Redundancy Resolution in Multiple Feasibility Maps via MultiFM-RRT

Marc Fabregat-Jaén[1][a], Adrián Peidró[1][b], María Flores[1][c], Luis Payá[1,2][d] and Óscar Reinoso[1,2][e]
[1]*Instituto de Investigación en Ingeniería de Elche, Universidad Miguel Hernández, Avda. Universidad s/n, Elche, Spain*
[2]*ValgrAI: Valencian Graduate School and Research Network of Artificial Intelligence, Camí de Vera s/n, Valencia, Spain*
fl

Keywords: Redundancy Resolution, RRT, MultiFM-RRT, Feasibility Maps, Redundant Manipulators.

Abstract: This paper presents MultiFM-RRT, a novel algorithm for redundancy resolution in kinematically redundant manipulators based on the exploration of multiple Feasibility Maps (FMs). They encode all valid configurations of redundant variables for a prescribed task trajectory, enabling global optimization and constraint satisfaction. Unlike traditional velocity-based methods, which are limited to local solutions, and grid-based methods, which are computationally intensive, MultiFM-RRT leverages the Rapidly-exploring Random Tree (RRT) framework to efficiently explore the space of feasible solutions across multiple feasibility maps. The algorithm incorporates singularity maps to enable transitions between different aspects, ensuring comprehensive coverage of the solution space. By computing feasibility maps online and guiding exploration with probabilistic sampling of goal and singularity sets, MultiFM-RRT achieves a balance between computational efficiency and global optimality. The proposed approach is demonstrated on a Stewart parallel platform, illustrating its ability to generate feasible, constraint-satisfying trajectories while efficiently handling redundancy.

1 INTRODUCTION

Kinematically redundant robots are characterized by having more degrees of freedom (DoF) n than the dimension m of the main or *primary* task they are required to perform. For instance, a planar manipulator with three joints (3-DoF) that must control the end-effector position (p_x, p_y) in the plane exhibits one degree of redundancy ($r = 1$), since only two DoF are necessary for this task. This surplus of DoF enhances the manipulator's versatility, as it can be exploited to fulfill additional *secondary* objectives or constraints, such as minimizing joint torques or energy, or avoiding singularities and obstacles (Peidro and Haug, 2023). However, this flexibility comes at the cost of increased computational complexity: the inverse kinematics problem (IKP) (i.e, determining the joint displacements required to achieve a desired end-effector position) admits infinitely many solutions. The challenge of selecting a suitable solution from this infinite set is known as the *redundancy resolution* problem.

In practice, redundancy resolution is typically addressed for a prescribed trajectory of the task variables. Given a desired task trajectory $\mathbf{x}(t)$, $0 < t < t_g$, a redundancy resolution algorithm must generate the time evolution $\mathbf{q}(t)$ of all n joint coordinates, taking into account the infinite solution set at each instant and any imposed constraints or secondary objectives. Approaches to this problem are generally classified as either velocity-based or position-based.

Velocity-based methods rely on the differential relationship $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$, where $\mathbf{J}$ is the non-square $m \times n$ Jacobian matrix. These methods, including pseudoinverse and optimization-based techniques, compute joint velocities $\dot{\mathbf{q}}$ and then integrate them. Pseudoinverse approaches (Whitney, 1969) yield $\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\dot{\mathbf{x}}$ plus a nullspace term for optimizing secondary criteria (Kazemipour et al., 2022), with $\mathbf{J}^{\dagger}$ denoting the Moore-Penrose pseudoinverse. Optimization-based methods formulate a minimization problem for $\dot{\mathbf{q}}$, subject to the equality constraint $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$, thus avoiding explicit inversion (Zanchettin and Rocco, 2017). The main strengths of velocity-based methods are their simplicity and suitability for real-time control. However, they are typically limited to locally optimal solutions, may encounter issues with non-holonomy and singularities (when using the pseudoinverse), and can struggle to enforce constraints that are more natu-

[a] https://orcid.org/0009-0002-4327-0900
[b] https://orcid.org/0000-0002-4565-496X
[c] https://orcid.org/0000-0003-1117-0868
[d] https://orcid.org/0000-0002-3045-4316
[e] https://orcid.org/0000-0002-1065-8944

Fabregat-Jaén, M., Peidró, A., Flores, M., Payá, L. and Reinoso, Ó.
Redundancy Resolution in Multiple Feasibility Maps via MultiFM-RRT.
DOI: 10.5220/0013835700003982

In *Proceedings of the 22nd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2025) - Volume 2*, pages 158-165
ISBN: 978-989-758-770-2; ISSN: 2184-2809

rally expressed at the position level, such as collision avoidance (Peidro and Haug, 2023).

Position-based methods address these limitations by working directly at the configuration level. Here, redundancy resolution involves identifying all sets of joint coordinates $\mathbf{q}$ that achieve a given task value $\mathbf{x}$. These solution sets are, in general, manifolds known as *self-motion manifolds* (Burdick, 1989). By exploring all self-motion manifolds for a given $\mathbf{x}$, one can identify the configuration $\mathbf{q}^*$ that globally optimizes secondary objectives, enabling globally optimal redundancy resolution (Fabregat-Jaén et al., 2025). This approach, however, is computationally demanding, as the computation of self-motion manifolds is nontrivial (Albu-Schäffer and Sachtler, 2023). A related strategy is provided by Feasibility Maps (FMs) (Wenger et al., 1993; Reveles et al., 2016), which represent all feasible values of a set of r redundant variables at each instant t along the trajectory. FMs facilitate trajectory planning in the space of redundant variables, allowing for the avoidance of obstacles and singularities while optimizing additional criteria.

Several researchers have proposed the use of FMs to address the redundancy resolution problem. For example, (Pámanes G et al., 2002) proposed using FMs to plan collision-free trajectories while minimizing the change in the free parameter. More recently, (Ferrentino and Chiacchio, 2020) presented a method to globally solve the redundancy resolution problem by performing an exhaustive grid search across all FMs, which is computationally expensive. In contrast, (Fabregat-Jaen et al., 2023) proposed a more time-efficient approach based on the Rapidly-exploring Random Tree (RRT) algorithm. In their method, only the FM corresponding to the initial configuration is explored, and the FM is computed on-the-fly as the algorithm progresses, eliminating the need for precomputation. In (Fabregat-Jaén et al., 2024), the concept of FMs was extended to include the notion of *augmented feasibility maps*, which incorporate the task dimension $\mathbf{x}$ into the FM space, allowing for simultaneous path planning and redundancy resolution.

In this paper, we introduce MultiFM-RRT, a novel algorithm that aims to strike a balance between the two aforementioned approaches. Similar to (Ferrentino and Chiacchio, 2020), MultiFM-RRT explores all FMs, but it does so more efficiently by leveraging the RRT algorithm. Additionally, it computes the FMs online, as in (Fabregat-Jaen et al., 2023), thereby avoiding the need for precomputation and improving computational efficiency.

The rest of the paper is organized as follows. Section 2 introduces the concept of feasibility maps and important related concepts. Section 3 presents the MultiFM-RRT algorithm, which extends the RRT framework to explore multiple feasibility maps simultaneously. Section 4 provides an example of the algorithm in action, and Section 5 concludes the paper with a summary of the contributions and future work.

2 FEASIBILITY MAPS

2.1 Inverse Kinematics Problem

Equation (1) defines the constraint that relates the joint space $\mathbf{q} \in \mathbb{R}^n$ and the task space $\mathbf{x} \in \mathbb{R}^m$:

$$\mathbf{F}(\mathbf{x}, \mathbf{q}) = \mathbf{0}_{m\times 1} \tag{1}$$

The IKP entails finding a joint configuration $\mathbf{q}$ that satisfies Equation (1) for a given task-space point $\mathbf{x}$. However, even for non-redundant robots (where $n = m$), the IK mapping is multi-valued in general. This means that there exist multiple joint-space points $\mathbf{q}$ that yield the same workspace point $\mathbf{x}$.

To better understand this property, the concept of *aspects* is introduced. As defined by (Borrel and Liégeois, 1986), aspects are connected sets of joint-space points where the Jacobian matrix $\mathbf{J}(\mathbf{x}, \mathbf{q}) = \frac{\partial \mathbf{F}}{\partial \mathbf{q}}$ (hereafter $\mathbf{J}$) mantains full rank m. In other words, aspects define subsets of the joint space where singularities do not exist. Following this definition, it is directly derived that transitions between aspects are marked by singularities, at which point $\mathbf{J}$ loses rank.

The multi-valuedness of the IK is exacerbated for redundant manipulators, where the IKP becomes underdetermined: as $n > m$, there are more unknowns than equations from which to solve the IKP. When redundancy arises, the degree of redundancy r is defined as the difference between the dimensionality of joint and task spaces: $r = n - m$.

2.2 Task Space Augmentation

When kinematic redundancy arises, by fixing r so-called *free parameters* to known values, the IKP becomes determined, and a unique solution can be found for each aspect. Note that the free parameters can be chosen arbitrarily, as long as they are a differentiable function of the joint space coordinates $\mathbf{q}$, and independent of the task space coordinates $\mathbf{x}$. However, in most cases, the best selection of free parameters is simply r joint variables to which values are freely assigned. For simplicity, for the rest of the paper, we will consider this case, and refer to the free parameters as $\mathbf{q}_r \in \mathbb{R}^r$, where $\mathbf{q}_r$ consists of r components of

$\mathbf{q}$. This addition of free parameters allows us to define an augmented task space $\mathbf{x}_a = (\mathbf{x}, \mathbf{q}_r) \in \mathbb{R}^{m+r}$.

In (Pámanes G et al., 2002), the consequences of introducing the free parameters $\mathbf{q}_r$ are analyzed. It is shown that the newly introduced rows in $\mathbf{J}$ give rise to additional singularities that are not present in the original Jacobian. As a result, new sets of aspects, referred to as *extended aspects*, are produced. This implies that, depending on the selection of $\mathbf{q}_r$, the number of extended aspects can vary.

The introduction of the augmented task space enables the definition of a unique inverse kinematics function, denoted as $\mathbf{g}(\cdot)$. This function maps each point in the augmented task space to a unique joint configuration within a specific extended aspect:

$$\mathbf{q} = \mathbf{g}(\mathbf{x}, \mathbf{q}_r, a) \tag{2}$$

where a is determines the specific extended aspect to which the returned joint configuration $\mathbf{q}$ belongs.

2.3 Feasibility Maps

Rather than treating the IKP as a pointwise problem, it is often formulated as the tracking of a continuous task. In this context, the objective is to find a continuous path $\mathbf{q}(t)$ in the joint space that follows a given trajectory $\mathbf{x}(t)$ in the task space. This trajectory is typically parameterized by an arc-length parameter t, which, for simplicity, we refer to as *time*. With this parametrization, once the task trajectory is specified, the task dimension m can effectively be reduced to 1, since the corresponding task space point $\mathbf{x}$ at any given time t is obtained by evaluating $\mathbf{x}(t)$. The concept presented next builds upon this idea.

According to (Wenger et al., 1993), the concept of *feasibility maps* (FMs) represents the set of all joint configurations $\mathbf{q}$ that satisfy the IKP for a given task trajectory $\mathbf{x}(t)$ and a selection of free parameters $\mathbf{q}_r$, in an $r+1$ dimensional space. For each possible selection of the extended aspect a, there exists a corresponding FM. Formally, for each a, the FM is defined as the connected set of points in the $(t, \mathbf{q}_r)$ space that, when solving the IKP via Equation (2) for the given a, yield a valid joint configuration $\mathbf{q}$:

$$\mathcal{FM}_a = \left\{ \begin{bmatrix} t \\ \mathbf{q}_r \end{bmatrix} \;\middle|\; \mathbf{q} = \mathbf{g}(\mathbf{x}(t), \mathbf{q}_r, a),\ \mathbf{q} \in \mathcal{Q} \right\} \tag{3}$$

where $\mathcal{Q}$ denotes the set of valid joint configurations. The domain of $\mathcal{Q}$ is determined by the robot's kinematic constraints, as well as any additional task-imposed constraints (e.g., joint limits, collision avoidance, etc.).

Figure 1 shows the feasibility maps for a 2-DoF planar manipulator represented in Figure 2. The robot's end effector is tasked with following a parabolic trajectory in the y axis: $\mathbf{x}(t) = p_y(t) = -6.662t^2 + 8.162t - 1.5$ with $t \in [0,1]$, while avoiding an ellipical obstacle. Joints are enforced to remain within the limits $q_i \in [-\pi, \pi]$ for $i = 1,2$. Having $r = 1$, a single free parameter is selected, which corresponds to the first joint variable q_1. The inverse kinematics function $\mathbf{g}(\cdot)$ for this case is:

$$\mathbf{q} = \begin{bmatrix} q_1 \\ \frac{\pi}{2} + \sigma(a)\left(\frac{\pi}{2} - \arcsin(p_y(t) - \sin q_1)\right) - q_1 \end{bmatrix} \tag{4}$$

where $\sigma(a) = \pm 1$ is a sign function that determines the resolution of the second joint variable q_2 based on the extended aspect a.

FMs in Figure 1 are computed by sweeping plane $(t, \mathbf{q}_1)$ between its ranges ($t \in [0,1]$ and $\mathbf{q}_1 \in [-\pi, \pi]$) with a given discretization step. For each point $(t, \mathbf{q}_1)$, $\mathbf{q}$ is computed using Equation (4). If the evaluation of a point results in non-real values (e.g., $|p_y(t) - \sin q_1| > 1$), the point is coloured purple. Similarly, if the point results in a joint configuration that collides with the environment, it is coloured red. Finally, if the point results in a joint configuration that violates the joint limits, it is coloured yellow. The remaining points, which are left uncoloured, represent feasible joint configurations, and conform the actual FM for the given extended aspect a.

2.4 Singularity Maps

(Wenger et al., 1993) introduced the concept of *singularity maps* (SMs), which represent the set of all singular points in the $(t, \mathbf{q}_r)$ space that lie at the boundaries between different aspects. SMs are adjacent to feasibility maps (FMs) and serve as *gateways* through which the robot can transition from one FM to another by passing through a singularity.

To compute the SMs, the time parameter t is swept, and for each t, the values of the free parameters $\mathbf{q}_r$ that produce singularities are determined. Specifically, for the recurring example, singularities occur when $|p_y(t) - \sin q_1| = 1$, which corresponds to the boundary of the domain of the arcsin function in Equation (4). These SMs are shown in orange in Figure 1.

3 MULTIFM-RRT

The MultiFM-RRT algorithm proposed in this paper extends the original RRT framework to enable the simultaneous exploration of multiple feasibility maps (FMs). Algorithm 1 summarizes the main steps of

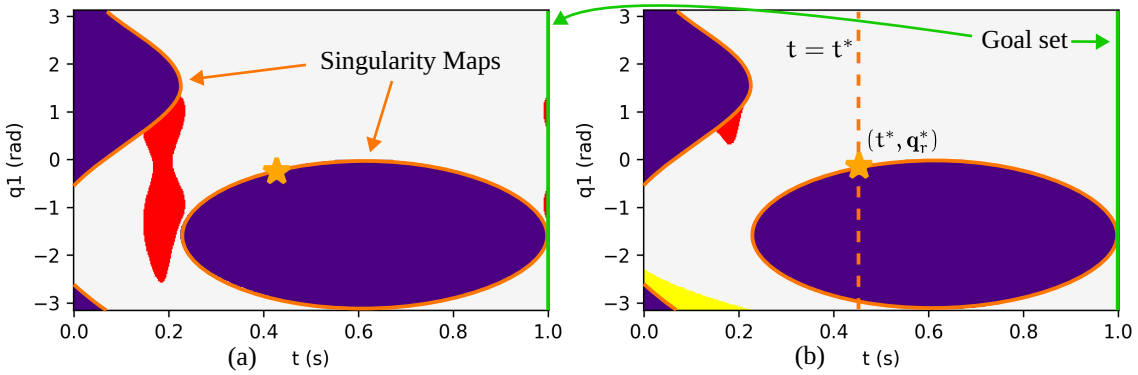


Figure 1: Feasibility maps for a 2-DoF planar manipulator. (a) FM for the first extended aspect ($\sigma = 1$). (b) FM for the second extended aspect ($\sigma = -1$). Colour regions indicate sets of invalid joint configurations: red for collisions, yellow for joint limits, and purple for complex solutions. Singularity maps are shown in orange, and goal sets are shown in green.

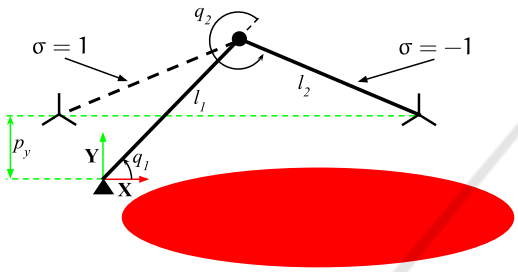


Figure 2: 2-DoF planar manipulator tasked with task $\mathbf{x} = p_y$.

Algorithm 1: MultiFM-RRT algorithm.

Initialize tree $\mathcal{T}$ with root node $\mathbf{n}_0 = \mathbf{s}_0$
for $i = 1, 2, \ldots, i_{max}$
  $\mathbf{s}_{random} \leftarrow$ SAMPLESTATE(α, β)
  $\mathbf{n}_{near} \leftarrow$ NEARESTNODE($\mathbf{s}_{random}$, $\mathcal{T}$)
  $\mathbf{n}_{new} \leftarrow$ STEER($\mathbf{n}_{near}$, $\mathbf{s}_{random}$, Δs)
  if FEASIBLECONNECTION($\mathbf{n}_{near}$, $\mathbf{n}_{new}$)
    Add $\mathbf{n}_{new}$ to $\mathcal{T}$ with parent $\mathbf{n}_{near}$
  end
end
$\mathcal{P} \leftarrow$ BESTPATH($\mathcal{T}, c(\cdot)$)

MultiFM-RRT. While the overall structure remains similar to the standard RRT, the key differences lie in the adaptation of specific procedures to accommodate the characteristics of FMs and the requirements of multi-map exploration.

First, the algorithm initializes a tree $\mathcal{T}$ with a root node $\mathbf{n}_0$ (without parent) representing the initial state $\mathbf{s}_0$ of the robot. A state $\mathbf{s} = [t, \mathbf{q}_r^{\mathrm{T}}, a]$ is defined as a point in the space $(t, \mathbf{q}_r)$ of a specific $\mathcal{FM}_a$. Remember that, given a state $\mathbf{s}$, the corresponding joint configuration $\mathbf{q}$ can be directly obtained by solving the IKP using Equation (2). A node $\mathbf{n}$ is a state $\mathbf{s}$ that has been added to the tree $\mathcal{T}$ with an associated parent, and is defined as a pair $\mathbf{n} = (\mathbf{s}, \mathbf{n}_{parent})$, where $\mathbf{n}_{parent}$ is the parent node of $\mathbf{n}$ in the tree.

The algorithm then enters its main loop, which runs for up to i_{max} iterations. This value sets an upper limit on the number of nodes that can be added to the tree $\mathcal{T}$; however, the actual number of nodes will typically be less, since many sampled states may be infeasible and therefore not included. In Section 4, we analyze how varying i_{max} influences the algorithm's performance and solution quality.

At each iteration, the algorithm samples a random state $\mathbf{s}_{random}$ from the FM space using the SAMPLESTATE procedure. By default, this procedure generates a random point in the $(t, \mathbf{q}_r)$ space of a randomly selected feasibility map $\mathcal{FM}_a$, with the extended aspect a chosen uniformly at random. In addition, two parameters, α and β, control the probability of sampling from specialized subsets: the *goal set* and the singularity maps (SMs), respectively. These probabilites are usually kept low, e.g., $\alpha = \beta = 0.05$ (5%).

The goal set, sampled with probability α, consists of all states $\mathbf{s} = [t, \mathbf{q}_r^{\mathrm{T}}, a]$ for which the task trajectory $\mathbf{x}(t)$ is completed, i.e., $t = t_{goal}$, where t_{goal} denotes the final time of the trajectory. In Figure 1, the goal set is represented by the green vertical lines at $t = 1$, which correspond to the end of the task trajectory. The goal set is used to guide the exploration towards the completion of the task.

Conversely, with probability β, the algorithm samples from the SMs: it selects a random time t^*, computes the intersections $\mathbf{q}_r^*$ between the hyperplane $t = t^*$ and the SMs, and randomly selects one of these intersections, such that $(t^*, \mathbf{q}_r^*)$ corresponds to a singularity, as defined in Section 2.4, and illustrated in Figure 1. Note that, by definition, singular points do not belong to any aspect, since they form the boundaries between aspects. As a result, a sampled state $\mathbf{s}_{random}$ from the SMs will not be associated with a valid extended aspect a. This property, and sampling

probability, is later leveraged to enable transitions between different FMs.

After sampling a state $\mathbf{s}_{random}$, the algorithm identifies the nearest node $\mathbf{n}_{near}$ in the tree $\mathcal{T}$ using the NEARESTNODE procedure. This procedure performs a nearest-neighbor search according to a user-defined cost function $c(\cdot)$, which, for simplicity, we have chosen as the Euclidean distance in the $(t, \mathbf{q}_r)$ space. Crucially, the search is restricted to two particular conditions:

1. The nearest node $\mathbf{n}_{near}$ must belong to the same FM as the sampled state $\mathbf{s}_{random}$, i.e., $a_{near} = a_{random}$. However, this condition is relaxed when either $\mathbf{s}_{random}$ or $\mathbf{n}_{near}$ belong to the SMs (i.e. have no defined aspect), in which case this condition is ignored.
2. The nearest node $\mathbf{n}_{near}$ must have a time value $t_{near} < t_{random}$, ensuring that the time parameter t remains monotonically increasing. This constraint ensures the completion of the task trajectory $\mathbf{x}(t)$ and prevents the algorithm from generating solutions that move backward in time.

Once the nearest node $\mathbf{n}_{near}$ is identified, the algorithm generates a new node $\mathbf{n}_{new}$ by steering from $\mathbf{n}_{near}$ toward the sampled state $\mathbf{s}_{random}$ using the STEER procedure. This procedure produces a new state $\mathbf{s}_{new}$ at a fixed distance Δs from $\mathbf{n}_{near}$ towards $\mathbf{s}_{random}$, controlling the resolution of the exploration. The assignment of the aspect a_{new} for $\mathbf{n}_{new}$ depends on whether either state is associated with a SM (i.e., has no defined aspect):

1. If $\mathbf{s}_{random}$ belongs to the SMs, then $\mathbf{n}_{new}$ inherits the aspect of $\mathbf{n}_{near}$ ($a_{new} = a_{near}$), unless $\mathbf{s}_{random}$ is within Δs of $\mathbf{n}_{near}$, in which case $\mathbf{n}_{new}$ is set to $\mathbf{s}_{random}$ with undefined aspect.
2. If $\mathbf{n}_{near}$ belongs to the SMs, then $\mathbf{n}_{new}$ inherits the aspect of $\mathbf{s}_{random}$ ($a_{new} = a_{random}$).

Finally, the algorithm checks whether the connection between $\mathbf{n}_{near}$ and $\mathbf{n}_{new}$ is feasible using the FEASIBLECONNECTION procedure. This procedure verifies that the path from $\mathbf{n}_{near}$ to $\mathbf{n}_{new}$ lies entirely within a FM, i.e., it does not cross complex-solution regions. This check is performed by discretizing the path using a given resolution Δf and evaluating the feasibility of each point along the path. Additionally, it checks that the path does not violate any additional constraints, such as joint limits or collisions. If the connection is feasible, the new node $\mathbf{n}_{new}$ is added to the tree $\mathcal{T}$ with $\mathbf{n}_{near}$ as its parent.

Once the tree $\mathcal{T}$ has been constructed by exhausting the maximum number of iterations i_{max}, the algorithm computes the best path $\mathcal{P}$ using the BESTPATH procedure. This procedure evaluates every path that joins the root node $\mathbf{n}_0$ to a node $\mathbf{n}_{goal}$ that belongs to the goal set, and selects the path with the lowest cost according to the user-defined cost function $c(\cdot)$. The resulting path $\mathcal{P}$ is a continuous trajectory in the joint space that satisfies the task trajectory $\mathbf{x}(t)$, since the time parameter t is monotonically increasing, reaches t_{goal}, and the path does not traverse infeasible regions. Note that the returned path $\mathcal{P}$ is polygonal due to the nature of the RRT algorithm. The smoothing of the path can be performed using any standard path smoothing technique, such as the one presented in (Fabregat-Jaén et al., 2024).

4 EXAMPLE

4.1 Application to a Stewart Platform

This section illustrates the application of the MultiFM-RRT algorithm to a Stewart platform. The considered Stewart platform, depicted in Figure 3, features a double-ring configuration: its six UPS (Universal-Prismatic-Spherical) linear actuators with lengths $q_1, q_2, \ldots, q_6$ are arranged in two concentric rings, each containing three actuators. The positions of the universal joints with respect to the fixed base center W are denoted $\mathbf{a}_j$ and are listed in Table 1. Similarly, the positions of the spherical joints relative to the mobile platform center Σ are given by $\mathbf{b}_j$ in Table 1. The pose of the mobile platform is described by $\mathbf{p} = (x, y, z, \alpha, \beta, \gamma)$, where (x, y, z) specifies the position of Σ with respect to W, and (α, β, γ) are the XYZ Euler angles defining the platform's orientation relative to the base. The relationship between the joint coordinates $\mathbf{q}$ and the platform pose $\mathbf{p}$ is:

$$\|\mathbf{R}(\alpha,\beta,\gamma)\mathbf{b}_j + [x,y,z]^{\mathrm{T}} - \mathbf{a}_j\|^2 - q_j^2 = 0, \;\; j = 1,\ldots,6 \tag{5}$$

where $\mathbf{R}(\alpha, \beta, \gamma)$ is the rotation matrix corresponding to the XYZ Euler angles (α, β, γ).

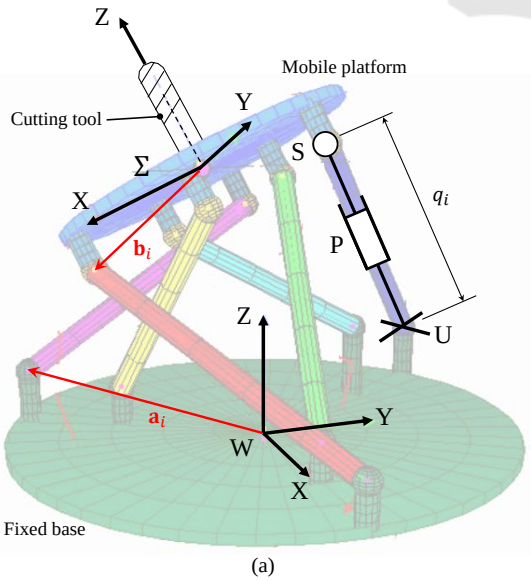
The robot is required to follow an arc-shaped trajectory in the (X, Y) plane, simulating a machining operation with a cutting tool along the Z axis of the mobile platform. Note that, since the cutting tool is rotating along the Z axis, the Euler angle γ is not relevant for this task, and the robot becomes kinematically redundant, where the objective is to track the following task trajectory:

$$\mathbf{x}(t) = \begin{bmatrix} x \\ y \\ z \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \cos(t) \\ \sin(t) \\ 0.8 \\ 0 \\ 0 \end{bmatrix}, \quad t \in [0, 0.8] \tag{6}$$

Table 1: Joint positions of the double-ring Stewart platform.

Joint j	$\mathbf{a}_j$	$\mathbf{b}_j$
1	$\begin{bmatrix} 0.6\cos(0) \\ 0.6\sin(0) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.6\cos(0) \\ 0.6\sin(0) \\ -0.05 \end{bmatrix}$
2	$\begin{bmatrix} 0.4\cos(0) \\ 0.4\sin(0) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.4\cos(180^\circ) \\ 0.4\sin(180^\circ) \\ -0.05 \end{bmatrix}$
3	$\begin{bmatrix} 0.6\cos(120^\circ) \\ 0.6\sin(120^\circ) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.6\cos(120^\circ) \\ 0.6\sin(120^\circ) \\ -0.05 \end{bmatrix}$
4	$\begin{bmatrix} 0.4\cos(120^\circ) \\ 0.4\sin(120^\circ) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.4\cos(300^\circ) \\ 0.4\sin(300^\circ) \\ -0.05 \end{bmatrix}$
5	$\begin{bmatrix} 0.6\cos(-120^\circ) \\ 0.6\sin(-120^\circ) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.6\cos(-120^\circ) \\ 0.6\sin(-120^\circ) \\ -0.05 \end{bmatrix}$
6	$\begin{bmatrix} 0.4\cos(-120^\circ) \\ 0.4\sin(-120^\circ) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.4\cos(-300^\circ) \\ 0.4\sin(-300^\circ) \\ -0.05 \end{bmatrix}$

An additional kinematic constraint is imposed on the robot: collisions between the linear actuators are prevented by ensuring that the distance between every pair of actuators remains greater than a minimum threshold, $d_{min} = 4$ cm. The length of the first linear actuator, q_1, is chosen as the free parameter, $\mathbf{q}_r = [q_1]$, and is restricted to the interval $[0.75, 1.75]$ m. With this selection of free parameter, the robot presents two FMs, corresponding to the two extended aspects a, which result from the two possible resolutions of γ from Equation (5) for $i = 1$ and for the given values of $\mathbf{x}(t)$ (Peidro et al., 2018).



(a)

Figure 3: Double-ring Stewart platform with six UPS linear actuators.

The MultiFM-RRT parameters are set as follows: the maximum number of iterations is $i_{max} = 2000$; the steering step size is $\Delta s = 0.5$ cm; the singularity and goal sampling biases are $\alpha = \beta = 0.05$; and the path discretization resolution for feasibility checks is $\Delta f = 0.02$.

Figure 4 displays the FMs computed for the Stewart platform during task execution. The same color scheme as in Figure 1 is applied. It is important to note that FMs are not precomputed, but are generated online as the algorithm progresses; they are shown here solely for illustration.

Figure 4 also depicts the tree $\mathcal{T}$ produced by a representative run of the MultiFM-RRT algorithm. The initial FM state, $\mathbf{n}_0 = [t, q_1, a] = [0, 1, 1]$, is marked with a green cross. Nodes belonging to the goal set are shown in green, while the selected solution path $\mathcal{P}$ is highlighted in blue. Nodes associated with the SMs are indicated in orange, acting as transition points between different FMs.

In this scenario, the MultiFM-RRT algorithm autonomously determines that, in order to reach the goal at $t = 0.8$, a transition to the second FM is required, since the first FM is obstructed by collision constraints. By leveraging a gateway node in the SMs, the algorithm transitions from the first to the second FM, successfully reaching the goal set. The resulting path $\mathcal{P}$ forms a continuous, collision-free trajectory in the joint space that fulfills the prescribed task $\mathbf{x}(t)$. An animation of the task execution is available at `https://imgur.com/DV11GZ5`.

4.2 Performance Analysis

To assess the performance of MultiFM-RRT, we performed a series of experiments varying the maximum number of iterations i_{max}. The objective was to evaluate how i_{max} affects both solution quality and computational effort. All experiments were run on a workstation equipped with an AMD Ryzen 7 5700X3D CPU and 32 GB RAM, using a Python implementation based on NumPy. The parameter i_{max} was swept from 250 to 3500 in steps of 250, yielding 14 distinct settings. For each value, results were averaged over 50 independent runs. Table 2 reports the mean runtime, path cost, and success rate for each configuration.

The results indicate that increasing i_{max} leads to higher average runtimes, as anticipated. Notably, the cost of the solution path $\mathcal{P}$ quickly converges to a value near 3.6, demonstrating that high-quality solutions are attainable even with moderate iteration

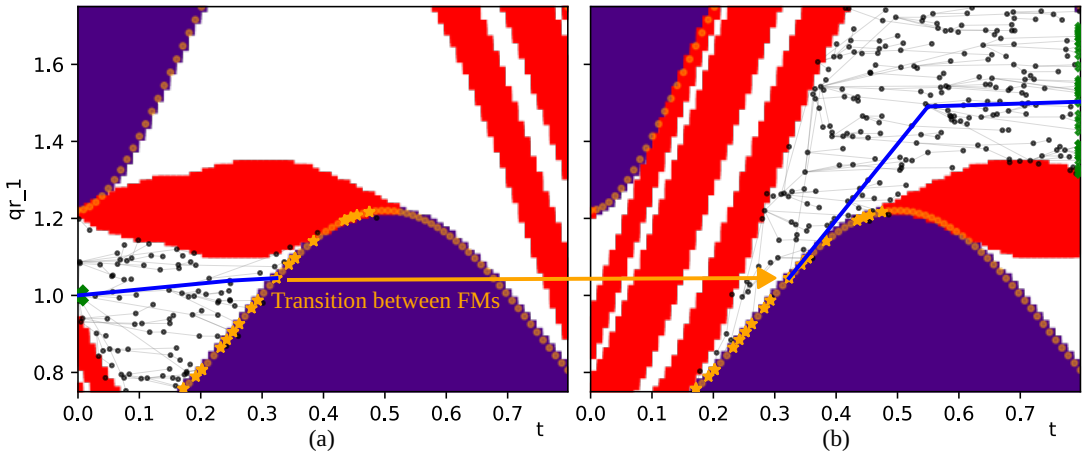


Figure 4: Feasibility maps and MultiFM-RRT tree for the Stewart platform executing the task trajectory. (a) FM for the first extended aspect ($a = 1$). (b) FM for the second extended aspect ($a = 2$).

Table 2: Performance results of MultiFM-RRT for the Stewart platform.

i_{max}	Runtime (s)	Cost of $\mathcal{P}$	Success rate
250	0.45	4.28	0.76
500	0.85	4.13	0.83
750	1.37	4.02	0.84
1000	1.86	4.08	0.95
1250	2.37	3.91	0.94
1500	2.81	3.82	0.98
1750	3.33	3.83	0.98
2000	**3.85**	**3.75**	**0.99**
2250	4.37	3.75	1.00
2500	4.95	3.63	1.00
2750	5.51	3.66	1.00
3000	5.90	3.62	1.00
3250	6.40	3.60	1.00
3500	6.97	3.59	1.00

counts. The success rate is defined as the proportion of trials in which the algorithm finds a continuous path that reaches the goal time t_{goal}, thereby completing the prescribed task trajectory.

Figure 5 shows the relationship between runtime and cost for the different tested values of i_{max}. Fitting a curve to the data reveals a clear exponential trend: as the cost decreases, runtime increases, and vice versa. Notably, the trade-off plateaus beyond a certain point, indicating diminishing returns for further increases in i_{max}. Based on this analysis, we selected $i_{max} = 2000$ as the optimal balance between runtime and solution quality for this example. This value was used in the experiment shown in Figures 3 and 4, and its performance is highlighted in Table 2.

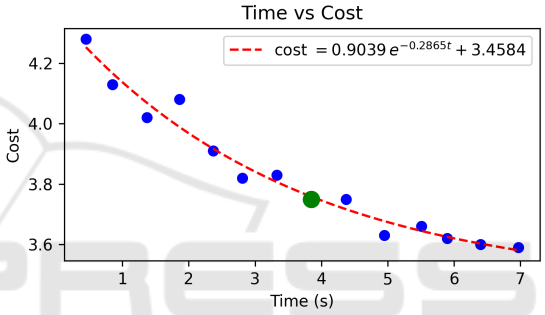


Figure 5: Relationship between runtime and cost of the solution path for different i_{max} values.

5 CONCLUSIONS

This paper presented MultiFM-RRT, a novel algorithm for redundancy resolution in kinematically redundant manipulators based on the exploration of multiple feasibility maps (FMs). By incorporating singularity maps into the RRT framework, MultiFM-RRT efficiently explores the space of feasible solutions, enabling transitions between different aspects and ensuring comprehensive coverage of the solution space. The algorithm computes FMs online, avoiding the computational burden of precomputation, and uses probabilistic sampling to guide exploration toward both goal and singularity sets.

The proposed approach was demonstrated on a Stewart platform, where MultiFM-RRT successfully generated feasible, collision-free trajectories that satisfied all task and kinematic constraints. Experimental results showed that the algorithm achieves high success rates and quickly converges to high-quality solutions, even with moderate iteration counts. The ability to autonomously transition between FMs via

SMs proved essential for solving tasks with complex constraints and multiple aspects.

Future work will focus on extending the algorithm to more complex robotic systems, incorporating additional constraints such as dynamic limits. Experiments with real robots will also be conducted to validate the approach in practical scenarios.

ACKNOWLEDGEMENTS

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.

REFERENCES

Albu-Schäffer, A. and Sachtler, A. (2023). Redundancy resolution at position level. *IEEE Transactions on Robotics*.

Borrel, P. and Liégeois, A. (1986). A study of multiple manipulator inverse kinematic solutions with applications to trajectory planning and workspace determination. In *Proceedings. 1986 ieee international conference on robotics and automation*, volume 3, pages 1180–1185. IEEE.

Burdick, J. W. (1989). 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*, pages 25–34. Springer.

Fabregat-Jaén, M., Peidró, A., Colombo, M., Rocco, P., and 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.

Fabregat-Jaen, M., Peidro, A., Gil, A., Valiente, D., and Reinoso, O. (2023). Exploring feasibility maps for trajectory planning of redundant manipulators using rrt. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE.

Fabregat-Jaén, M., Peidró, A., González-Amorós, E., Flores, M., and Reinoso, O. (2024). Augmented feasibility maps: A simultaneous approach to redundancy resolution and path planning.

Ferrentino, E. and Chiacchio, P. (2020). On the optimal resolution of inverse kinematics for redundant manipulators using a topological analysis. *Journal of Mechanisms and Robotics*, 12(3):031002.

Kazemipour, A., Khatib, M., Al Khudir, K., Gaz, C., and De Luca, A. (2022). Kinematic control of redundant robots with online handling of variable generalized hard constraints. *IEEE Robotics and Automation Letters*, 7(4):9279–9286.

Pámanes G, J. A., Wenger, P., and Zapata D, J. L. (2002). Motion planning of redundant manipulators for specified trajectory tasks. *Advances in Robot Kinematics: Theory and Applications*, pages 203–212.

Peidro, A. and Haug, E. J. (2023). Obstacle avoidance in operational configuration space kinematic control of redundant serial manipulators. *Machines*, 12(1):10.

Peidro, A., Reinoso, O., Gil, A., Marín, J. M., and 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.

Reveles, D., Wenger, P., et al. (2016). Trajectory planning of kinematically redundant parallel manipulators by using multiple working modes. *Mechanism and Machine Theory*, 98:216–230.

Wenger, P., Chedmail, P., and Reynier, F. (1993). A global analysis of following trajectories by redundant manipulators in the presence of obstacles. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 901–906. IEEE.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on man-machine systems*, 10(2):47–53.

Zanchettin, A. M. and Rocco, P. (2017). Motion planning for robotic manipulators using robust constrained control. *Control Engineering Practice*, 59:127–136.