNet.

4.1 Generation of the Reachable Points

The first step to obtain a description of the RS of a robot is to determine which points of the 3D space belong to it. In Section 2, the main methods used in other relevant works to tackle this task have been presented. To overcome their main limitations previously discussed, we designed a procedure to obtain the required points by exploiting only the kinematic model of the arm, and using each joint value of the corresponding span only once, in order to achieve an efficient solution having a wide applicability.

The starting point of the proposed procedure is given by the construction of the kinematic model of the manipulator. It is built using the definition of the reference frames detailed in Section 3.1. Furthermore, two additional inputs are required: the desired number of samples for each joint, denoted as M, and the coordinates of the representative points in the reference system located at the last joint of the arm. While it is true that a higher value of M produces a larger number of points, hence a more precise description of the RS, it must be noticed that a physical upper bound does exist. Such a limit is determined by the encoders' resolution mounted on the joints of the manipulator. Encoders are characterised by a finite resolution, which defines the smallest distinguishable difference between two positions. If the number of motion samples M is set too high, the difference between successive samples may fall below the resolution of the encoder, making them indistinguishable. Furthermore, when considering all the joints, this can result in an excessive number of generated points, i.e., greater than the actually reachable positions. Consequently, this introduces unnecessary overhead and computational inefficiencies. Due to the fact that this upper bound is dependent on the encoder's resolution, it is not set within the algorithm, but it is left to the user to limit M appropriately.

In the most common robotic structures, the arm of the manipulator is typically composed of three joints. For this reason, the following description of the proposed procedure considers an arm constituted by three joints, although it is possible to extend the approach to redundant structures where the arm is made by n joints. Given the considerations made in Section 3.2 regarding the wrist joints, together with the definition of the representative point, only the arm joints' motion is considered in order to generate the point cloud.

A key point to clarify is the order in which the joints are taken into account in the procedure. They are considered in a reversed order, starting from the last one of the arm and then moving backwards. This is possible thanks to the invariance of the reference system $\mathcal{R}_i$ with respect to the motion of joint j_i to which it is attached.

Starting from the last arm joint, i.e., joint j_3, all its movements are sampled creating a span of M equally spaced values of δ_3, taking into account also the joint constraints. Equation 1 is then applied with $i = 3$, to all the considered values, to determine for each of them the resulting $\tilde{\mathbf{p}}^3_{mov}$, including the homogeneous coordinates of the representative point in $\mathcal{R}_3$ after the joint motion, starting from $\tilde{\mathbf{p}}^3$. All the computed solutions are then arranged as columns of matrix $\tilde{\mathbf{P}}^3 \in \mathbb{R}^{4\times M}$. In this way, all the possible locations of the representative point in $\mathcal{R}_3$, resulting from the motion of joint j_3 only, are computed and stored in a compact manner. Moreover, having all the points stored in matrix $\tilde{\mathbf{P}}^3$ allows to compute their corresponding homogeneous coordinates in $\mathcal{R}_2$, easily by

pre-multiplying such matrix by the homogeneous transformation matrix $\mathbf{T}_3^2$ as defined in Eq. 2. The columns of the resulting matrix $\tilde{\mathbf{P}}_3^2 \in \mathbb{R}^{4\times M}$ contain then the homogeneous coordinates in $\mathcal{R}_2$ of the representative point after the motion of j_3.

Proceeding backwards to joint j_2, its movements are sampled, creating a span of M equally spaced values of δ_2, which are used to solve (1). This time, instead of considering the coordinates of a single point as $\tilde{\mathbf{p}}^2$, the entire matrix $\tilde{\mathbf{P}}_3^2$ is used for the computation. The result is a matrix of M columns, which contains the homogeneous coordinates in $\mathcal{R}_2$ of all the points in the columns of $\tilde{\mathbf{P}}_3^2$, after the motion of joint j_2. By iterating on all the values of δ_2, a total of M matrices are obtained, and their columns are arranged together in a single matrix $\tilde{\mathbf{P}}^2$. Such a matrix is composed of a total of M^2 columns, representing all the possible homogeneous coordinates of the representative point due to the motion of joints j_3 and j_2. The corresponding coordinates in $\mathcal{R}_1$ are computed by pre-multiplying $\tilde{\mathbf{P}}^2$ by $\mathbf{T}_2^1$, hence obtaining matrix $\tilde{\mathbf{P}}_2^1 \in \mathbb{R}^{4\times M^2}$.

The computations done for joint j_2 are repeated for all the remaining arm joints, proceeding backwards. In the case of a non-redundant arm, having three joints, the final result is matrix $\tilde{\mathbf{P}}^0 \in \mathbb{R}^{4\times M^3}$, whose columns contain the homogeneous coordinates in $\mathcal{R}_0$ of the representative point in all the possible locations resulting from the motion of all the arm joints. In the most general case, including redundant arms having n joints, with $n > 3$, matrix $\tilde{\mathbf{P}}^0$ has a total of M^n columns. The complete iterative procedure described here is summarized in Algorithm 1.

Algorithm 1 Point cloud generation.

Input: robot's kinematic model, M, $\tilde{\mathbf{p}}^n$
Output: point cloud representing the robot's RS

$\tilde{\mathbf{P}}_{\mathbf{all}} \leftarrow \tilde{\mathbf{p}}^n$ # Init. the list of all points $\tilde{\mathbf{P}}_{\mathbf{all}}$ with $\tilde{\mathbf{p}}^n$
$\tilde{\mathbf{P}}_{\mathbf{all}}$.toMatrix() # Convert the list to a Matrix
for i from n to 1 **do** # Consider all joints going backwards
  $span_i \leftarrow$ linspace($j_{i,lb}$, $j_{i,ub}$, M) # Samples j_i possible movements
  $\tilde{\mathbf{P}}_{\mathbf{new}} \leftarrow$ empty_list
  for δ_i in $span_i$ **do** # Consider all M samples of j_i
    $\tilde{\mathbf{P}} \leftarrow \mathbf{T}_i^{i-1} \cdot \mathbf{M}_i^i(\delta_i) \cdot \tilde{\mathbf{P}}_{\mathbf{all}}$
    $\tilde{\mathbf{P}}_{\mathbf{new}}$.append($\mathbf{P}$) # Add the result to the list
  end for
  $\tilde{\mathbf{P}}_{\mathbf{all}} \leftarrow \tilde{P}_{\mathbf{new}}$ # Update the list of all points
  $\tilde{\mathbf{P}}_{\mathbf{all}}$.toMatrix() # Convert the list to a Matrix
end for

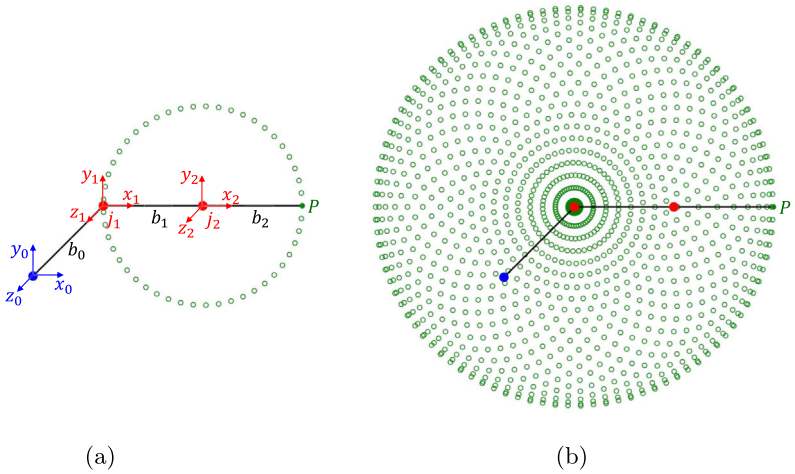
The proposed approach for generating a point cloud representing all potential positions of the selected representative point is also applicable to mobile manipulators. However, also in this context, the point cloud generation process is executed by considering only the motion of the arm joints, thereby neglecting the additional three DOFs introduced by the mobile base. Consequently, the resultant set of points is representative of those positions that can be reached from the current pose of the mobile base.

Using this procedure, it is possible to obtain a 3D point cloud composed of a total of M^n elements, whose homogeneous coordinates are the columns of matrix $\tilde{\mathbf{P}}^0$. It is possible to notice that each joint value is used only once to compute all its contributions to the positioning of the representative point. Moreover, the use of matrices to store the point coordinates at each intermediate step, together with the possibility of exploiting matrix products for the computation, can lead to a very efficient software implementation of the proposed procedure, as discussed later in Section 5.1. The computational complexity of the algorithm can be analyzed in terms of the number of matrix multiplications and element-wise multiplications performed. The number of matrix multiplications required to process all possible M samples of the $i-th$ joint j_i is constant and is $\mathcal{O}(M)$. Since this operation is repeated for each of the n joints of the manipulator, the overall complexity regarding the matrix multiplications is $\mathcal{O}(nM)$. In the case of element-wise multiplications, the worst-case scenario occurs in the final step of the algorithm. At this stage, for each of the M samples of the last joint, a total of $\mathcal{O}(M^{n-1})$ multiplications are performed to deal with the M^{n-1} points previously computed. Consequently, the overall complexity of the algorithm in terms of element-wise multiplications is $\mathcal{O}(M^n)$. Note that M^n corresponds to the total number of generated points.

Figure 3 sketches the steps of the procedure, applied to a 2R planar manipulator, whose joints can rotate unconstrained. The two revolute joints and their reference frames are reported in red, while the fixed frame R_0 is in blue. The representative point P and all its possible locations are displayed in green. Link b_0 is fixed, and it is rotated of 45° with respect to axis x_0. Figure 3a shows the result of the first step, hence the possible locations of P resulting from the rotation of joint j_2 only. Each point obtained is then rotated as a result of the motion of j_1. The result of this last iteration is shown in Fig. 3b, which also represents the RS estimate for the considered manipulator.

4.2 Optimization-based Approach

Once the point cloud has been obtained, it can be used to build a mathematical model of the manipulator's RS based



(a) (b)

Fig. 3 Example of the procedure used to obtain the set of points representing the RS applied to a 2R planar manipulator. (a) Reachable points obtained rotating only joint j_2. (b) Result obtained rotating each previous point around joint j_1

on the equation of an ellipsoid, which can be expressed as:

$$\left(\frac{x-x_c}{A}\right)^2+\left(\frac{y-y_c}{B}\right)^2+\left(\frac{z-z_c}{C}\right)^2=1, \tag{4}$$

where $x_c, y_c, z_c \in \mathbb{R}$ are the coordinates of the center, and $A, B, C \in \mathbb{R}^+$ are the lengths of the semi-axes. Equation 4 can be rewritten in matrix form as:

$$(\mathbf{x}-\mathbf{c})^T\,\mathbf{E}\,(\mathbf{x}-\mathbf{c})=1, \tag{5}$$

where $\mathbf{x}=[x,y,z]^T$, $\mathbf{c}=[x_c,y_c,z_c]^T$, and $\mathbf{E}\in\mathbb{R}^{3\times3}$ is a positive definite matrix defined as:

$$\mathbf{E}=\begin{bmatrix}\frac{1}{A^2} & 0 & 0\\ 0 & \frac{1}{B^2} & 0\\ 0 & 0 & \frac{1}{C^2}\end{bmatrix}. \tag{6}$$

It is possible to notice that Eqs. 4 and 5 are completely defined by the 6 parameters x_c, y_c, z_c, A, B, C. For this reason, they can be used as decision variables of an optimization problem.

The objective function of the optimization problem must be designed properly to find an optimal solution for an ellipsoid that includes all the points $\mathbf{p}$ belonging to the point cloud $\mathcal{P}$. Moreover, the dimension of the semi-axes A, B, and C must be bounded. This latter aspect is important to avoid that the ellipsoid includes points of the 3D space that do not belong to the RS of the manipulator.

Let $\mathcal{P}=\{\mathbf{p_1},\mathbf{p_2},...,\mathbf{p_m}\}$ be the finite set of 3D points representing the point cloud obtained using the procedure detailed in Section 4.1.

Since a point $\mathbf{x}\in\mathbb{R}^{3\times1}$ belongs to the inner part of an ellipsoid or to its surface if and only if the following equation holds:

$$(\mathbf{x}-\mathbf{c})^T\,\mathbf{E}\,(\mathbf{x}-\mathbf{c})\le1, \tag{7}$$

it is possible to define a set $\mathcal{P}_{cont}$ containing only the points $\mathbf{p}\in\mathcal{P}$ that satisfy (7) as:

$$\mathcal{P}_{cont}=\left\{\mathbf{p}_i\in\mathcal{P}\mid(\mathbf{p}_i-\mathbf{c})^T\mathbf{E}(\mathbf{p}_i-\mathbf{c})\le1\right\}. \tag{8}$$

The number of points $\mathbf{p}\in\mathcal{P}$ that belongs to the ellipsoid can be defined as $|\mathcal{P}_{cont}|$, where $|\cdot|$ denotes the cardinality of the set. Since $\mathcal{P}_{cont}\subseteq\mathcal{P}$, it is possible to deduce that $|\mathcal{P}_{cont}|$ is bounded:

$$0\le|\mathcal{P}_{cont}|\le|\mathcal{P}|. \tag{9}$$

Using the definition of $\mathcal{P}_{cont}$ in Eq. 8, the optimization problem can then be defined as:

$$\begin{aligned}\underset{x_c,y_c,z_c,A,B,C}{\arg\min}\quad & -|\mathcal{P}_{cont}|+\frac{4}{3}\pi ABC\\ \text{subject to:}\quad & A>0,\ B>0,\ C>0,\end{aligned} \tag{10}$$

where $\frac{4}{3}\pi ABC$ is the volume of the ellipsoid.

It is possible to notice that the objective function contains $-|\mathcal{P}_{cont}|$ instead of just $|\mathcal{P}_{cont}|$. The minus sign is needed to try to include all the points of $\mathcal{P}$ in $\mathcal{P}_{cont}$, hence to reach the maximum value of $|\mathcal{P}_{cont}|$, following a minimization fashion. In other words, since $|\mathcal{P}_{cont}|$ is limited, we are converting a maximization problem into a minimization one while maintaining the same solution. The contribution of the volume of the ellipsoid is needed to bound the overall dimension of the ellipsoid itself. The use of the designed objective function, which aims to minimize the product of A, B and C, while trying to maximize $|\mathcal{P}_{cont}|$, results in an ellipsoid that contains the point cloud representing the RS of the robotic arm, while avoiding to include too many points that cannot be effectively reached by the EE.

The optimization problem defined in Eq. 10 allows to obtain the ellipsoid equation representing the external

envelop of the RS of the robotic arm. Changing the objective function, e.g., introducing a weight or a score w_{p_i} for each point, it is possible to obtain an ellipsoid that encloses only the optimal points according to some criteria. It is worth to notice that, due to the minimization fashion of the optimization problem, only the points with the lowest values of w_{p_i} will be enclosed in the resulting equation. Consequently, the optimality criteria should be defined accordingly. The minimization problem in Eq. 10 can be rewritten as:

$$\begin{aligned} &\underset{x_c,y_c,z_c,A,B,C}{\arg\min} \quad \sum_{\mathbf{p_i}\in\mathcal{P}_{cont}} w_{p_i} + \frac{4}{3}\pi ABC \\ &\text{subject to:} \quad A > 0,\ B > 0,\ C > 0. \end{aligned} \tag{11}$$

Changing the objective function allows to achieve a high level of customization, if needed. Indeed, the resulting equation can be used to represent the entire manipulator's RS, or, alternatively, to construct an easy representation of the points that allow to reach the best values of a desired quality metric. Different methods for the solution of the optimization problem will be compared in Section 5.3.

4.3 PointNet-based Approach

The alternative method that we propose to find the parameters of the ellipsoid representing the RS of a manipulator is based on the use of a Convolutional Neural Network (CNN). Indeed, ML models can be used to solve optimization problems, since the optimization process is central for the training phase.

The input of the proposed CNN is the point cloud obtained as a result of the procedure illustrated in Section 4.1. Dealing with point clouds can be difficult, since they are irregular and unordered, and for this reason, they often challenge traditional models [40]. Even though some approaches propose to convert them to regular structures like voxels or meshes, directly using point clouds simplifies learning and avoids heavy data transformations, enhancing efficiency [41].

The first architecture created to work directly with point clouds is PointNet [40], whose high-level architecture is illustrated in Fig. 4. Originally designed for object classification, part segmentation, and scene semantic parsing, PointNet exhibits the ability to extract features from the point cloud efficiently. Features extraction is achieved mainly by exploiting 2D convolutional layers. They are essential to capture spatial relationships between points, and they also limit the number of learnable parameters thanks to the weight-sharing property. The two alignment networks, namely "Input transform" and "Feature transform" shown in Fig. 4, ensure the invariance against geometric transformations and the ability to learn hierarchical features from unordered data. The extracted relationships are then aggregated by the max pooling layer, whose result constitutes the global features of the point cloud. The output of the feature extractor is further processed by a Multilayer Perceptron (MLP) network to compute the desired output.

Motivated by PointNet's effectiveness and efficiency in extracting relevant features from point clouds, we trained this network to solve our parameter estimation problem, hence to find the ellipsoid parameters. In addition, we analyzed the impact of the number of parameters on the final results. Indeed, by changing the sizes of convolutional layers and the number of neurons in the MLP, we created three alternative architectures that differ only in the number of parameters. Their performances will be illustrated and experimentally compared in Section 5.4.

4.3.1 Characteristics of the Training Dataset

To train and test the end-to-end architectures described before, a custom dataset of point clouds representing ellipsoids has been created. The generation of each point cloud and its corresponding label is achieved through the implementation of an iterative two-phase procedure. In the initial phase, the six parameters of the ellipsoid equation defined in Eq. 4, i.e., the label, are randomly sampled from a uniform

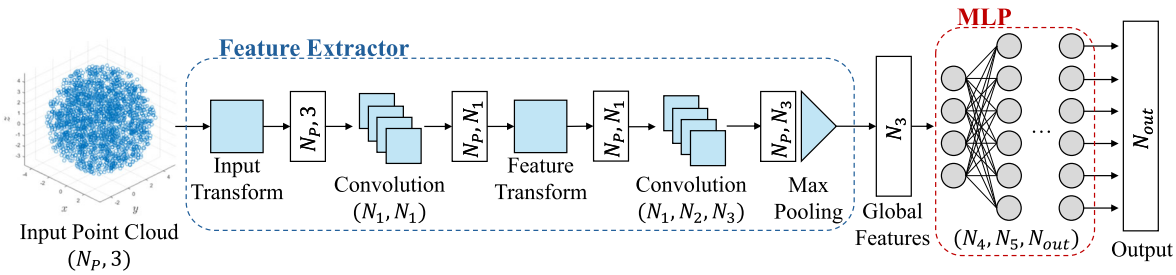


Fig. 4 A high-level architecture of PointNet. The input point cloud has a total of N_p 3D points. Outputs of each layer are shown as white rectangles, containing also their dimension. The number of convolutional layers, and the number of neurons are indicated by N_i, while the dimension of the output N_{out} can change depending on the application

distribution between 0 and 1. Once the equation is established, the second phase generates 3D points within a large cube, retaining only those that satisfy the newly generated ellipsoid equation. This second step is repeated until the desired number of points is obtained, thus assigning them to the generated label. Employing this two-phase approach, a total of 10,000 point clouds are generated, each of them comprising 2,048 3D points.

All 6 parameters of each point cloud have been generated using a uniform distribution between 0 and 1. This choice was driven by the way ML models learn, together with the mechanical properties of manipulators commonly used in the research field. Researchers demonstrated that MLPs excel at learning from uniform and Gaussian distributions because they align well with the continuous functions MLPs approximate [42, 43]. The smoothness and predictability of Gaussian distributions, and the simplicity of uniform distributions, make it easier for MLPs to effectively capture and model the underlying patterns. Moreover, the RS of robotic arms, particularly those mounted on mobile manipulators, typically ranges from a few centimeters to several meters. Consequently, the total span of the reachable region is comparable. The ellipsoid that encloses this space has axes extending slightly beyond 1 meter in length.

5 Experimental Evaluation

This section presents the results of the experiments that validate the proposed methodologies. Firstly, the effectiveness of the point cloud generation method proposed in Section 4.1 is evaluated, with a comparison made against more traditional approaches based on FK and IK computation. Subsequently, the two methodologies for estimating the equation of the ellipsoid representing the RS of a manipulator (outlined in Sections 4.2 and 4.3) are evaluated. The evaluation is conducted using a range of metrics to assess the time required and the accuracy of the results. For each of the two categories of proposed approaches, different algorithms and/or models are analysed and compared. All the computations for every test have been made using a laptop computer equipped with an Intel i7-1165G7 processor [44] and running Ubuntu 20.04 with Python 3.8.10. The use of a high-performance GPU has been avoided, in order to check the feasibility of applying this approach to low-resource robots (i.e., to robots not equipped with a GPU). The code used for the experimental evaluation is available in [45].

5.1 Point Cloud Generation Performances

A comparative analysis of the various methodologies employed for the generation of the point cloud representing the RS is provided here. The performance of the methodology presented in Section 4.1 is compared with two alternative approaches based on the solution of the FK and IK problem, whose details have been illustrated in Section 2.2. The complexity analysis presented in Section 4.1 is further validated by the point-generation time analysis provided in this section. The point clouds were generated using the kinematic model of the UR5e robot from Universal Robots [46] as a reference. Figure 5 illustrates the average time required by each method for varying numbers of samples. To ensure consistency, the

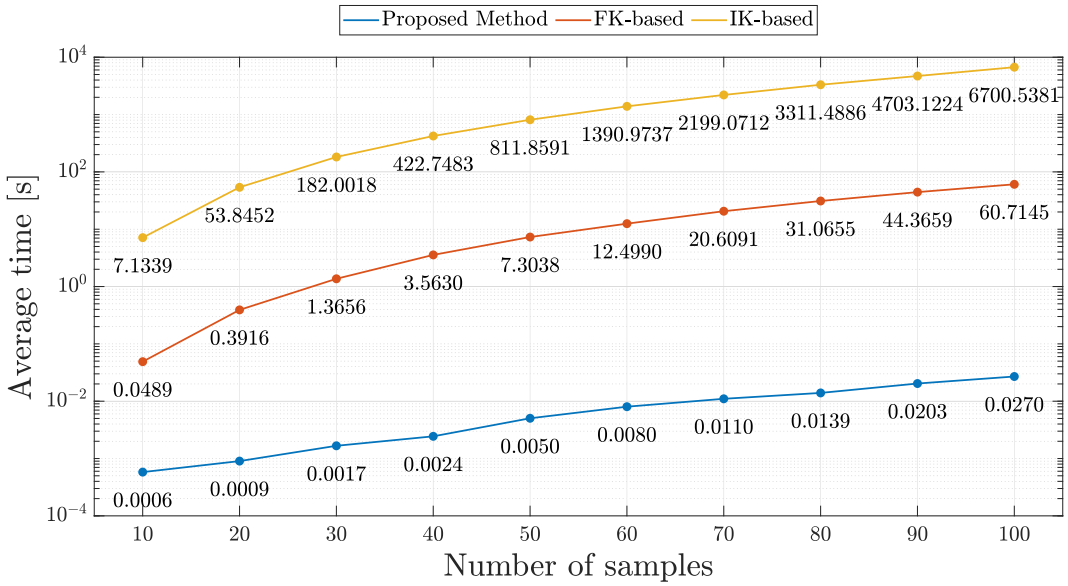


Fig. 5 Comparison of the times needed to generate a point cloud using different methods. The Y-axis (average time) is on a logarithmic scale for better visualization

reported values are the mean of ten trials conducted for each number of samples.

The method proposed for the generation of point clouds, comprising a variable number of elements ranging from 10^3 to 10^6, requires a processing time between 0.6 and 27 ms, as depicted in Fig. 5. On a logarithmic scale, the observed linear scaling pattern shows that the proposed method can effectively handle increasing numbers of points without a drastic rise in computational cost, indicating its superior scalability compared to the alternative methods.

The discrepancy in performance between the proposed method and the FK-based one can be attributed to the number of times a given joint value is considered. In the FK-based method, a joint value is employed multiple times to compute all the potential combinations, while in the proposed method, each value is used only once, optimizing the computation process. Moreover, the proposed method directly generates the desired point cloud, bypassing the need for pre-calculated sets of elements, which makes it inherently more efficient. This is particularly evident when compared to the IK-based method, where the existence of a solution to the IK problem must be verified for each element in the pre-calculated set. For the sake of building a comparison between the proposed method and the IK-based one, the pre-calculated set used for the latter comprised a 3D grid of uniformly spaced samples. The length of each side of the grid spanned twice the length of the robot in its extended state, ranging from the negative to the positive value. The total number of points of this pre-computed set corresponds to the values shown on the x-axis in Fig. 5.

The proposed method achieves remarkable efficiency by eliminating explicit loops in the generation process and leverages the power of vectorised operations. This enables the simultaneous execution of multiple computations, significantly reducing the computational overhead that results from sequential processing. The replacement of iterative loops with vectorised operations not only streamlines the computation process, but also ensures effective scaling as the number of samples per joint increases, thereby demonstrating the performance-oriented optimization of the method.

5.2 Definition of the 3D-IoU

One of the metrics we used to evaluate the performance of the methods for the estimation of the ellipsoid equation enclosing the RS is the three-dimensional Intersection over Union (3D-IoU) [47]. Initially defined to be used for object detection and segmentation tasks, this metric extends the traditional two-dimensional IoU metric [48], which measures the overlap between two-dimensional shapes, to the three-dimensional space, where the volumetric overlap between three-dimensional volumes is assessed. It can be defined as:

$$3D\text{-}IoU = \frac{V_{intersection}}{V_{union}}, \tag{12}$$

where $V_{intersection}$ is the volume of the intersection of the 3D shapes, and V_{union} is the total volume covered by both of them. The 3D-IoU provides a comprehensive measure of spatial overlap, which is essential for applications requiring precise localization and understanding of objects in three-dimensional space [49, 50]. In the following sections, this metric is employed for a comprehensive assessment of the similarity between the generated ellipsoid equation and the ground truth one.

$V_{intersection}$ and V_{union} have been computed leveraging the Monte Carlo integration (MCI) method [51]. The fundamental principle of MCI is to use random sampling to estimate the integral of a function f over a domain D, approximating it as the mean of the function values at randomly selected points within that domain, scaled by the volume of D. This can be expressed as:

$$\int_D f(\mathbf{x})d\mathbf{x} \approx V\frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i), \tag{13}$$

where V is the volume of $D \subseteq \mathbb{R}^m$ and N is the total number of samples $\mathbf{x}_i \in \mathbb{R}^m$ generated over D.

The problem of computing the volume V of the sampling region in Eq. 13 can be solved by the MCI method by leveraging a sampling region that encompasses the one of interest, but whose volume can be easily determined. The value of V can then be computed approximately as:

$$V = \int_\Omega d\mathbf{x} \approx V_B\frac{N_\Omega}{N}, \tag{14}$$

where V_B is the volume of the region $B \subset \mathbb{R}^m$ defined such that $\Omega \subset B$, N is the total number of random samples generated over B, and N_Ω is the number of generated samples belonging to Ω.

To compute $V_{intersection}$ and V_{union} using Eq. 14, we defined a box-shaped region $B \subset \mathbb{R}^3$ encompassing both ellipsoids, and we generated 10 million sampling points with a uniform distribution inside it. Finally, the desired volumes are computed as:

$$V_{intersection} = V_B\frac{N_{intersection}}{N}, \tag{15}$$

$$V_{union} = V_B\frac{N_{union}}{N}, \tag{16}$$

where V_B is the volume of the box-shaped region B, N is the total number of sampling points generated, $N_{intersection}$ is

Table 1 Configuration parameters used for the PS algorithm

Name	Value	Description
delta	0.1	Distance of the initial patterns from the initial search point
rho	0.1	Delta reduction factor for unsuccessful moves
step_size	0.1	Size of the step following the promising direction

the total number of samples that satisfy both ellipsoids equations, namely belonging to their intersection, and N_{union} is the number of points satisfying either one of the two equations, namely belonging to their union.

5.3 Optimization Problem Evaluation

In this section, different methods are applied and compared to solve the optimization problems outlined in Section 4.2 for the computation of the ellipsoid's parameters. The effectiveness of these methods is evaluated through a series of tests considering three key parameters: (i) the average time required to find the optimal solution, (ii) the relative percentage error of each parameter, and (iii) the 3D-IoU defined in Section 5.3. The experimental validation is conducted using Python 3.8.10 and the pymoo library [52]. The results presented in this section are obtained by comparing the outcomes of the optimization-based approach with the labels of a benchmark set containing 1,000 point clouds. This dataset was generated following the same rules outlined in Section 4.3.1. The choice of using such a benchmark set is motivated by the need to have the parameters of the ellipsoid equation that encloses the corresponding point cloud. However, it is not common for manipulator manufacturers to provide an analytical representation of the RS.

Three optimization algorithms are employed: Pattern Search (PS) [53], Genetic Algorithm (GA) [54], and Particle Swarm Optimization (PSO) [55]. For both GA and PSO, the default hyperparameter settings provided by the pymoo library are utilized, as reported in [52], while for the PS algorithm, some hyperparameter values are set to optimize performance for the given problem, and to better compare it with the others. These customized values, which deviate from the default settings, are reported in Table 1. The termination condition of all algorithms is based on the value of the objective function. If the improvement is less than 10^{-6}, the algorithm is ended.

The starting point of the optimization process x_0, which is provided to all algorithms, is identical to ensure a consistent basis for comparison. It consists of two components: an initial estimate of the ellipsoid's center and an initial guess of the axis lengths. The initial approximation of the center is calculated as the mean point of the point cloud, with its coordinates being the average of all points' coordinates in the cloud. For the axis lengths, an initial value of 1 is assigned to each axis. By starting from this initial point in the search space, the algorithms are initialized in a balanced manner, promoting an unbiased exploration of potential solutions [56, 57]. This ensures that no algorithm benefits from more favorable initial conditions, allowing for a fair and reliable comparison of their performance.

In order to provide an accurate assessment of the overall performance of the optimization algorithms, it is essential to analyze both Figs. 6 and 7 together, showing the average time required and the relative percentage errors, respectively. In particular, Fig. 7 shows both the mean and the median relative percentage errors for each parameter of the ellipsoid to facilitate a more accurate understanding of the actual performance achieved. Indeed, the median values are less sensible to the presence of outliers, i.e., those cases where the solution found is highly imprecise in comparison with the ground truth. Consequently, the median values are more representative of the results obtained with respect to the mean values. From Figs. 6 and 7, it can be observed that all the algorithms demonstrate a notable level of effectiveness, as proven by the low median relative percentage error (less than 1%) and the relatively short computation times (few seconds or less) in all cases. Upon closer examination, the PS algorithm exhibits the shortest average convergence time, with a mean computation time of only 370 ms. However, its median relative percentage errors are the highest, and it is also sensitive to outliers, as evidenced by the mean errors, particularly with regard to the size of the ellipsoid axes. Conversely, the slowest algorithm is GA, with a computational time of 2.24 seconds. However, its median performance is comparable to that of the PS algorithm. Furthermore, it is particularly susceptible to outliers in the computation of the ellipsoid's center coordinates. The optimal balance is achieved by the PSO algorithm, which requires 1.48 seconds on average to identify the opti-

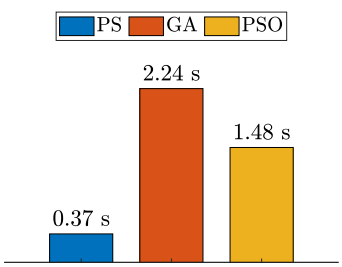


Fig. 6 Average time required by each optimization algorithm to find the optimal solution

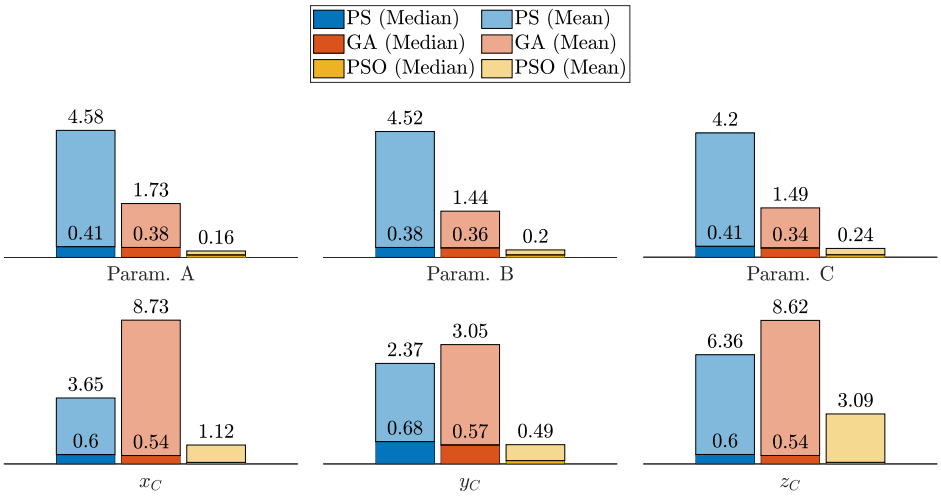


Fig. 7 Comparison of the relative percentage errors for the predictions obtained with the different optimization algorithms. Values are reported in [%]. The median errors for the PSO algorithm are all $\leq 0.1\%$

mal solution, while demonstrating a very high accuracy and robustness, denoted by the very low mean and median percentage errors.

The performance of the different methods is also highlighted by the mean values of the 3D-IoU metric, shown in Fig. 8. All algorithms achieve an average 3D-IoU exceeding 93% for both the mean and median values. Among them, the PSO algorithm exhibits the most accurate and robust results. Figure 9 provides a visual representation of the performance, showing the projections on the XY, XZ, and YZ planes of both the ground truth and the estimated ellipsoid. The errors affecting the predicted parameters are also highlighted by the colors of the overlapped ellipsoids. A visible portion of the red surface indicates a difference in the predicted shape.

5.4 PointNet Evaluation

The alternative approach, proposed in this study to estimate the parameters of the ellipsoid equation, leverages the feature extraction capabilities of the PointNet model, when applied to point cloud data. As anticipated in Section 4.3, in order

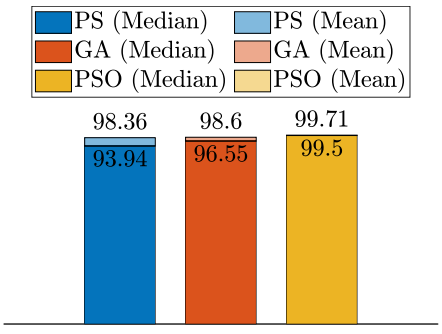


Fig. 8 Comparison between the different optimization algorithms using the 3D-IoU as metric. Values are reported in [%]

to evaluate the influence of the total number of learnable parameters on the model's performance, we designed and implemented three distinct architectures that vary solely in the number of learnable parameters, while maintaining an identical structure in terms of layer sequence and type. These architectures are referred to as the "Full", "Small", and "Tiny" models, reflecting the relative scale of their parameterization. They are tested and compared using the same metrics used for the evaluation of the optimization algorithm in Section 5.3: (i) the average prediction time, (ii) the relative percentage error, and (iii) the 3D-IoU.

The Full model used in this study is identical to the original PointNet architecture proposed by Qi et al. in [40], and trained for the comparison with the smaller, reduced-parameter models. The Small model is constructed by halving the number of convolutional layers composing the different parts of the original network. In turn, the Tiny model is obtained by further reducing the number of layers in the Small model by half. Figure 10 presents a comparison of the total number of learnable parameters for each model: the Full model is made of 2,424,643 parameters, the Small model has 613,283 parameters, and the Tiny model is characterized by 182,227 parameters. This comparison highlights the significant reduction in model complexity across the three architectures.

The training process for the Full, Small, and Tiny PointNet models is conducted under the same conditions to enable a fair comparison. The dataset consists of 10,000 point clouds, split into 80% for training and 20% for validation. Each model is trained using the Adam optimizer, with a variable learning rate. This hyperparameter is subject to dynamic regulation through a callback function, with the adjustment being based on the validation loss. In the absence of a decrease in validation loss over a period of 25 epochs, the learning rate undergoes a reduction by a factor of 10. The initial value is set to 10^{-3}, with the possibility of reducing

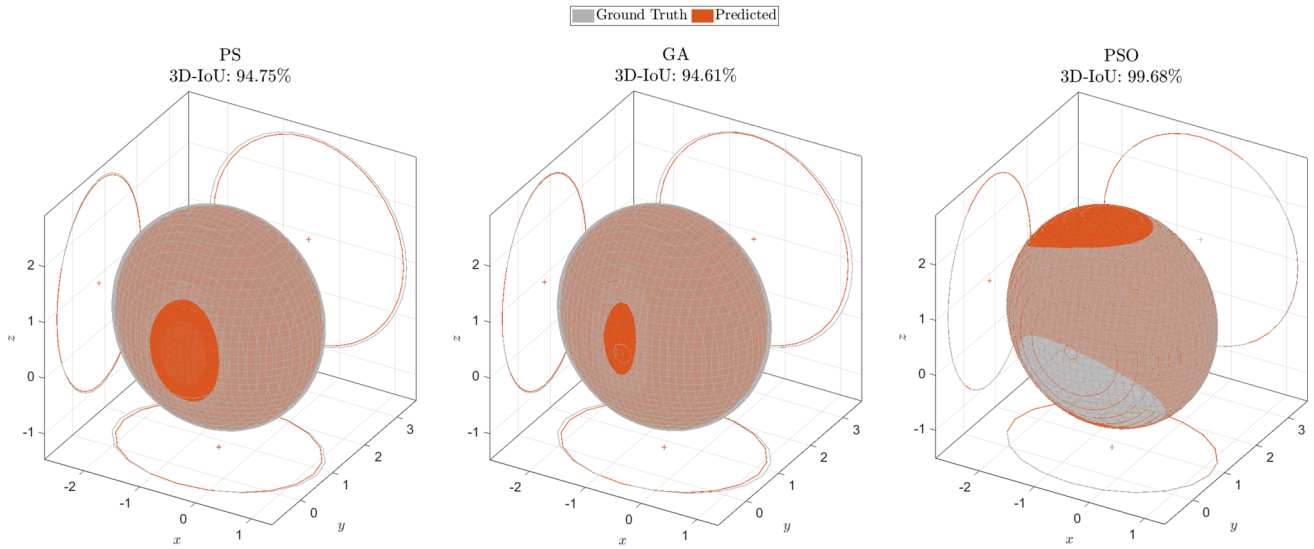


Fig. 9 Visual representation of 3D-IoU performance for each optimization algorithms. The orange ellipsoids show predicted shapes overlaid on the gray ground truth, with projections on the XY, XZ, and YZ planes highlighting spatial discrepancies and illustrating the accuracy of the generated ellipsoids

it to as low as 10^{-6}. Training is carried out over 100 epochs with a batch size of 16, selected to manage computational resources effectively. The loss function used is the Mean Squared Error (MSE), appropriate for the regression task. As part of data pre-processing, feature normalization is performed on the point cloud coordinates, enhancing the training stability. This setup is applied across the Full, Small, and Tiny models to isolate the effect of the number of learnable parameters on model performance. The training processes for the Full, Small and Tiny models are visualized in Fig. 11a, b and c, respectively.

The first evaluation metric employed to compare the various models is the mean prediction time. The experiments are conducted on the same dataset used to evaluate the performance of the optimization algorithms, which consists of a total of 1,000 point clouds. Figure 12 shows the average prediction times for the Full, Small, and Tiny PointNet models. Despite the significant reduction of the number of learnable parameters–ranging from about 2.4 million in the Full model to around 182 thousand in the Tiny model–the differences in average prediction times are relatively minor. The Full model has an average prediction time of 44.21 ms, while the Small and Tiny models achieve slightly faster times of 37.35 ms and 34.2 ms, respectively. These results indicate that although the total number of parameters is drastically lower in the smaller models, the reduction in computational time is not proportionally large.

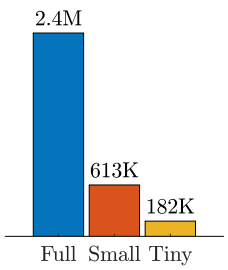


Fig. 10 Comparison between the number of parameters of the different PointNet models

Figure 13 compares the relative percentage errors of the parameters predicted by the Full, Small, and Tiny PointNet models. Both the median and mean values of the errors are provided to offer a comprehensive view of the models' performance.

The Full model demonstrates lower relative percentage errors compared to the Small and Tiny models. For most parameters, the Full model achieves the smallest median and mean errors, indicating more stable and accurate performance. Considering the median errors, the Small model performs similarly to the Full model, suggesting a trade-off between accuracy and its reduced number of parameters. Although it is less stable, as indicated by the higher mean relative errors.

On the other hand, the Tiny model, although showing comparable median errors to the other models for the majority of the predicted parameters, exhibits consistently higher mean errors. This indicates greater variability in its predictions and a higher probability of producing larger errors in certain instances.

Overall, while the reduction in model complexity from the Full to the Small model leads to only minor increases in prediction error, the Tiny model shows a noticeable decrease in performance stability. This indicates that further reducing the number of parameters beyond a certain point may compromise model accuracy, as evidenced by the increased

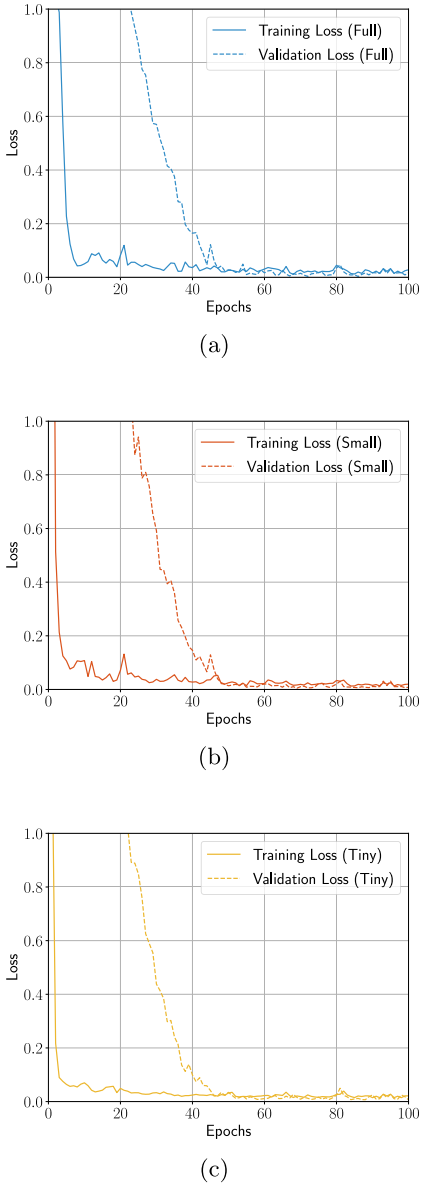


Fig. 11 Visualization of the MSE losses obtained during the training process of the different models. On the bottom right corner of each graph, the last value reached for the training loss is reported

variability in the Tiny model's performance compared to the other two architectures.

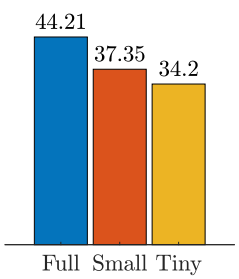


Fig. 12 Comparison between the average prediction time $[ms]$ required by the different models

The performance of the PointNet models is also evaluated by applying the 3D-IoU metric (Fig. 14). The results show that all the models maintain quite high accuracy in their predictions. The Full model achieves the highest values for both the median and the mean value. While there is a clear trend of decreasing accuracy as the model size decreases, even the Tiny model demonstrates robust results.

A detailed visual comparison of the models' output is provided in Fig. 15. The Full, Small, and Tiny models are compared against the ground truth by projecting the estimated ellipsoids onto the XY, XZ, and YZ planes. The Full model shows the closest alignment with the ground truth, exhibiting minimal spatial discrepancies. The Small and Tiny models display slightly larger deviations, particularly along the ellipsoid boundaries, though their overall shape approximations remain accurate. These visualizations complement the quantitative results, demonstrating the Full model's superior capacity to capture fine spatial details, while the smaller models, though less precise, still produce reasonably accurate representations of the target shapes.

The carried-out analysis underscores the balance between model complexity and predictive performance, with the Full model offering the highest accuracy at the cost of slightly higher computational cost, while the Small and Tiny models offer more computationally efficient solutions with a moderate trade-off in spatial accuracy.

5.5 Comparison between PointNet and the Optimization Problem

Based on the analysis carried out in the previous sections comparing the performance of various optimization algorithms and PointNet models, the best trade-off between computational time and accuracy for each category of solutions is offered by the PSO algorithm and the Full PointNet model, respectively.

A direct comparison between these two solutions highlights their respective strengths. The PSO algorithm consistently achieved the lowest relative percentage errors across all the parameters and attained the highest 3D-IoU values, indicating its superior accuracy and robustness in approximating the ellipsoid equation. Despite its precision, PSO's mean computational time is 1.48 seconds. While not fast enough for real-time applications, this is acceptable in contexts where precise, non-recurrent computations are required. On the other hand, the Full PointNet model, although faster since its average prediction time is only 44.21 ms, exhibits higher relative percentage errors and lower 3D-IoU values compared to PSO. This makes the Full model more suitable for applications where speed is a priority.

In determining the optimal approach for computing the ellipsoid equation representing the RS of a manipulator, an important consideration is that this computation does not typ-

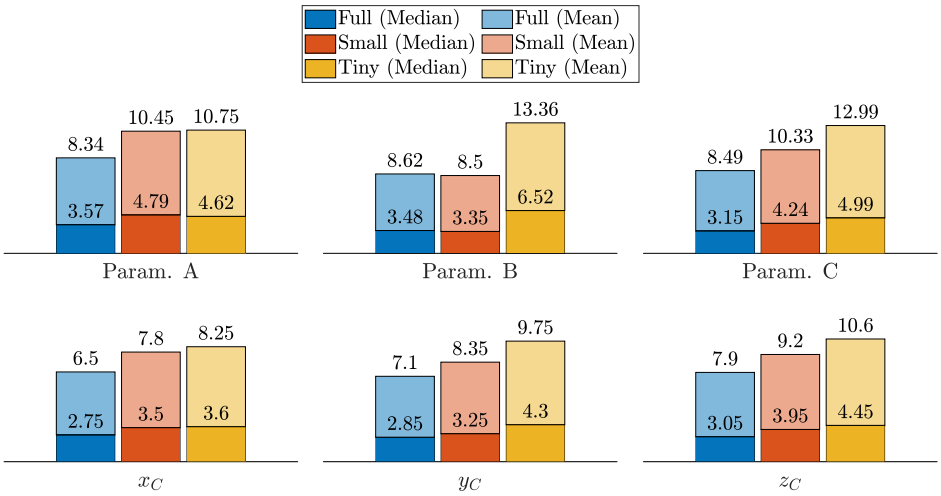


Fig. 13 Comparison of the relative percentage errors for the predictions obtained with the different PointNet models. Errors are reported in [%]

ically need to be performed in real-time, as long as the RS does not change over time. In this context, accuracy is more valuable than computation speed, as the ellipsoid representation will serve as a foundational parameter for subsequent manipulator planning and control tasks. Given these requirements, the PSO algorithm can be selected as the best choice. Its ability to provide more precise and robust results outweighs the slightly longer computation time, making it the ideal solution for estimating the ellipsoid equation that encapsulates the RS of a manipulator.

5.6 Evaluation on Real Robots

The comparative analysis carried out in Sections 5.3 and 5.4, complemented by the observations in Section 5.5, demonstrates that the optimization algorithm introduced in Section 4.2, powered by the PSO algorithm, represents the optimal trade-off between accuracy and computation time for the generation of an ellipsoid equation representing the RS of a manipulator. However, it should be noted that all the tests have been conducted using a benchmark set of point clouds that are not representative of the RS of real manipulators.

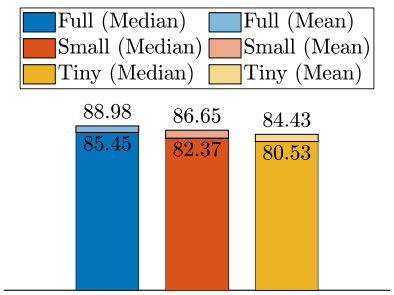


Fig. 14 Comparison between the different PointNet models using the 3D-IoU as metric. Values are reported in [%]

In this section, the capabilities of the PSO algorithm are evaluated in terms of its capacity to derive ellipsoid equations starting from point clouds that accurately represent the RS of real robots. More in detail, the complete approach is tested using one robot model for each category of manipulators. For the fixed-base manipulators, the UR5e from Universal Robotics [46] is used; it is composed of an anthropomorphic arm equipped with a spherical wrist. On the other hand, as an example for mobile manipulators, the LoCoBot WX250 from Trossen Robotics is used [58]; it has an anthropomorphic arm equipped with a wrist made of three revolute joints, mounted on top of a differential drive mobile base.

The point clouds representing the RS of each manipulator are obtained using the approach presented in Section 4.1. The number of samples for each joint of the arm is fixed at 20, resulting in the generation of sets containing 8,000 3D points (given that the manipulators' arms are not redundant). As previously outlined in Section 4.1, the additional DOFs resulting from the mobile base of the LoCoBot WX250 are not considered. The resulting point cloud is representative of the points that are reachable from the current pose of the mobile base.

To create a visual representation of the robot model, along with the generated point cloud, which represents its RS, and the predicted ellipsoid equation, we employed ROS Noetic [59] and the Rviz library [60]. Figure 16 illustrates the resulting output. From both Fig. 16a and b, it can be noticed that the predicted ellipsoid equation (in green) accurately represents the point cloud (in red). No points are left outside the shape, and the axis lengths are not excessive. Furthermore, Fig. 16b illustrates how the PSO algorithm handles a point cloud with a non-uniform distribution. Indeed, the approximation of the centre as the mean point of the cloud is significantly influenced by this particular distribution of points. Nevertheless, the optimization algorithm is able to adapt the initial approximation of the centre to include all

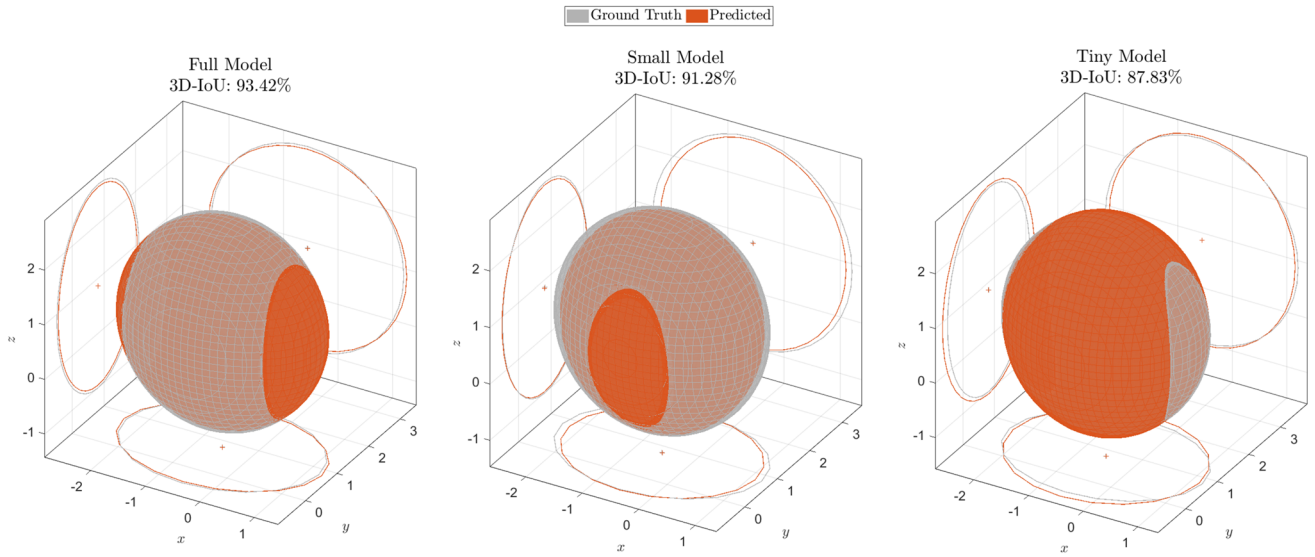


Fig. 15 Comparison between the ground truth ellipsoidal shape (in gray) and those predicted by the different PointNet models (in red)

the points within the shape, while simultaneously minimising the overall volume.

The visual analysis in Fig. 16b is further supported by the numerical comparison presented in Table 2. The RS equation obtained for the LoCoBot WX250 by applying the complete approach in Section 4.1 is compared against a model built starting from its technical specifications available at [62]. Such specifications have been used to approximate the RS to a sphere with radius $R = 0.570\ m$, which is the maximum reach specified for the wrist. This measure was chosen because it matches our choice of considering the last joint of the wrist as the representative point to build the point cloud.

Table 2 compares the results obtained by exploiting several point clouds generated using a different number of samples per joint. It points out the high level of accuracy that can be reached using the proposed method. The relative percentage errors obtained for the axis lengths are below 0.5%, while the 3DIoU metric is above 98.5% even when a few samples per joint are used. While increasing the sampling rate decreases the error and increases the 3DIoU, also the overall time needed to obtain the equation arises. Notably, the configurations with 20 to 30 samples per joint provide a good trade-off: they lead to a very low relative percentage error, while ensuring a high value for the 3DIoU metric and low computational times.

The ellipsoid equation enclosing the RS of the LoCoBot WX250 has been used to test the stability of the PSO algorithm. Moreover, the global convergence and the invariance of the initial solution have been considered, also taking into account the different point clouds. Three sets of points were generated using 5, 10 and 20 samples, and for each of them, a total of 1,000 optimization problems have been solved.

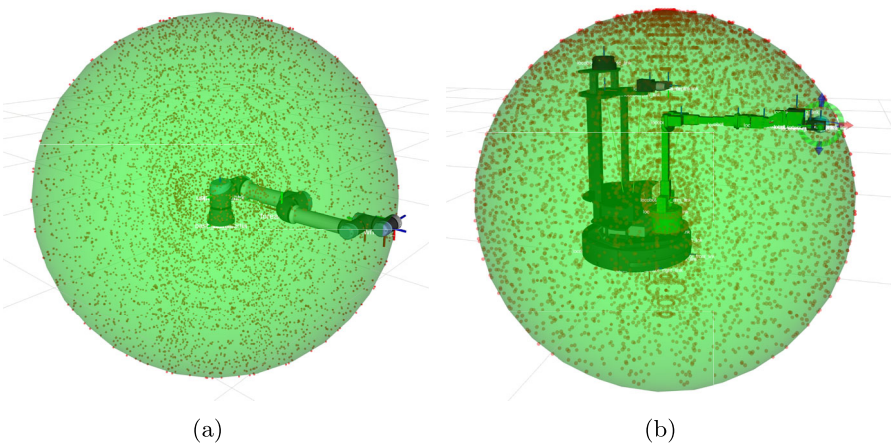


Fig. 16 Visualization of the RS of different types of manipulators together with the predicted ellipsoid equation. (a) UR5e [46]. (b) LoCoBot WX250 [58]

Table 2 Results obtained using the optimization-based method (powered by PSO algorithm) to build the RS of the LoCoBot WX250

Samples per joint	Relative perc. error [%]			3DIoU metric [%]	Point Cloud gen. time [ms]	Opt. prob. sol. time [s]
	A	B	C			
5	0.356	0.356	0.356	98.93	0.52	1.14
10	0.063	0.063	0.064	99.81	0.65	0.96
15	0.030	0.030	0.030	99.91	0.81	1.54
20	0.020	0.020	0.021	99.94	1.05	1.31
25	0.009	0.010	**0.013**	**99.97**	1.28	2.23
30	**0.006**	**0.007**	0.025	99.96	1.64	3.10
35	0.017	0.017	0.017	99.95	2.04	4.70
40	0.007	0.007	0.022	99.96	3.71	5.34
45	0.016	0.017	0.018	99.95	5.18	8.08
50	0.049	0.159	0.015	99.81	4.16	10.16

The lowest values for the errors and the highest value for the 3DIoU metric have been highlighted

These problems have been initialized with solutions whose elements were randomly generated: a uniform distribution between 0 and 1 has been used to generate the axis lengths, whereas a uniform distribution between -1 and 1 has been exploited to generate the center coordinates. The termination condition is based on the value of the objective function, as specified in Section 5.3. As demonstrated in Fig. 17, the PSO algorithm ensures the independence from the assigned initial solution and guarantees the convergence to the optimal solution with a tolerance for the objective value as low as 10^{-6}, avoiding sub-optimal results for the proposed optimization problem.

6 Conclusions and Future Works

An accurate representation and modeling of the set of points in space that can be reached by a manipulator is crucial for various robotic applications. This paper introduced a method to estimate the RS of a robotic manipulator, and to build a comprehensive and analytical representation of it based on an ellipsoid equation. Such a method is applicable to a variety of robots, including fixed-base and mobile manipulators. It involves generating point clouds using solely the robot's kinematic properties and estimating the ellipsoid parameters using two approaches: an optimization-based method and an ML model, specifically PointNet. The approach proposed for point cloud generation proved to be highly efficient, significantly reducing computational time when compared to traditional methods based on the solution of FK and IK problems. This efficiency was achieved by minimizing redundant calculations, allowing for faster generation of the reachable points.

The comparison of different approaches for the computation of the ellipsoid's parameters demonstrated that, while the PointNet-based approach offers the best solution in terms of speed, the designed optimization problems yield higher accuracy, especially when solved using the PSO algorithm. The performance and the impact of other optimization algorithms, like those based on evolutionary strategies and Bayesian optimization, will be analyzed in future studies. Different PointNet models were created and trained to ana-

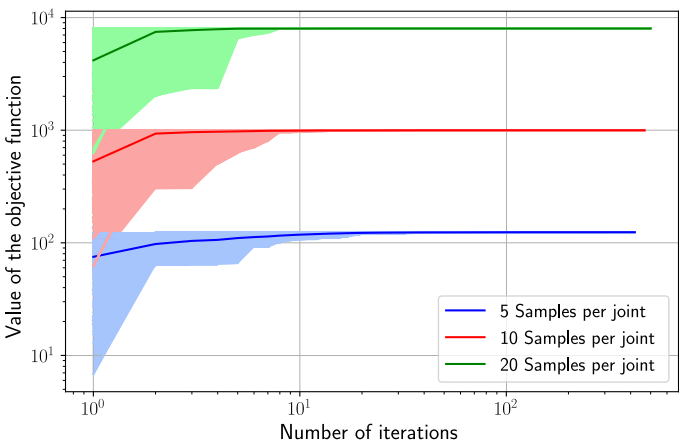


Fig. 17 Evolution of the objective values obtained in intermediate iterations of the PSO algorithm. For the sake of visualization and comparison, the absolute values are considered, and the logarithmic scale is used. Different optimal values are due to the different number of elements in the point cloud

lyze the impact of the number of learnable parameters on performance. By optimizing model complexity, we aim at enhancing the robots' ability to increase their autonomy by reducing computational load. The full-size PointNet model, despite having the highest number of parameters, hence requiring the highest prediction time on average, provided the most consistent results, though not at the level of precision achieved by the optimization algorithm powered by the PSO algorithm. This suggests that while PointNet is highly suitable for real-time applications due to its speed, the optimization-based approach should be preferred for tasks where accuracy is more important. Using PointNet to provide an initial estimate of the optimization-based approach can be a valuable research direction, requiring further investigation to assess its influence on computational efficiency and solution quality.

Future research can focus on applying the ellipsoid-based RS representation to practical robotic applications. One potential direction is to integrate this representation directly into robotic motion planning, enabling manipulators to adapt more effectively to dynamic environments. The compactness and efficiency of the ellipsoid model could also be leveraged in applications such as robot placement optimization and collision avoidance, where fast computation of reachable areas is critical. Finally, this method can be applied to mobile manipulators or robotic arms used in HRC, where safety and adaptability are crucial.

Author Contributions The study conception and design was mainly conducted by Rosario Francesco Cavelli. Material preparation, code development, data collection and analysis were performed by Rosario Francesco Cavelli. The first draft of the manuscript was written by Rosario Francesco Cavelli and all authors commented on previous versions of the manuscript. Pangcheng David Cen Cheng and Marina Indri acquired funding, supervised the work, co-authored the manuscript, served as advisors, and contributed to editing the final project. All authors read and approved the final manuscript.

Funding Open access funding provided by Politecnico di Torino within the CRUI-CARE Agreement. This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

Code/Data Availability Code and data used for this work have been made publicly available at https://github.com/Saro0800/Robotic-manipulators-reachability-space-modeling.git.

Declarations

Ethical Approval Not applicable.

Conflicts of Interest/Competing Interests The authors declare no conflicts of interest or competing interests.

Consent to Participate Not applicable.

Consent to Publish Not applicable.

References

1. Boehler, Q., Gervasoni, S., Charreyron, S.L., Chautems, C., Nelson, B.J.: On the workspace of electromagnetic navigation systems. IEEE Trans. Rob. **39**(1), 791–807 (2023). https://doi.org/10.1109/TRO.2022.3197107
2. Yang, J., Dymond, P., Jenkin, M.: Exploiting hierarchical probabilistic motion planning for robot reachable workspace estimation. In: Informatics in Control Automation and Robotics: Revised and Selected Papers from the International Conference on Informatics in Control Automation and Robotics 2009, pp. 229–241 (2011). Springer
3. Guan, Y., Yokoi, K., Zhang, X.: Numerical methods for reachable space generation of humanoid robots. Int. J. Robot. Res. **27**(8), 935–950 (2008)
4. Holmes, P., Kousik, S., Zhang, B., Raz, D., Barbalata, C., Roberson, M., Vasudevan, R.: Reachable sets for safe, real-time manipulator trajectory design (2020). ArXiv. https://doi.org/10.15607/RSS.2020.XVI.100
5. Lee, J., Ahn, J., Bakolas, E., Sentis, L.: Reachability-based trajectory optimization for robotic systems given sequences of rigid contacts. In: 2020 American Control Conference (ACC), pp. 2158–2165 (2020). IEEE
6. Sacchi, N., Sangiovanni, B., Incremona, G.P., Ferrara, A.: A configuration space reference generation approach for real-time collision avoidance of industrial robot manipulators. In: 2021 European Control Conference (ECC), pp. 1316–1321 (2021). https://doi.org/10.23919/ECC54610.2021.9655095
7. Singletary, A., Nilsson, P., Gurriet, T., Ames, A.D.: Online active safety for robotic manipulators. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 173–178 (2019). https://doi.org/10.1109/IROS40897.2019.8968231
8. Pereira, A., Althoff, M.: Safety control of robots under computed torque control using reachable sets. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 331–338 (2015). IEEE
9. Zacharias, F., Borst, C., Hirzinger, G.: Capturing robot workspace structure: representing robot capabilities. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3229–3236 (2007). IEEE
10. Porges, O., Stouraitis, T., Borst, C., Roa, M.A.: Reachability and capability analysis for manipulation tasks. In: ROBOT2013: First

Iberian Robotics Conference: Advances in Robotics, Vol. 2, pp. 703–718 (2014). Springer
11. Zacharias, F., Borst, C., Wolf, S., Hirzinger, G.: The capability map: A tool to analyze robot arm workspaces. Int. J. Humanoid Rob. **10**(04), 1350031 (2013)
12. Son, S.-W., Kwon, D.-S.: A convex programming approach to the base placement of a 6-DOF articulated robot with a spherical wrist. Int. J. Adv. Manuf. Technol. **102**(9–12), 3135–3150 (2019)
13. Long, P., Padir, T.: Evaluating robot manipulability in constrained environments by velocity polytope reduction. In: 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), pp. 1–9 (2018). https://doi.org/10.1109/HUMANOIDS.2018.8624962
14. Guan, Y., Yokoi, K.: Reachable space generation of a humanoid robot using the monte carlo method. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1984–1989 (2006). https://doi.org/10.1109/IROS.2006.282406
15. Anderson-Sprecher, P., Simmons, R.: Voxel-based motion bounding and workspace estimation for robotic manipulators. In: 2012 IEEE International Conference on Robotics and Automation, pp. 2141–2146 (2012). https://doi.org/10.1109/ICRA.2012.6225256
16. Dong, J., Trinkle, J.C.: Orientation-based reachability map for robot base placement. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1488–1493 (2015). https://doi.org/10.1109/IROS.2015.7353564
17. Quan, Y., Zhao, C., Lv, C., Wang, K., Zhou, Y.: The dexterity capability map for a seven-degree-of-freedom manipulator. Machines **10**(11) (2022). https://doi.org/10.3390/machines10111038
18. Vahrenkamp, N., Muth, D., Kaiser, P., Asfour, T.: Ik-map: An enhanced workspace representation to support inverse kinematics solvers. In: 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), pp. 785–790 (2015). https://doi.org/10.1109/HUMANOIDS.2015.7363443
19. Sandakalum, T., Ang, M.H.: Reachnet : Reachability maps in the presence of obstacles. In: 2023 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), pp. 108–113 (2023). https://doi.org/10.1109/CIS-RAM55796.2023.10370757
20. Zacharias, F., Borst, C., Beetz, M., Hirzinger, G.: Positioning mobile manipulators to perform constrained linear trajectories. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2578–2584 (2008). https://doi.org/10.1109/IROS.2008.4650617
21. Zacharias, F., Sepp, W., Borst, C., Hirzinger, G.: Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories. In: 2009 9th IEEE-RAS International Conference on Humanoid Robots, pp. 55–61 (2009). https://doi.org/10.1109/ICHR.2009.5379601
22. Forstenhäusler, M., Wetner, T., Dietmayer, K.: Optimized mobile robot positioning for better utilization of the workspace of an attached manipulator. In: 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 2074–2079 (2020). https://doi.org/10.1109/AIM43001.2020.9158922
23. Chaves-Arbaiza, I., García-Vaglio, D., Ruiz-Ugalde, F.: Smart placement of a two-arm assembly for an everyday object manipulation humanoid robot based on capability maps. In: 2018 IEEE International Work Conference on Bioinspired Intelligence (IWOBI), pp. 1–9 (2018). https://doi.org/10.1109/IWOBI.2018.8464192
24. Burget, F., Bennewitz, M.: Stance selection for humanoid grasping tasks by inverse reachability maps. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 5669–5674 (2015). https://doi.org/10.1109/ICRA.2015.7139993
25. Makhal, A., Goins, A.K.: Reuleaux: Robot base placement by reachability analysis. In: 2018 Second IEEE International Conference on Robotic Computing (IRC), pp. 137–142 (2018). https://doi.org/10.1109/IRC.2018.00028
26. Birr, T., Pohl, C., Asfour, T.: Oriented surface reachability maps for robot placement. In: 2022 International Conference on Robotics and Automation (ICRA), pp. 3357–3363 (2022). https://doi.org/10.1109/ICRA46639.2022.9811600
27. Jauhri, S., Peters, J., Chalvatzaki, G.: Robot learning of mobile manipulation with reachability behavior priors. IEEE Robotics and Automation Letters **7**(3), 8399–8406 (2022). https://doi.org/10.1109/LRA.2022.3188109
28. Zhang, H., Sheng, Q., Hu, J., Sheng, X., Xiong, Z., Zhu, X.: Cooperative transportation with mobile manipulator: A capability map-based framework for physical human–robot collaboration. IEEE/ASME Trans. Mechatron. **27**(6), 4396–4405 (2022). https://doi.org/10.1109/TMECH.2022.3155601
29. Yoshikawa, T.: Manipulability of robotic mechanisms. Int. J. Robot. Res. **4**(2), 3–9 (1985)
30. Son, S.-W., Kwon, D.-S.: A convex programming approach to the base placement of a 6-dof articulated robot with a spherical wrist. Int. J. Adv. Manuf. Technol. **102**, 3135–3150 (2019)
31. Haviland, J., Corke, P.: A purely-reactive manipulability-maximising motion controller (2020). arXiv preprint arXiv:2002.11901
32. Chiacchio, P., Bouffard-Vercelli, Y., Pierrot, F.: Evaluation of force capabilities for redundant manipulators. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 4, pp. 3520–35254 (1996). https://doi.org/10.1109/ROBOT.1996.509249
33. Althoff, M., Krogh, B.H.: Reachability analysis of nonlinear differential-algebraic systems. IEEE Trans. Autom. Control **59**(2), 371–383 (2014). https://doi.org/10.1109/TAC.2013.2285751
34. Skuric, A., Padois, V., Daney, D.: Approximating robot reachable space using convex polytopes. In: International Workshop on Human-Friendly Robotics, pp. 45–60 (2022). Springer
35. Schepp, S.R., Thumm, J., Liu, S.B., Althoff, M.: Sara: A tool for safe human-robot coexistence and collaboration through reachability analysis. In: 2022 International Conference on Robotics and Automation (ICRA), pp. 4312–4317 (2022). https://doi.org/10.1109/ICRA46639.2022.9811952
36. Skuric, A., Padois, V., Rezzoug, N., Daney, D.: On-line feasible wrench polytope evaluation based on human musculoskeletal models: An iterative convex hull method. IEEE Robot. Autom. Lett. **7**(2), 5206–5213 (2022). https://doi.org/10.1109/LRA.2022.3155374
37. Zacharias, F., Borst, C., Hirzinger, G.: Online generation of reachable grasps for dexterous manipulation using a representation of the reachable workspace. In: 2009 International Conference on Advanced Robotics, pp. 1–8 (2009)
38. Vahrenkamp, N., Asfour, T., Dillmann, R.: Efficient inverse kinematics computation based on reachability analysis. Int. J. Humanoid Rob. **9**(04), 1250035 (2012)
39. Guamán Rivera, R., García Alvarado, R., Martínez-Rocamora, A., Auat Cheein, F.: Workspace analysis of a mobile manipulator with obstacle avoidance in 3D printing tasks. Appl. Sci. **11**(17), 7923 (2021)
40. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
41. Tychola, K.A., Vrochidou, E., Papakostas, G.A.: Deep learning based computer vision under the prism of 3D point clouds: a systematic review. The Visual Computer, 1–43 (2024)
42. Patra, J.C., Kot, A.C.: Nonlinear dynamic system identification using chebyshev functional link artificial neural networks. IEEE Trans. Syst. Man Cybern. Part B (Cybernetics) **32**(4), 505–511 (2002)

43. Rendle, S., Krichene, W., Zhang, L., Anderson, J.: Neural collaborative filtering vs. matrix factorization revisited. In: Proceedings of the 14th ACM Conference on Recommender Systems, pp. 240–248 (2020)
44. Intel Core i7-1165G7 processor technical specifications. Accessed Oct 2024. https://www.intel.com/content/www/us/en/products/sku/208921/intel-core-i71165g7-processor-12m-cache-up-to-4-70-ghz-with-ipu/specifications.html
45. Repository with the code used for the experimental evaluation. [Online; Accessed Dec 2024]. https://github.com/Saro0800/Robotic-manipulators-reachability-space-modeling.git
46. Robots, U.: UR5e from Universal Robots. Accessed May 2024. https://www.universal-robots.com/products/ur5-robot/
47. Zhou, D., Fang, J., Song, X., Guan, C., Yin, J., Dai, Y., Yang, R.: Iou loss for 2D/3D object detection. In: 2019 International Conference on 3D Vision (3DV), pp. 85–94 (2019). IEEE
48. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) challenge. Int. J. Comput. Vision **88**, 303–338 (2010)
49. Ming, Q., Miao, L., Ma, Z., Zhao, L., Zhou, Z., Huang, X., Chen, Y., Guo, Y.: Deep dive into gradients: Better optimization for 3D object detection with gradient-corrected IoU supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5136–5145 (2023)
50. Zheng, Y., Zhang, D., Xie, S., Lu, J., Zhou, J.: Rotation-robust intersection over union for 3D object detection. In: European Conference on Computer Vision, pp. 464–480 (2020). Springer
51. Robert, C.P., Casella, G., Robert, C.P., Casella, G.: Monte Carlo integration. Monte Carlo statistical methods, 71–138 (1999)
52. Blank, J., Deb, K.: pymoo: Multi-objective optimization in python. IEEE Access **8**, 89497–89509 (2020)
53. Hooke, R., Jeeves, T.A.: “Direct Search" Solution of Numerical and Statistical Problems. J. ACM (JACM) **8**(2), 212–229 (1961)
54. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press (1992)
55. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN’95-international Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995). ieee
56. Wang, H., Gong, J.: A comparative study of GA, PSO and SCE algorithms for estimating kinetics of biomass pyrolysis. Emerg. Manag. Sci. Technol. **3**(1) (2023)
57. Bayona, E., Sierra-García, J.E., Santos, M.: Comparative analysis of metaheuristic optimization methods for trajectory generation of automated guided vehicles. Electronics **13**(4), 728 (2024)
58. Robotics, T.: LoCoBot WX250 with 6 DOF Arm (with Lidar) technical specifications. Accessed Feb 2024. https://docs.trossenrobotics.com/interbotix_xslocobots_docs/specifications/locobot_wx250s.html
59. Stanford Artificial Intelligence Laboratory et al.: Robot Operating System (ROS). Accessed Oct 2024. https://www.ros.org
60. Kam, H.R., Lee, S.-H., Park, T., Kim, C.-H.: Rviz: a toolkit for real domain data visualization. Telecommun. Syst. **60**, 337–345 (2015)

Rosario Francesco Cavelli received the M.Sc. degree cum laude in Computer Engineering from Politecnico di Torino, Turin, Italy, in 2023. In 2024, he was a Research Scholar with the Robotics Research Group, coordinated by Prof. Marina Indri, at the Department of Electronics and Telecommunications, Politecnico di Torino. He is currently pursuing a Ph.D. in Robotics Engineering at Politecnico di Torino, with research interests in human–robot interaction, task-level planning, and smart manipulation for collaborative applications, leveraging both classical methods and AI-based techniques. He is an active member of the IEEE–HKN Honor Society and IEEE Young Professional.

Pangcheng David Cen Cheng received a double degree in M.Sc. in Mechatronic Engineering in 2017 from Politecnico di Torino (Italy) along with a B.Sc. degree in Electrical Engineering, with specialization in Electronics from Universidad Central de Venezuela (Caracas, Venezuela). He obtained his Ph.D. in Electrical, Electronics, and Communications Engineering at Politecnico di Torino in 2023. In 2018, he worked as a Research Assistant at Politecnico di Torino for the development of robotic applications in Smart Factories within the HuManS project coordinated by COMAU S.p.A. Since 2023, he has been an Assistant Professor at Politecnico di Torino in collaboration with the FAIR project. Currently, he is working with control systems, path planning algorithms, and sensor data fusion for mobile robots and manipulators for collaborative applications with humans, employing classical and AI-based approaches.

Marina Indri received the M.Sc. degree in electronic engineering and the Ph.D. degree in control and computer engineering from Politecnico di Torino, Turin, Italy, in 1991 and 1995, respectively. She is currently an Associate Professor with the Department of Electronics and Telecommunications of Politecnico di Torino, teaching robotics and automatic control. She coauthored about 100 papers in international journals, books, and conference proceedings, in the fi elds of robotics and automatic control. She participated in various research projects, and was involved also in research joint activities with industrial partners. Her current research interests include industrial and mobile robotics areas. Prof. Indri was the recipient of the Best Paper Award in Factory Automation at IEEE ETFA 2013 and IEEE ETFA 2024 and the 2nd prize of the euRobotics Technology Transfer Award in 2014, and was among the fi nalists of the same Award in 2017 for joint works with COMAU S.p.A. She is Senior Editor of the IEEE/ASME Transactions on Mechatronics.