DOI: 10.2507/36th.daaam.proceedings.xxx

EFFICIENT COMPUTATION OF 2D SELF-MOTION MANIFOLDS: A METHOD COMPARISON IN A 5-DOF ROBOT

Marc Fabregat-Jaén, Adrián Peidró, Luis Payá & Óscar Reinoso

This Publication has to be referred as: Fabregat-Jaén, M[arc]; Peidró, A[drián]; Payá, L[uis] & Reinoso, Ó[scar] (2025). Efficient Computation of 2D Self-Motion Manifolds: A Method Comparison in a 5-DOF Robot, Proceedings of the 36th DAAAM International Symposium, pp.xxxx-xxxx, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-xx-x, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/36th.daaam.proceedings.xxx

Abstract

This paper presents a comparative evaluation of methods for computing self-motion manifolds in redundant robotic manipulators. Redundant robots have more degrees of freedom than required to perform their task. We consider a 5-degree-of-freedom robot tasked with reaching three-dimensional positions, which results in two degrees of redundancy. This implies that the inverse kinematic problem of the robot has infinitely many solutions that lie on two-dimensional self-motion manifolds. Computing these manifolds is of paramount importance for planning feasible and globally optimal motions. In this paper, we compare different methods for computing these manifolds: cellular automata, continuation and sampling-based methods. Our results show that the established techniques like continuation and sampling provide better performance. We further discuss the individual strengths and limitations of each approach. By offering this balanced evaluation, our study aims to guide researchers and users in selecting appropriate tools for analysing redundancy in industrial robotic systems.

Keywords: Redundant robots; Self-motion manifolds; Redundancy resolution; Robotics;

1. Introduction

Kinematic redundancy arises when a robotic manipulator possesses more degrees of freedom (DOF) than are strictly required to accomplish a given task. This property, common in many industrial and collaborative robots, enables the system to exploit its extra DOF to achieve secondary objectives such as obstacle avoidance, singularity evasion, or energy minimization [1]. However, this flexibility comes at the cost of increased computational complexity: the inverse kinematics problem (IKP) for redundant manipulators is underdetermined, admitting infinitely many solutions for a given end-effector position.

Redundancy resolution methods are typically classified into velocity-level and position-level approaches. Velocity-level methods, such as those based on the pseudoinversion of the Jacobian [2] or optimization-based schemes [3], provide local solutions by integrating joint velocities to follow a desired task trajectory. While effective for real-time control,

these methods are inherently local and may fail to capture the global structure of the solution space, often resulting in suboptimal or non-cyclic motions [4].

In contrast, position-level methods aim to characterize the entire set of solutions for a given task. Self-motion manifolds [5] and Feasibility Maps [6] are examples of approaches that describe the infinite set of solutions. Among these, self-motion manifolds (SMMs) are of particular interest: they are surfaces in the joint space along which the end-effector position remains constant. The computation and analysis of SMMs are crucial for applications requiring global optimality, cyclic motions, or exhaustive feasibility analysis: capabilities that are especially valuable in industrial contexts such as welding, painting, or assembly, where repeatability and reliability are paramount.

Several methods have been proposed for computing self-motion manifolds. Continuation methods systematically trace the solution set from an initial configuration. The original continuation approach, employed by [7], incrementally constructs the manifold by following the Jacobian's null space, which is tangent to the manifold at each point, to predict subsequent points. These points are then corrected using the Newton-Raphson method, ensuring convergence to the manifold. This technique allows for the computation of one-dimensional self-motion curves for robots with a single degree of redundancy. Later, [8] generalized the method to higher-dimensional manifolds by constructing an atlas of local parameterizations (charts) that collectively cover the manifold.

Sampling-based methods typically identify the self-motion manifold in two stages: first, they sample a set of valid joint configurations to generate a point cloud that approximates the manifolds, and then cluster these points to identify disjoint manifolds. Ref. [9] proposed sampling the point cloud by randomly selecting joint configurations and checking whether they map (via forward kinematics) sufficiently close to the desired task value. [10] presented a more systematic approach by sweeping every possible combination of redundant joint variables over their feasible range and solving the remaining joint variables to satisfy the task constraint. This method avoids computing points that would later be discarded, as the resulting points are guaranteed to belong to the manifold.

Other strategies for SMM computation have also been explored. For instance, [11] proposed a cellular automata-based method that discretizes the joint space and iteratively updates the state of each cell. [12] introduced a swarm intelligence approach, specifically the artificial bee colony algorithm, to sequentially trace the SMMs, similarly to continuation methods, without requiring the Jacobian matrix. Multi-objective optimization has also been proposed [13] to compute the manifolds, and a branch-and-bound method [14] systematically shrinks the search space to exclude infeasible regions.

Despite the variety of available techniques, comprehensive comparisons of their performance and suitability for practical industrial problems remain limited in the literature. This paper addresses this gap by evaluating and comparing several representative methods for computing two-dimensional self-motion manifolds in a 5-DOF robot. By analysing their strengths and limitations, we aim to provide guidance for researchers and practitioners in selecting appropriate tools for redundancy analysis and exploitation in industrial robotic systems.

The remainder of the paper is organized as follows: Section 2 presents the inverse kinematics of the studied 5-DOF robot; Section 3 describes the method presented in [10] for computing self-motion manifolds by sweeping two redundant joint angles; Section 4 presents the higher-dimensional continuation method [8]; Section 5 presents a discussion of the results obtained from the different methods; and Section 6 concludes the paper and outlines future work.

2. Inverse Kinematics of the Studied 5-DOF Robot

The robot studied in this paper is a 5-DOF serial robot defined by the Denavit-Hartenberg (DH) parameters defined in Table 1. The robot and DH frames are shown in Fig. 1. This robot is taken from an example presented in [11].

Transformation between frames $i-1 \rightarrow i$	θ_i (rad)	d_i	a_i (m)	α_i (rad)
$0 \rightarrow 1$	q_1	0	0	$\pi/2$
$1 \rightarrow 2$	q_2	0	0.25	0
$2 \rightarrow 3$	q_3	0	0.25	$\pi/2$
$3 \rightarrow 4$	q_4	0	0.25	0
$4 \rightarrow 5$	q_5	0	0.25	0

Table 1. Denavit-Hartenberg (DH) parameters of the studied 5-DOF robot.

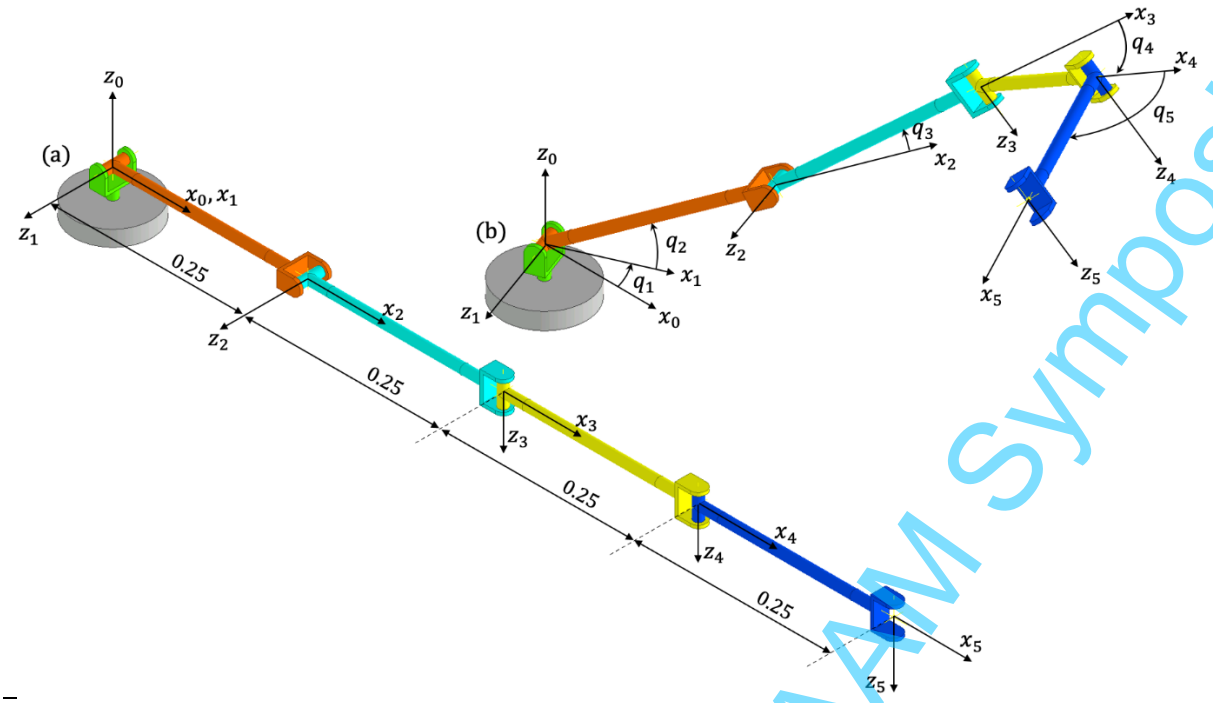

Fig. 1. (a) Robot in the reference pose. (b) Robot in a more arbitrary pose defined by: $q_1 = 20°$, $q_2 = 30°$, $q_3 = 10°$, $q_4 = 30°$, and $q_5 = 45°$.

By multiplying the DH matrices from the fixed base of the robot to its end-effector, we obtain the homogeneous transformation matrix that represents the position and orientation of the end-effector with respect to the fixed base frame:

$$\mathbf{T} = {}^{0}\mathbf{T}_1 \, {}^{1}\mathbf{T}_2 \, {}^{2}\mathbf{T}_3 \, {}^{3}\mathbf{T}_4 \, {}^{4}\mathbf{T}_5, \tag{1}$$

where the DH matrix between consecutive frames $i-1$ and i is:

$${}^{i-1}\mathbf{T}_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

In this paper, we are interested only in the position coordinates of the end-effector, neglecting its orientation. The position of the end-effector is taken from the last column of $\mathbf{T}$ as computed in (1), which has the following symbolic expression:

$$\begin{aligned} x &= \tfrac{1}{4}[\cos q_1 \cdot (a \cdot \cos(q_2 + q_3) + \cos q_2) + b \cdot \sin q_1] \\ y &= \tfrac{1}{4}[\sin q_1 \cdot (a \cdot \cos(q_2 + q_3) + \cos q_2) - b \cdot \cos q_1] \\ z &= \tfrac{1}{4}[a \cdot \sin(q_2 + q_3) + \sin q_2], \end{aligned} \tag{4}$$

where $a = 1 + \cos q_4 + \cos(q_4 + q_5)$ and $b = \sin q_4 + \sin(q_4 + q_5)$.

The inverse kinematic problem consists in determining the values of the joint coordinates $\mathbf{q} = [q_1, \ldots, q_5]^T$ to achieve a desired position $\mathbf{p} = [x, y, z]$. Since we are using five angles to control the position of only three coordinates, the robot is kinematically redundant, that is, there are two redundant degrees of freedom to control the position of the end-effector. The inverse kinematic problem in that case is underdetermined: there are three equations from which to solve five unknowns. For a given $\mathbf{p}$, the solution set of these equations in the space $(q_1, \ldots, q_5)$ will generically be a surface, or a set of disjoint surfaces. These surfaces are the self-motion manifolds (SMMs): moving the angles $[q_1, \ldots, q_5]$ along these surfaces does not modify the position $\mathbf{p}$ of the end-effector.

Since we cannot solve all joints $\mathbf{q} = [q_1, \ldots, q_5]^T$ from (3) for a given $\mathbf{p} = [x, y, z]$, we will solve $[q_1, q_2, q_3]$ in terms of $[q_4, q_5]$, which will remain as free parameters in the following. Note that, for given values of $[q_4, q_5]$, the values of $[a, b]$ in (3) are known.

To solve this system of equations assuming that $[q_4, q_5]$ are known, first we note that the first two equations are linear in $\cos q_1$ and $\sin q_1$:

$$\begin{bmatrix} a\cdot\cos(q_2+q_3)+\cos q_2 & b \\ -b & a\cdot\cos(q_2+q_3)+\cos q_2 \end{bmatrix}\begin{bmatrix}\cos q_1\\ \sin q_1\end{bmatrix}=\begin{bmatrix}x\\ y\end{bmatrix}4, \tag{4}$$

from which we solve the sine and cosine of q_1:

$$\begin{bmatrix}\cos q_1\\ \sin q_1\end{bmatrix}=\begin{bmatrix} a\cdot\cos(q_2+q_3)+\cos q_2 & b \\ -b & a\cdot\cos(q_2+q_3)+\cos q_2 \end{bmatrix}^{-1}\begin{bmatrix}x\\ y\end{bmatrix}4 \tag{5}$$

$$\begin{bmatrix}\cos q_1\\ \sin q_1\end{bmatrix}=\frac{1}{D}\underbrace{\begin{bmatrix} a\cdot\cos(q_2+q_3)+\cos q_2 & -b \\ b & a\cdot\cos(q_2+q_3)+\cos q_2 \end{bmatrix}}_{\mathbf{P}}\begin{bmatrix}x\\ y\end{bmatrix}4, \tag{6}$$

with $D=\det(\mathbf{P})=(a\cdot\cos(q_2+q_3)+\cos q_2)^2+b^2$. We can eliminate the unknown q_1 by imposing that $\cos^2 q_1+\sin^2 q_1=1$, which yields:

$$\cos^2 q_1+\sin^2 q_1=\begin{bmatrix}\cos q_1\\ \sin q_1\end{bmatrix}^T\begin{bmatrix}\cos q_1\\ \sin q_1\end{bmatrix}=\frac{16}{D^2}\begin{bmatrix}x\\ y\end{bmatrix}^T\underbrace{\mathbf{P}^T\mathbf{P}}_{\begin{bmatrix}D & 0\\ 0 & D\end{bmatrix}}\begin{bmatrix}x\\ y\end{bmatrix}=\frac{16}{D}(x^2+y^2)=1 \tag{7}$$

That is:

$$16(x^2+y^2)=D=(a\cdot\cos(q_2+q_3)+\cos q_2)^2+b^2 \tag{8}$$

$$16(x^2+y^2)-b^2=(a\cdot\cos(q_2+q_3)+\cos q_2)^2 \tag{9}$$

$$\pm4\sqrt{(x^2+y^2)-b^2}=a\cdot\cos(q_2+q_3)+\cos q_2 \tag{10}$$

Now, (5) can be written as:

$$4z=a\cdot\sin(q_2+q_3)+\sin q_2 \tag{11}$$

Equations (10) and (11) constitute a system of two equations and two unknowns. More importantly: the form of these equations is identical to those of a 2R planar serial arm, as demonstrated in Fig. 2. Thus, q_2 and q_3 can be solved by solving the inverse kinematic problem of such equivalent 2R arm, which is a straightforward solution that can be found in any textbook on robot kinematics. The inverse kinematics of the 2R arm is summarized in Fig. 2, which yields two possible solutions for a given position of its end-effector $[x', y']$. Since there are two possible values of x' due to the "±" as noted in Fig. 2, the system formed by (10) and (11) has up to four possible solutions for (q_2, q_3). For each solution, a unique value of $\sin q_1$ and $\cos q_1$ is obtained from (6), which yields a unique corresponding value of $q_1 = \text{atan2}(\sin q_1, \cos q_1)$, completing the resolution of the inverse kinematic problem of the 5-DOF robot of Fig. 1 for a given pair $[q_4, q_5]$ and a desired position $\mathbf{p} = [x, y, z]$.

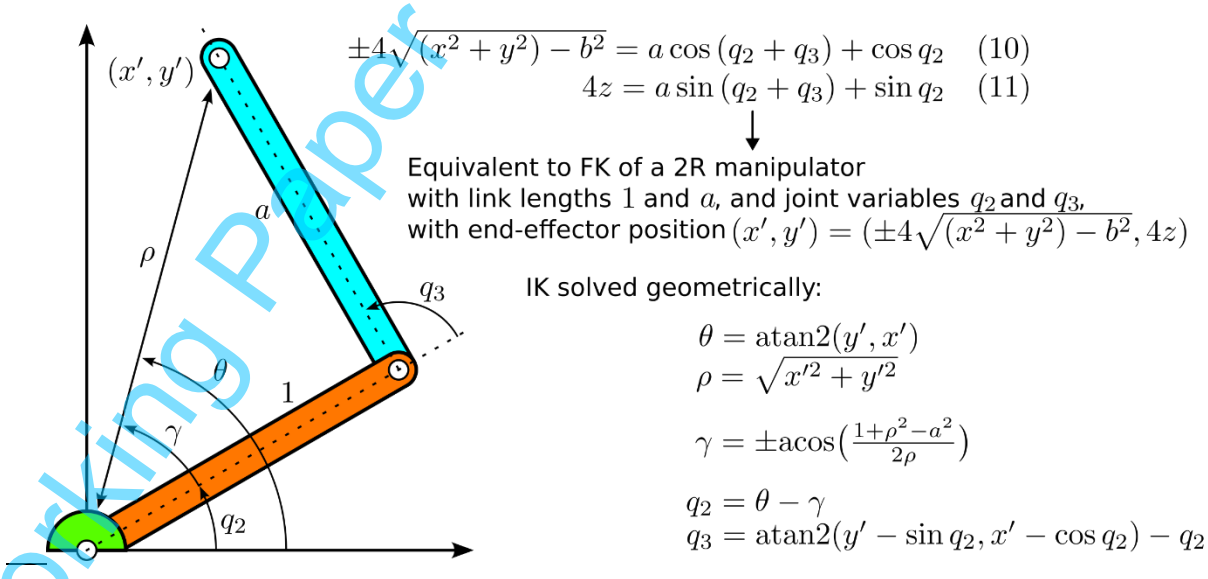

Fig. 2. Resulting system of equations, equivalent to that of a 2R planar arm.

3. Computing self-motion manifolds by sweeping two redundant joint angles

The sweeping method for computing self-motion manifolds, introduced by [10], systematically explores the feasible ranges of r redundant joint variables, where $r = n - m$ is the degree of redundancy, n is the number of DOFs, and m is the dimension of the task. For each discretized values of these variables, the remaining $n - r$ joint variables are solved to satisfy the task and constraints. After sweeping through the entire range, the resulting point cloud is clustered in order to identify disjoint self-motion manifolds.

The algorithm to compute SMMs via sweeping redundant joints proceeds as follows, and is illustrated in Fig. 3, which corresponds to the SMMs of the 2R arm of Fig. 2 for task value $y' = 1$ m, and an arbitrary task constraint.

1. Discretize each of the r redundant joints into N points within their feasible ranges, forming a grid of N^r combinations.
2. For each grid point, solve the inverse kinematics for the remaining $n - r$ joints to satisfy the task constraint.
3. Collect all valid solutions to construct a point cloud representing the self-motion manifolds.
4. Cluster the point cloud to identify disjoint self-motion manifolds.

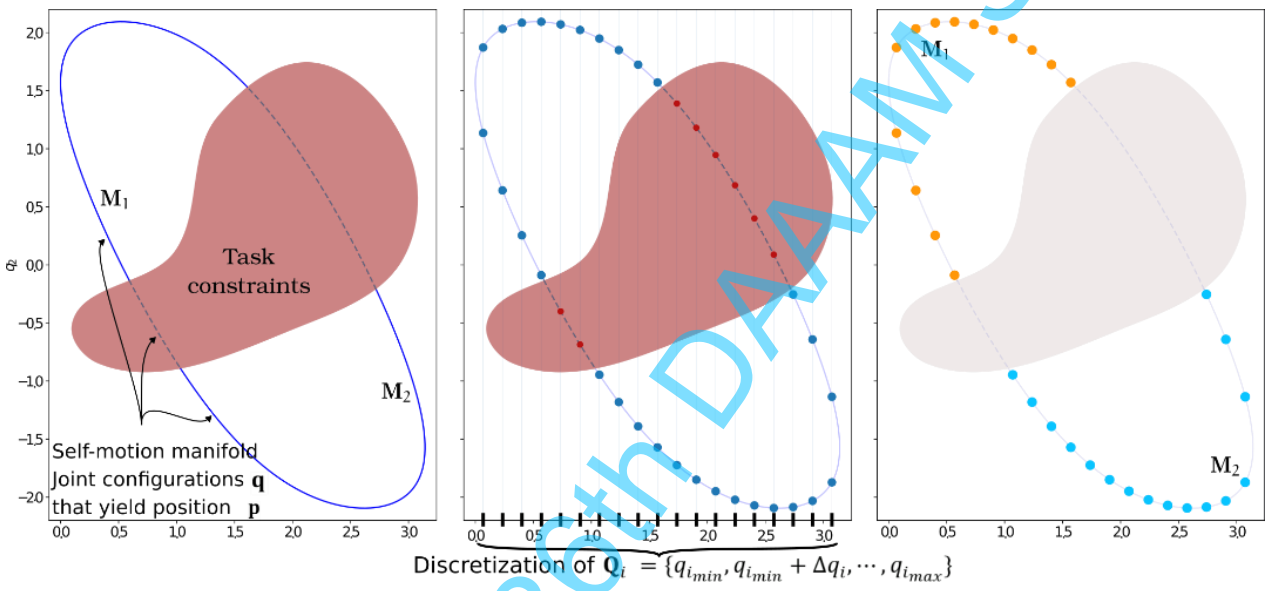

Fig. 3. Illustration of the sweeping method for computing self-motion manifolds of the 2R arm with $y' = 1$ m.

Let us consider the 5-DOF robot in Fig. 1 to further exemplify the algorithm. It has $n = 5$ joints and $m = 3$ task dimensions (3D end-effector position), yielding $r = n - m = 2$ redundant joints. We select the last two joints, (q_4, q_5), to sweep over their feasible ranges. Each joint i is discretized into a set $\mathbf{Q}_i = \{q_{i_{min}}, q_{i_{min}} + \Delta q_i, \cdots, q_{i_{max}}\}$, where $\Delta q_i = (q_{i_{max}} - q_{i_{min}}) / (N - 1)$, and N is the number of discretization points per joint. The Cartesian product $\mathbf{Q}_r = \mathbf{Q}_4 \times \mathbf{Q}_5$ yields a grid of N^2 pairs (q_4, q_5).

For each pair (q_4, q_5), the inverse kinematics are solved for all possible solutions of the remaining joints (q_1, q_2, q_3), as detailed in Section 2. Solutions are checked for validity (e.g., joint limits, collision avoidance) and retained if feasible. The resulting point cloud accurately represents the self-motion manifolds at the specified end-effector position.

Since the point cloud may contain points from multiple disjoint self-motion manifolds, a clustering algorithm is applied to distinguish them. Specifically, we use the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm [15], which is well-suited for identifying clusters of arbitrary shape without requiring the number of clusters *a priori*. DBSCAN groups closely packed points and returns the identified clusters as disjoint self-motion manifolds, as Fig. 3(c) illustrates.

4. Computing self-motion manifolds by continuation

3.1. One-dimensional self-motion manifolds (curves)

The basis for the continuation method for computing self-motion manifolds was first introduced by [7]. Starting from an initial robot configuration, this approach generates a one-dimensional self-motion curve by iteratively advancing along the null space of the Jacobian, which is tangent to the manifold at each point. Fig. 4 illustrates this process.

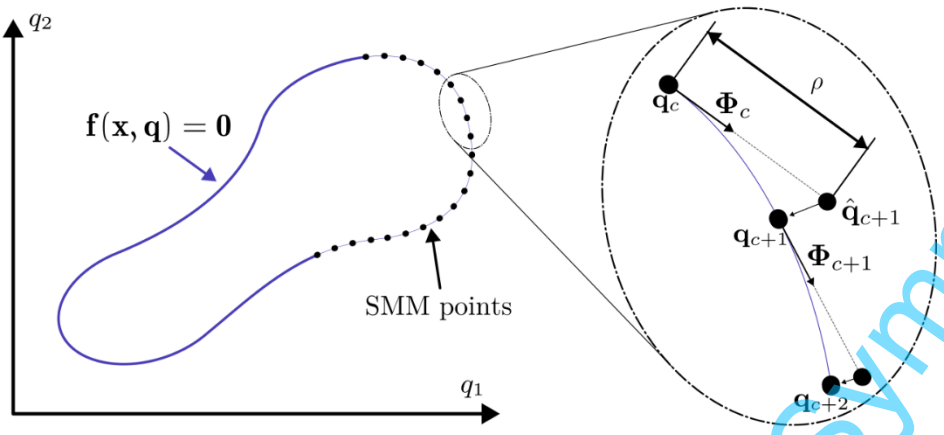

Fig. 4. Illustration of the continuation method for computing 1D self-motion manifolds.

The logic of the algorithm can be summarized as follows:

1. Compute a seed configuration $\mathbf{q}_0$ that lies on the SMM, if not already provided, and initialize $\mathbf{q}_c \leftarrow \mathbf{q}_0$.
2. Determine an orthonormal basis $\boldsymbol{\Phi}_c$ for the null space of the Jacobian matrix $\mathbf{J}$ at $\mathbf{q}_c$.
3. Take a step of size ρ along $\boldsymbol{\Phi}_c$, which is tangent to the SMM at $\mathbf{q}_c$, to obtain the candidate point $\hat{\mathbf{q}}_{c+1}$.
4. Project $\hat{\mathbf{q}}_{c+1}$ back onto the SMM by means of Newton's method, yielding the corrected point $\mathbf{q}_{c+1}$.
5. Repeat steps 2-4, updating the current configuration $\mathbf{q}_c \leftarrow \mathbf{q}_{c+1}$, until the entire manifold is traced.

The Jacobian matrix $\mathbf{J}(\mathbf{q})$ is the matrix of partial derivatives of the task function $\mathbf{f}(\mathbf{x},\mathbf{q})$ with respect to the joint variables $\mathbf{q}$:

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial q_1} & \cdots & \frac{\partial f_m}{\partial q_n} \end{bmatrix} \tag{12}$$

To obtain an orthonormal basis $\boldsymbol{\Phi}_c$ for the null space, the Singular Value Decomposition (SVD) of the Jacobian is used:

$$\mathbf{J}(\mathbf{q}) = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{13}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\mathbf{S}$ is a diagonal matrix of singular values. The null space is spanned by the $n-m$ last columns of $\mathbf{V}$.

After taking a step, the candidate configuration $\hat{\mathbf{q}}_{c+1}$ must be corrected back onto the SMM defined by $\mathbf{f}(\mathbf{x},\mathbf{q})=\mathbf{0}$. This is accomplished by using the Newton-Raphson method, which iteratively updates $\hat{\mathbf{q}}_{c+1}$ as follows:

$$\hat{\mathbf{q}}_{c+1} \leftarrow \hat{\mathbf{q}}_{c+1} - \mathbf{J}(\hat{\mathbf{q}}_{c+1})^{\dagger}\,\mathbf{f}(\mathbf{x},\hat{\mathbf{q}}_{c+1}), \tag{14}$$

where $\mathbf{J}(\hat{\mathbf{q}}_{c+1})^{\dagger}$ denotes the Moore-Penrose pseudoinverse of the Jacobian. This process is repeated until convergence, i.e., until $\left\|\mathbf{J}(\hat{\mathbf{q}}_{c+1})^{\dagger}\,\mathbf{f}(\mathbf{x},\hat{\mathbf{q}}_{c+1})\right\|$ is below a specified threshold ε. At convergence, $\hat{\mathbf{q}}_{c+1}$ lies on the manifold, and $\mathbf{q}_{c+1}$ is set to this value.

3.2. Higher-dimensional self-motion manifolds

The continuation method can be generalized to compute higher-dimensional self-motion manifolds, as described by [8]. We have based our approach in the interpretation by [16]. The central idea is to build an atlas of local parameterizations (charts) that collectively cover the manifold. The main concepts and steps are outlined below, and illustrated in Fig. 5.

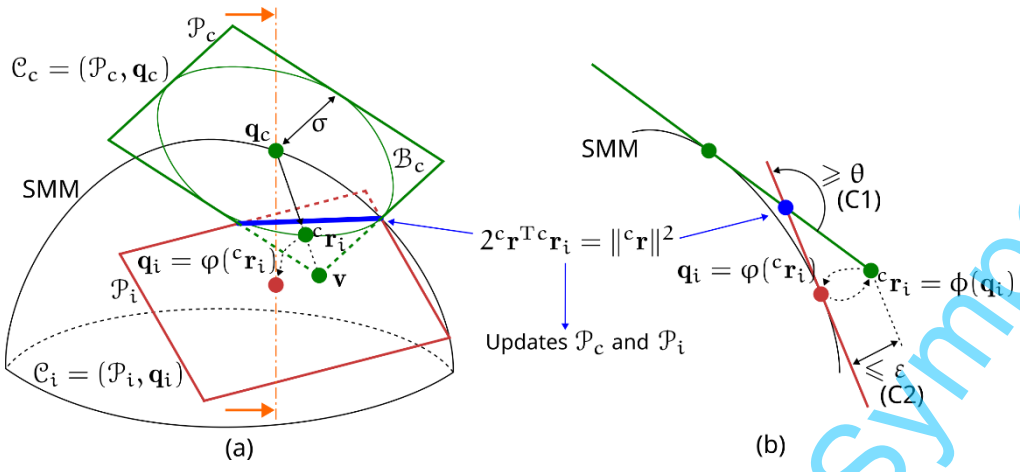

Fig. 5. Illustration of the continuation method for computing 2D self-motion manifolds. (b) is a cross-section defined by the plane represented in (a) in dash-and-dotted orange.

3.2.1. *Charts*

Given a point $\mathbf{q}_c$ on the manifold, a chart C_c provides a local parametrization in the neighbourhood of $\mathbf{q}_c$. This is achieved via a mapping $\varphi: \mathbb{R}^r \to \mathbb{R}^n$, which maps local r-dimensional coordinates in the chart to n-dimensional joint coordinates, with $\varphi(\mathbf{0}) = \mathbf{q}_c$.

Following [7] and in analogy to Section 4.1, the mapping $\mathbf{q}_i = \varphi_c(^c\mathbf{r}_i)$ is constructed in two steps:

1. Given a local coordinate vector $^c\mathbf{r}_i = [\rho_1, \dots, \rho_r]$ (expressed relative to chart C_c, denoted by superscript ^{c}), a candidate point $\hat{\mathbf{q}}_i$ in the joint space is computed by stepping in the tangent space:

$$\hat{\mathbf{q}}_i = \mathbf{q}_c + \mathbf{\Phi}_c \, ^c\mathbf{r}_i, \tag{15}$$

   where $\mathbf{\Phi}_c$ is an orthonormal basis for the tangent space of the manifold at $\mathbf{q}_c$.

2. The candidate $\hat{\mathbf{q}}_i$ is then projected onto the manifold by solving

$$\mathbf{f}(\mathbf{x}, \hat{\mathbf{q}}_i) = \mathbf{0} \tag{16}$$

   using the Newton-Raphson method as described in Section 4.1, yielding at convergence the corrected point $\mathbf{q}_i$.

Note that our approach differs slightly from the original method, which solves the following system of equations for correcting $\hat{\mathbf{q}}_i$:

$$\begin{cases} \mathbf{f}(\mathbf{x}, \hat{\mathbf{q}}_i) = \mathbf{0} \\ \mathbf{\Phi}_c^T(\mathbf{q}_i - \hat{\mathbf{q}}_i) = 0 \end{cases} \tag{17}$$

This formulation ensures that the correction step projects orthogonally to the tangent space $\mathbf{\Phi}_c$ of the manifold at $\mathbf{q}_c$. In contrast, our approach (by employing the Moore-Penrose pseudoinverse, which minimizes the change in $\hat{\mathbf{q}}_i$), updates the candidate configuration by moving it towards the manifold along a direction perpendicular to the manifold (as it is the least-norm solution) at the convergence point $\mathbf{q}_i$, i.e., perpendicular to $\mathbf{\Phi}_i$.

The inverse mapping $\phi_c: \mathbb{R}^n \to \mathbb{R}^r$, which recovers local chart coordinates from a joint configuration, is given by:

$$^c\mathbf{r}_i = \phi_c(\mathbf{q}_i) = \mathbf{\Phi}_c^T(\mathbf{q}_i - \mathbf{q}_c), \tag{18}$$

which projects the joint configuration $\mathbf{q}_i$ onto the tangent space $\mathbf{\Phi}_c$.

3.2.2. *Atlas of charts*

An atlas is a coordinated collection of charts that together cover the entire manifold. An atlas A is defined as a set of charts $A = \{C_1, C_2, \dots\}$, where each chart C_c is centred at a point $\mathbf{q}_c$ (i.e., $\varphi_c(\mathbf{0}) = \mathbf{q}_c$), as described in Section 3.2.1.

In [8]'s algorithm, each chart C_c is associated with an r-dimensional polytope $\mathcal{P}_c$ that defines its region of validity. This polytope is initially a hypercube enclosing an r-dimensional hyperball $\mathcal{B}_c$ of radius σ centered at $\mathbf{q}_c$, as illustrated in Fig. 5.

A chart is considered *bounded* only when every vertex of its polytope $\mathcal{P}_c$ lies within the hyperball $\mathcal{B}_c$. To bound the chart C_c, the algorithm iteratively selects a vertex $\mathbf{v}$ of $\mathcal{P}_c$ that is outside $\mathcal{B}_c$, and generates a local coordinate ${}^c\mathbf{r}_i$ in chart C_c as:

$$ {}^c\mathbf{r}_i = \alpha\sigma \frac{\mathbf{v}}{\|\mathbf{v}\|}, \tag{19} $$

where α is a scaling factor (initially 1) ensuring ${}^c\mathbf{r}_i$ lies within $\mathcal{B}_c$.

This local coordinate is mapped to joint space using φ_c, yielding $\mathbf{q}_i = \varphi_c({}^c\mathbf{r}_i)$. The validity of $\mathbf{q}_i$ is checked by two conditions:

$$ \|\mathbf{q}_i - \mathbf{q}_c + \mathbf{\Phi}_c\, {}^c\mathbf{r}_i\| \leq \varepsilon \qquad \text{(C1)} \tag{20} $$

$$ \|\mathbf{\Phi}_c^T \mathbf{\Phi}_i\| \geq \cos\theta \qquad \text{(C2)} \tag{21} $$

Condition (C1) controls the deviation from the tangent space, while (C2) ensures the angle between the tangent spaces at C_c and C_i is not too large, which would indicate the need for more charts in regions of high curvature.

If both conditions are satisfied, $\mathbf{q}_i$ is used to initialize a new chart C_i with its own local parameterization φ_c and tangent space $\mathbf{\Phi}_i$. If not, α is reduced by a user-defined factor β (e.g., $\beta = 0.8$), and the process repeats until a valid $\mathbf{q}_i$ is found or α reaches a minimum threshold. Starting with a larger α (i.e., 1) allows the algorithm to attempt larger charts before subdividing, improving coverage efficiency.

When a new chart C_i is created, the algorithm coordinates neighbouring charts C_c and C_i by adding the following inequality constraint to $\mathcal{P}_c$:

$$ 2\, {}^c\mathbf{r}\, {}^c\mathbf{r}_i = \|\, {}^c\mathbf{r}_i\|^2 \tag{22} $$

This constraint restricts the validity region of C_c by cropping a half-space defined by the hyperplane orthogonal to ${}^c\mathbf{r}_i$ at ${}^c\mathbf{r}_i$. Fig. 5 shows the new chart C_i in red and the restricted region in blue. Importantly, this boundary constraint is applied to every chart in the atlas A for which $\mathbf{q}_i$ is a valid point, i.e., for every chart C_j where ${}^j\mathbf{r}_i = \phi_j(\mathbf{q}_i)$ is within $\mathcal{P}_j$.

This iterative process continues until all vertices of every polytope $\mathcal{P}_c$ are contained within their respective hyperballs $\mathcal{B}_c$, at which point all charts are considered bounded and the atlas fully covers the self-motion manifold.

Additional constraints, such as joint limits or obstacle avoidance, can be incorporated by introducing further conditions ((C3), (C4), etc.) in the same manner as (C1) and (C2), ensuring that all generated points $\mathbf{q}_i$ remain within the feasible region defined by these constraints.

5. Discussion

In this section, we compare the performance of the three methods for computing self-motion manifolds: the sweeping method, the continuation method, and the cellular automata method. Sweeping and continuation methods have been implemented in Python, on a machine with an Intel Core i7-9700F CPU and 32 GB of RAM. The cellular automata method has not been implemented by us, but we have used the results provided in [11] for comparison.

The cellular automata (CA) method for computing self-motion manifolds, as proposed in [11], models the robot's configuration space (C-space) as a grid of high-dimensional elements. Instead of exhaustively sampling or clustering the entire C-space, the method uses a grid-based evolution search strategy inspired by cellular automata. Each grid element represents a continuous region of joint space, and its state evolves according to simple transition rules based on whether it contains part of the self-motion manifold. The computation begins by identifying a set of seed elements that intersect the manifold. These elements then propagate the search to their neighbours, iteratively expanding the set of elements identified as containing the manifold. This process continues until no new elements are found.

For consistency and fairness in our comparison, we sought to visually match the resolution used in [11], as the original work did not specify the discretization parameters. In our implementations of both the sweeping method and the higher-dimensional continuation method, we carefully selected comparable resolutions. Specifically, the sweeping method was

implemented by discretizing each axis of the redundant joints (q_4, q_5) into $N = 300$ points. The continuation method was executed using the following parameters: $\theta = \frac{\pi}{6}$, $\varepsilon = 0.1$, $\sigma = 0.3$.

Fig. 6 illustrates the two-dimensional self-motion manifold computed by each method for the studied 5-DOF robot when the end-effector is at the position $\mathbf{x} = [0.2, 0, 0.3]$ m. Fig. 6(a) and Fig. 6(c) show the results of our implementations of the sweeping and continuation methods, respectively; while Fig. 6(d) shows the result of the CA method, directly taken from [11]. Note how the manifolds produced by the sweeping method in Fig. 6(a) present regions of discontinuities in the areas where the manifold is tangent to the horizontal lines defined by fixed values of the redundant joints (q_4, q_5). A solution to this problem is to increase the resolution, which produces the results presented in Fig. 6(b) when $N = 6000$.

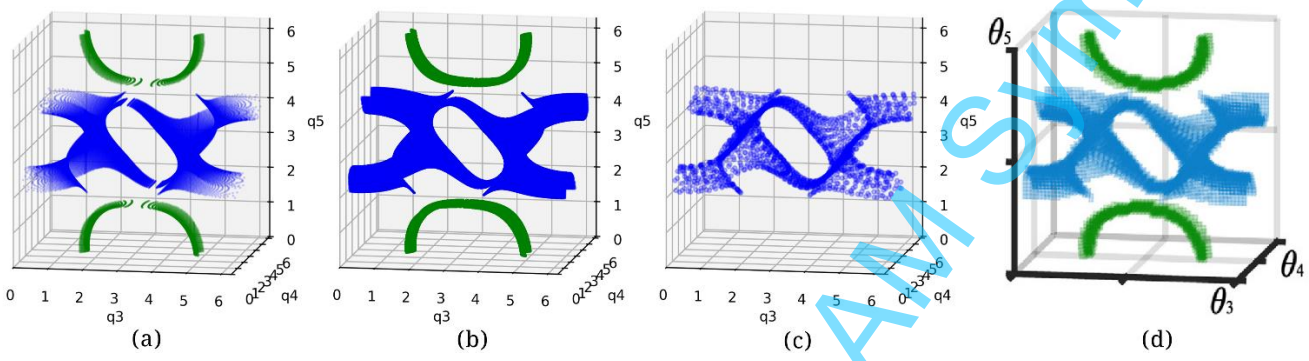

Fig. 6. Results of the different methods for computing SMMs: sweeping (a-b), continuation (c), cellular automata (d).

Table 2 shows the computation times for each method, where the CA results are directly taken from [11], and the sweeping and continuation times are averaged over 50 runs. The results of our comparative study reveal distinct strengths and limitations for each method of computing self-motion manifolds in the 5-DOF robot.

Method	Runtime (s)	Number of points
Sweeping [10]	0.46	14,4648
Sweeping [10] ($N = 6000$)	165.46	5.7×10^7
Higher-dimensional continuation [8]	11.9	2,227
Cellular automata [11]	11347	N/A

Table 2. Performance of the different methods for computing SMMs.

The sweeping method stands out as the fastest approach by a significant margin, with computation times orders of magnitude lower than the other methods. Moreover, it is able to identify all disjoint self-motion manifolds present in the configuration space. The primary limitation of the sweeping method is that it does not densely sample regions where the manifold is tangent to the horizontal lines defined by fixed values of the redundant joints (q_4, q_5). This can result in sparse point clouds in certain areas of the solution set. However, this drawback can be mitigated by increasing the resolution of the sweep or by employing redensification strategies, which we propose as a direction for future work. Another drawback is the requirement of deriving an analytical solution of the inverse kinematics, as done in Section 2.

The higher-dimensional continuation method offers a more uniform and dense coverage of the self-motion manifold. Unlike the sweeping method, it does not require an analytical solution to the inverse kinematics; it only relies on the Jacobian, which can be computed numerically if necessary. This flexibility comes at the cost of increased computational time, although it remains substantially faster than the cellular automata approach. A notable limitation of the continuation method is that it only explores the manifold connected to the initial seed configuration, and cannot identify all disjoint manifolds without multiple initializations.

The cellular automata (CA) method is the slowest by a wide margin, with computation times exceeding 10,000 seconds. While it is capable of identifying all disjoint self-motion manifolds, the computational cost is prohibitive for practical applications. The method's exhaustive grid-based search ensures completeness, but the trade-off in efficiency is substantial. We recommend its use when the identification of all disjoint manifolds is paramount, but no analytical solution to the inverse kinematics problem is available and computation speed is not critical.

6. Conclusion and Future Work

This paper has presented a comparative evaluation of three representative methods for computing two-dimensional self-motion manifolds in a 5-DOF redundant robot: the sweeping method, the higher-dimensional continuation method,

and the cellular automata approach. Our results demonstrate that the sweeping method is the most efficient, providing rapid identification of all disjoint manifolds with minimal computational cost. The continuation method, while slower, offers dense coverage of the manifold and does not require an analytical inverse kinematic solution, making it suitable for complex robots where only the Jacobian is available. The cellular automata method, although exhaustive and complete, requires prohibitive computational times when compared to the other methods.

Future work will focus on addressing the limitations identified in each method. For the sweeping approach, we plan to investigate redensification strategies to improve sampling density in regions where the manifold is tangent to the discretized horizontal lines. Additionally, we will explore new methods for computing self-motion manifolds that combine the strengths of the studied methods, while addressing their issues.

Overall, our study provides guidance for researchers and practitioners in selecting appropriate tools for redundancy analysis and exploitation in industrial robotic systems, and lays the groundwork for further advances in efficient and comprehensive computation of self-motion manifolds.

7. Acknowledgments

Work supported by grant PRE2021-099226, funded by MCIN/AEI/10.13039/501100011033 and the ESF+, and by project PID2024-159765OA-I00, funded by the State Research Agency of the Spanish Government.

8. References

[1] Diek, D., Stuja, K., & Aburaia, M. (2022). Design and development of a flexible, cable-controlled and hyper redundant robot with Strength-Adjustable ball joints. In Proceedings of the 33rd DAAAM International Symposium (pp. 0024-0028).
[2] Flacco, F., De Luca, A., & Khatib, O. (2015). Control of redundant robots under hard joint constraints: Saturation in the null space. IEEE Transactions on Robotics, 31(3), 637-654.
[3] Faroni, M., Beschi, M., Pedrocchi, N., & Visioli, A. (2018). Predictive inverse kinematics for redundant manipulators with task scaling and kinematic constraints. IEEE Transactions on Robotics, 35(1), 278-285.
[4] Albu-Schäffer, A., & Sachtler, A. (2023). Redundancy resolution at position level. IEEE Transactions on Robotics, 39(6), 4240-4261.
[5] Fabregat-Jaén, M., Peidró, A., Colombo, M., Rocco, P., & Reinoso, Ó. (2025). Topological and spatial analysis of self-motion manifolds for global redundancy resolution in kinematically redundant robots. Mechanism and Machine Theory, 210, 106020.
[6] Wenger, P., Chedmail, P., & Reynier, F. (1993, May). A global analysis of following trajectories by redundant manipulators in the presence of obstacles. In [1993] Proceedings IEEE International Conference on Robotics and Automation (pp. 901-906). IEEE.
[7] Burdick, J. W. (1989, May). On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds. In Advanced Robotics: 1989: Proceedings of the 4th International Conference on Advanced Robotics Columbus, Ohio, June 13–15, 1989 (pp. 25-34). Berlin, Heidelberg: Springer Berlin Heidelberg.
[8] Henderson, M. E. (2002). Multiple parameter continuation: Computing implicitly defined k-manifolds. International Journal of Bifurcation and Chaos, 12(03), 451-476.
[9] DeMers, D., & Kreutz-Delgado, K. (1994, May). Canonically parameterized families of inverse kinematic functions for redundant manipulators. In Proceedings of the 1994 IEEE International Conference on Robotics and Automation (pp. 1881-1886). IEEE.
[10] Peidro, A., Reinoso, O., Gil, A., Marín, J. M., & Paya, L. (2018). A method based on the vanishing of self-motion manifolds to determine the collision-free workspace of redundant robots. Mechanism and Machine Theory, 128, 84-109.
[11] Wu, T., Zhao, J., & Xie, B. (2023). A novel method for computing self-motion manifolds. Mechanism and Machine Theory, 179, 105121.
[12] Zhou, Z., Zhao, J., Zhang, Z., & Li, X. (2023). Motion planning method of redundant dual-chain manipulator with multiple constraints. Journal of Intelligent & Robotic Systems, 108(4), 69.
[13] Banfield, I., & Rodríguez, H. (2021, August). Generation of the self-motion manifolds of a functionally redundant robot using multi-objective optimization. In Climbing and Walking Robots Conference (pp. 438-452). Cham: Springer International Publishing.
[14] Yang, Y., Wu, Y., & Pan, J. (2021). An Interval Branch-and-Bound-Based Inverse Kinemetics Algorithm Towards Global Optimal Redundancy Resolution. arXiv preprint arXiv:2104.12183.
[15] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In kdd (Vol. 96, No. 34, pp. 226-231).
[16] Jaillet, L., & Porta, J. M. (2012). Path planning under kinematic constraints by rapidly exploring manifolds. IEEE Transactions on Robotics, 29(1), 105-117.