Topological and spatial analysis of self-motion manifolds for global redundancy resolution in kinematically redundant robots

Marc Fabregat-Jaén[a], Adrián Peidró[a], Matteo Colombo[b], Paolo Rocco[b], Óscar Reinoso[a,c]

[a] *Instituto de Investigación en Ingeniería de Elche, Universidad Miguel Hernández, Avda. de la Universidad s/n, Elche, 03202, Alicante, España*
[b] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo Da Vinci 32, Milano, 20133, Lombardia, Italia*
[c] *ValgrAI: Valencian Graduate School and Research Network of Artificial Intelligence, Camí de Vera s/n, Edificio 3Q, Valencia, 46022, Valencia, España*

Abstract

This paper introduces a novel framework for global redundancy resolution in kinematically redundant robots, which have more degrees of freedom than the dimensions required to complete their task. The method is based on the concept of self-motion manifolds (SMMs), which are subsets of the joint space where the robot can move without affecting the task. Given a task trajectory, a sequence of SMMs is generated by building a graph where each node represents a c-bundle, which are sets of SMMs that share the same topology. The graph is then explored to establish feasible paths, from which preliminary joint trajectories are derived. The joint trajectories undergo an iterative optimization process that moves each joint trajectory point along the SMM of the associated task instant. The method is capable of handling kinematic constraints, such as joint limits and collisions, and it is designed to be adaptable to the kinematic complexity of the robot, real-time requirements, or optimality. The effectiveness and global optimality of the method in solving redundancy is validated through simulations with different robots and degrees of redundancy.

Keywords: Self-motion manifolds, Global redundancy resolution, Redundant manipulators, Motion planning, Obstacle avoidance

1. Introduction

Kinematic redundancy arises when the degrees of freedom (DoFs) of a robot outnumber the dimensions of the task space. This phenomenon leads to an infinite number of potential solutions to the inverse kinematics problem (IKP) for a given task $\mathbf{x}$. The process of choosing solutions from this infinite set is known as *redundancy resolution*.

When a redundant robot is tasked with executing a prescribed task trajectory $\mathbf{x}(t)$, a continuous joint-space time history $\mathbf{q}(t)$ that resolves how to use the extra DoFs (i.e., redundancy resolution) must be found. The joint-space trajectory $\mathbf{q}(t)$ must not only satisfy task constraints, like the desired end-effector trajectory, but also employ the redundant DoFs to satisfy other kinematic constraints, such as joint limits [1] or collision avoidance [2].

The redundancy resolution problem has been extensively studied in the literature, and a broad collection of methods have been proposed to address it. Depending on the approach used and the scope of the method, these can be categorized in a spectrum that ranges from *local* to *global* redundancy resolution methods, as depicted in Fig. 1.

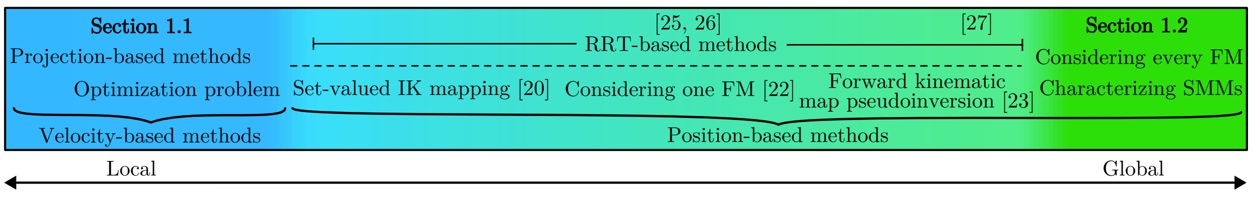

Figure 1: Spectrum of redundancy resolution methods depending on the scope.

1.1. Local redundancy resolution

Redundancy resolution is traditionally approached by means of the velocity differential equation $\dot{\mathbf{x}} = \mathbf{J}\,\dot{\mathbf{q}}$, where $\mathbf{J}(\mathbf{q})$ is the configuration-dependent non-square Jacobian matrix that maps joint velocities $\dot{\mathbf{q}}$ to task velocities $\dot{\mathbf{x}}$. *Velocity-based* methods are local techniques that only consider the vicinity of the current configuration $\mathbf{q}$, given that the Jacobian matrix $\mathbf{J}$ depends on $\mathbf{q}$. Depending on how such Jacobian matrix is used, local redundancy resolution methods can be further classified into two categories: *projection-based* and *optimization-based* methods.

Projection-based methods aim to resolve redundancy by pseudoinverting the Jacobian matrix: $\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\,\dot{\mathbf{x}}$ [3], where $(\cdot)^{\dagger}$ denotes the Moore-Penrose pseudoinverse, plus an additional null-space projection term that accomplishes secondary objectives without affecting the main task [4]. Other works have built upon this idea. For example, the authors of [5] proposed a framework for incorporating multiple hierarchically ordered tasks using the null-space projection. Flacco et al. [6] extended this framework into the SNS method, which allows for the simultaneous resolution of multiple tasks by progressively utilizing the residual DoFs that remain from higher-priority tasks.

On the other hand, optimization-based methods formulate the redundancy resolution as an optimization problem, where the objective function is typically a performance index that quantifies the quality of the solution. Rather than pseudoinverting the Jacobian, these methods employ the velocity differential equation $\dot{\mathbf{x}} = \mathbf{J}\,\dot{\mathbf{q}}$ as an equality constraint in the optimization problem. This way, typical problems that arise from pseudoinverting the Jacobian, such as singularities, are avoided. Optimization problems can be solved using a variety of techniques, such as sequential quadratic programming [7], model predictive control [8], or nonlinear versions of those [9], among others. Some works have tried to address the locality of these methods by adding a random component to the optimization process [10, 11], but the problems we will discuss next are still present.

Despite the computational efficiency and flexibility of local redundancy resolution techniques, they present certain challenges. The authors of [12] explored these issues for projection-based approaches, some of which may be extended to all local redundancy resolution methods. Given their dependence on the pseudoinversion of the Jacobian matrix, projection-based techniques may produce non-cyclic solutions for the same periodic task trajectory. In industrial settings, where repeatability is crucial, such unpredictability may be considered unacceptable, as it is expected that a closed workspace cycle should return the robot to its original configuration. Another issue is that local methods may not always find a solution, even when one exists. This is due to the fact that the exploration is led by the Jacobian matrix, whose lack of a global scope may lead the algorithm to empty solution sets, due to kinematic constraints or singularities, which materialize as external or internal barriers of the workspace of the robot [13]. Finally, local methods may find a solution that is locally optimal with respect to a given cost function, but not necessarily globally optimal. Once again, this is due to the fact that the scope of these techniques is limited to the proximity of the current configuration, which may prevent them from exploring the entire solution space, missing potentially better solutions.

1.2. Global redundancy resolution

On the other end of the spectrum, global redundancy resolution methods are *position-based* approaches that address the issues of local techniques by characterizing the entire solution space. We consider that the global scope of a method is determined by its ability to identify all possible solutions to the IKP. This is typically achieved by calculating the *self-motion manifolds* (SMMs) of the task, which are the subsets of the joint space where the robot can move without affecting the values of the task variables [14], and include all possible solutions to the IKP. Further details on SMMs and their generation will be provided in Section 3.

For instance, Lück [15] proposed building a graph of sets of SMMs for its later exploration in order to extract a joint-space trajectory. [16] presented an RRT-based path planning approach that employs SMMs to determine the continuity between sampled points, and to derive a joint trajectory. The authors of [12] introduced a formal strategy for redundancy resolution at the position level, employing foliations that are orthogonal to the SMMs to establish a global alternative to the null-space projection term of projection-based local methods.

Another approach to global redundancy resolution is the use of *feasibility maps* (FMs) [17]. In contrast with SMMs, which are pointwise representations of the solution space (i.e., SMMs are solutions for a given task-space point), FMs are trajectory-wise representations (i.e., FMs are solutions for a given task-space trajectory). FMs are based on task-space augmentation: for a given task trajectory $\mathbf{x}(t)$, a redundant-coordinates vector $\mathbf{r}$ is added to the task coordinates. FMs are the set of points $(t, \mathbf{r})$ that satisfy the kinematic equations of the robot and the imposed constraints. For every point $(t, \mathbf{r})$, the joint configuration $\mathbf{q}$ that satisfies the desired task $\mathbf{x}(t)$ is solved and checked for feasibility. Every solution $\mathbf{q}$ will lie on an *extended aspect* [18], and there will exist as many FMs as extended aspects, as each one is associated with a possible solution to the IKP. Given that every FM is considered, the method characterizes the entire solution space, thus qualifying as a global redundancy resolution tool. For example, Ferrentino and Chiacchio [19] run an exhaustive dynamic program that explores every FM to find the optimal solution.

1.3. In-between approaches

There exist a range of position-based approaches to redundancy resolution that, while not strictly global because they do not exhaustively scan all possible solutions to the IKP for a given task value, have a broader scope than local velocity-based techniques.

Haug [20] presents a set-valued inverse kinematic mapping of redundant manipulators at configuration (or position) level, where the n joint coordinates $\mathbf{q}$ are parameterized in terms of the m task coordinates $\mathbf{x}$ and $n - m$ self-motion coordinates $\mathbf{v}$ that account for redundant degrees of freedom. This mapping is built around a given configuration $\overline{\mathbf{q}}$ of the manipulator by extracting a basis $\mathbf{V}$ of the nullspace of $\mathbf{J}(\overline{\mathbf{q}})$. While in general this mapping is valid only around $\overline{\mathbf{q}}$, its validity can be extended as the trajectory of the manipulator progresses by updating $\mathbf{V}$ whenever the parameterization becomes ill-conditioned. Moreover, depending on the manipulator, the range of validity of such mapping may even become very broad without requiring reparameterizations at all [20]. This mapping elegantly solves deficiencies of traditional generalized-inverse velocity-based approaches such as lack of cyclicity [20], and performs much better than these approaches when dealing with constraints that are better defined at configuration than at velocity level, such as avoidance of obstacles [21].

Another example is the work [22], which proposed a motion planning algorithm that solves the redundancy by exploring one of the FMs associated with the task. The exploration is performed by an RRT-based algorithm that samples points in the FM (instead of doing it in the task or joint spaces) and builds a tree that connects the sampled points. Since only one FM is considered, the method's global scope is restricted to the extended aspect of the solution associated with that FM.

Hauser and Emmons [23] proposed a probabilistic roadmap that concurrently samples points in both the task space and the associated SMM for each task point. Then, a heuristic local three-stage search algorithm is employed to construct a pseudoinversion of the forward kinematic map. Since this pseudoinversion returns a unique $\mathbf{q}$ for each task point $\mathbf{x}$ (instead of all $\mathbf{q}$ that map to $\mathbf{x}$), this may limit finding potentially better solutions. For this reason, although we classify this near the completely global methods in the spectrum of Fig. 1, it is not classified in Section 1.2 with other completely global methods that do obtain all solutions for each task $\mathbf{x}$.

Approaches that utilize sampling-based path planning algorithms cannot be strictly categorized in a single point of the spectrum of Fig. 1, as the scope of the method depends on the exploration strategy. A popular choice is the Rapidly-exploring Random Tree (RRT) algorithm [24], upon which the authors of [25] based their solution, projecting each sampled configuration onto the constraint manifold (i.e., pairs of $(\mathbf{x}, \mathbf{q})$ that satisfy the kinematic constraints of the robot), which allowed for the direct incorporation of additional constraints into the exploration process. Jaillet and Porta [26] proposed a similar approach, where the constraint manifold is described by an atlas, which is iteratively constructed and coordinated, simultaneously with the expansion of the RRT. The AtlasRRT* method, introduced in [27], extends upon the previous work and its asymptotic optimality potentially qualifies it as global redundancy resolution, as it will eventually identify the necessary parts of the constraint manifold. The large amount of existing RRT-based algorithms for redundancy resolution makes it difficult to categorize them all, as they cover a wide range of the spectrum.

1.4. Contributions

In this work, we propose a novel framework for global redundancy resolution. The contributions of this paper can be summarized as follows:

- Similarly to [15], we build a graph of c-bundles (i.e., collections of close SMMs sharing the same topology), but instead of building this graph for the entire task space, we focus the computational effort on creating the graph spanned by a desired task trajectory $\mathbf{x}(t)$, which is a more practical approach for real-time applications, where one only needs to solve a concrete trajectory rather than the complete configuration space of the robot. The application of the graph is also unique. While Lück [15] uses the information of the graph to discretize the entire joint and task spaces, we use it to establish the set of feasible graph paths that achieve completion of the desired task trajectory. From each graph path, which we will later define as *c-bundle chains*, we extract a preliminary raw joint trajectory. Finally, these raw joint trajectories are optimized using a novel optimization approach that, for each time instant t, keeps the joint configuration of the robot on the SMM corresponding to the task point $\mathbf{x}(t)$, while respecting kinematic constraints such as joint limits and obstacle avoidance.

- We present a variant of the sampling-based method for SMM generation presented in [13]. The performance of the method is improved by vectorizing the sampling of the SMMs, which allows for a significant reduction in the computational cost, and the clustering and matching operations are re-defined. In addition, we extend the method in order to "glue" SMMs along the time dimension to generate the so-called *self-motion domain* (SMD) of the task trajectory, from which the c-bundle graph is inferred.

- We employ the SMD to solve the redundancy resolution problem, and obtain a joint trajectory that is globally optimal in terms of some performance index, while respecting kinematic constraints. The global optimality of the method is discussed and validated through experimental results.
- The presented framework is highly flexible, as it allows the user to select from various computational alternatives at each stage of the algorithm. Among the three stages of the framework, users can select from various alternatives for computation of the SMD, raw trajectory generation, and the optimization stages. This adaptability can address a variety of user-specific needs, whether they are related to generalization (adaptability to complex kinematic chains), redundancy dimension, real-time requirements, or global optimality.
- Our SMM computation method is significantly faster than those presented in the literature. In fact, the method is capable of generating the entire SMD (collection of SMMs discretized along the task trajectory) in a fraction of the time required by other methods to generate SMMs at a single task point.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of the state of the art on global redundancy resolution. Section 3 provides an in-depth explanation of SMMs and their generation, as well as some concepts that are relevant to the proposed method, which is presented in Section 4. In Section 5, we validate the method through simulations. Section 6 discusses the method's optimality, efficiency and scalability. Finally, Section 7 concludes the paper and suggests future directions.

2. Related work

Global redundancy resolution has been addressed by a variety of methods in the literature. We consider that the global scope of a method is determined by its ability to identify all possible solutions to the IKP (e.g., the SMMs for a given task $\mathbf{x}$). In this section, we discuss some global redundancy resolution methods.

The recent work in [12] presents a formal strategy to redundancy resolution at position level, employing foliations that are orthogonal to the SMMs. They propose a coordinate-growing approach to generate the foliations, but it is only applicable to one-dimensional tasks, and is described as rather expensive in execution time. For tasks of higher dimensions, which are the norm in practice, the authors employ neural networks to approximate the foliations. However, it is not without its drawbacks, such as the loss of strict orthogonality and the large training time required for the neural networks.

In [15], Lück proposed another approach. He builds a graph where each node represents a c-bundle, and each edge represents a feasible transition between them. Each element of the graph (nodes and edges) corresponds to a significant concept related to SMMs, that will be introduced in Section 3. The information contained in the graph is then discretized, both in the joint and task spaces, resulting in a connectivity graph. The latter graph is finally explored following the A* algorithm, obtaining the globally optimal solution on the discretized space. Specifically, the algorithm outputs a discrete solution in the joint space (the centroid of each discretized cell). While continuity can be achieved by connecting the nodes, it will result in a non-smooth trajectory with high acceleration peaks.

The work of Zhou et al. [16] also presents a method that concurrently addresses path planning and redundancy resolution. They propose a two-stage approach. First, an RRT is built in the task space, determining continuity between sampled points based on the length of the SMMs at each task point. Consequently, the construction of the RRT may necessitate a sufficiently small maximum step size. In the second stage, the authors connect the sampled SMMs in the joint space and derive the joint trajectory based on an exhaustive search that evaluates an averaged performance index (e.g., reciprocal condition number) along each path (i.e., sequence of SMMs).

In the context of FMs, the work of Ferrentino and Chiacchio [19] proposes a global redundancy resolution method. The authors first generate the set of FMs (one per extended aspect) associated with the prescribed task. Then, they run an exhaustive dynamic program that explores every FM to find the optimal solution. The method is computationally expensive, as it requires the evaluation of numerous points within each FM and consider potential combination among FMs. Additionally, the previous generation of multiple FMs is necessary, which is also time-consuming.

These methods have been successful in solving the global redundancy resolution problem, but they present some limitations. Namely, they offer execution times for the global redundancy resolution that are too expensive. In addition, except for the work of Albu-Schäffer and Sachtler [12], none of the methods have been tested with degrees of redundancy higher than one. With our method, we aim to address these limitations, providing a global redundancy resolution method that is capable of globally solving higher degrees of redundancy in a reasonable timeframe.

3. Self-motion manifolds

Let Eq. (1) be the constraint that defines the relationship between 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 is defined as finding a joint configuration $\mathbf{q}$ that satisfies $\mathbf{F}(\mathbf{x}, \mathbf{q}) = \mathbf{0}$ for a given task point $\mathbf{x}$. For redundant manipulators, where $n > m$, the IKP yields an infinite number of solutions $\mathbf{q}$ that satisfy the equation. Generically, these solutions lie on a finite number of r-dimensional disjoint manifolds in the joint space. Burdick [14] coined the term *self-motion manifolds* (SMMs) to refer to these manifolds, as moving the joint configuration along them does not affect the task-space variables. These representations are particularly useful for redundancy resolution, as they provide a global view of the infinite solutions to the IKP.

Consider a hypothetical 2-DoF $\mathbf{q} = [q_1, q_2]^{\mathrm{T}}$ redundant manipulator executing a one-dimensional task trajectory $\mathbf{x}(t) = [x]$, represented in Fig. 2(a). The resulting degree of redundancy is $r = 1$, which produces one-dimensional SMMs (i.e., curves) in the joint space for a fixed $\mathbf{x}$. A hypothetical example of the resulting SMMs for the task value $\mathbf{x}(t = f)$ is shown in Fig. 2(b). The plotted curves (SMMs $\mathbf{M}_f^1$ and $\mathbf{M}_f^2$) in the joint space are the set of joint configurations $\mathbf{q}$ that satisfy Eq. (1) for that specific task point $\mathbf{x}(f)$. The presence of two disjoint SMMs at that task point, denoted as $\mathbf{M}_f^1$ and $\mathbf{M}_f^2$, implies that the robot will not be able to transition from a configuration pertaining to one of the SMMs to a configuration within the other without violating Eq. (1).

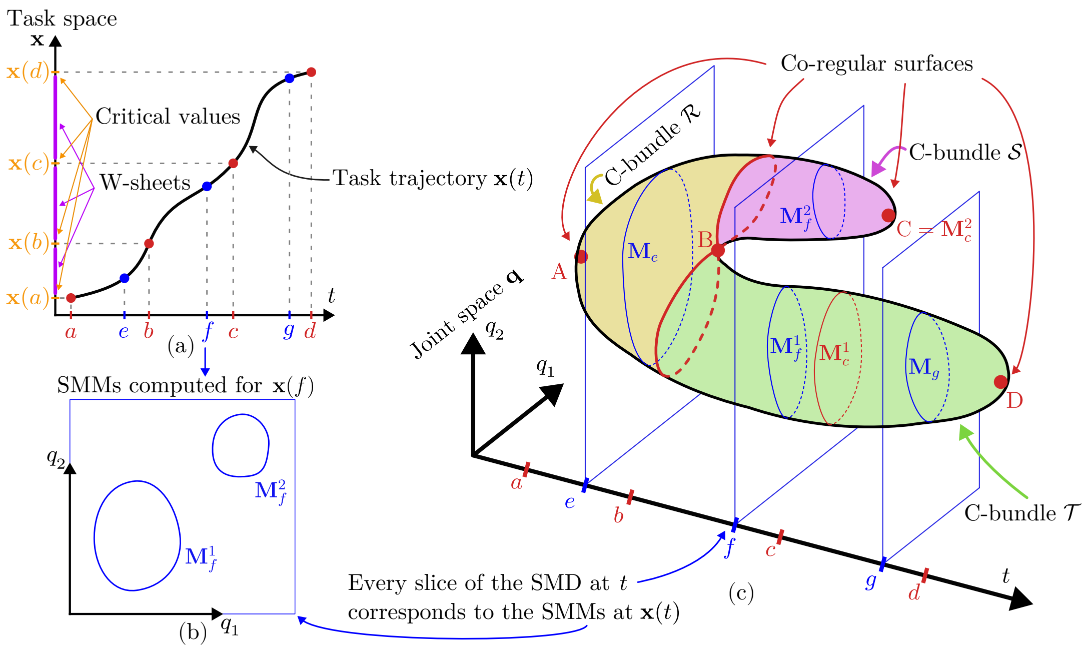

Figure 2: (a) Task trajectory $\mathbf{x}(t)$. (b) One-dimensional SMMs (curves) at task point $\mathbf{x}(f)$. (c) SMD for task trajectory $\mathbf{x}(t)$.

Every SMM corresponds to the infinite joint configurations $\mathbf{q}$ that satisfy Eq. (1) for a fixed task point $\mathbf{x}$. If a task trajectory $\mathbf{x}(t)$ is considered, where t is an arc-length parameter (hereafter referred to as time for simplicity), the solutions $\mathbf{q}$ that satisfy $\mathbf{F}(\mathbf{x}(t), \mathbf{q}) = \mathbf{0}$ form a manifold of $(r + 1)$ dimensions, where every slice of the manifold at a specific time t corresponds to the SMM for that task point $\mathbf{x}(t)$ (see Fig. 2(b) and the slice at $t = f$ in Fig. 2(c)). Such $(r + 1)$-dimensional manifold, named *self-motion domain* (SMD) in [28], is exemplified in Fig. 2(c) for a representative task trajectory $\mathbf{x}(t)$. The SMMs of three different time values $t = \{e, f, g\}$ are plotted in blue, which result from intersecting the SMD with the planes orthogonal to the axis t at each instant.

Some analysis can be performed based on the observed SMMs. For $t = e$ and $t = g$, there are single one-dimensional manifolds, denoted as $\mathbf{M}_e$ and $\mathbf{M}_g$, respectively. All joint configurations within these manifolds are mapped to $\mathbf{x}(e)$ and $\mathbf{x}(g)$, respectively, when projected to the task space. Consequently, $\mathbf{M}_e$ and $\mathbf{M}_g$ are the *preimages* or solutions to the IKP for task points $\mathbf{x}(e)$ and $\mathbf{x}(g)$, respectively. On the other hand, $\mathbf{M}_f^1$ and $\mathbf{M}_f^2$, which are the preimage manifolds of $\mathbf{x}(f)$ (intersection of the SMD with the plane $t = f$), are two disjoint one-dimensional SMMs. Therefore, it can be inferred that some transformation in the SMMs has occurred as the task trajectory progresses, since the number of disjoint SMMs has changed from one at $t = e$, to two at $t = f$, and back to one at $t = g$.

To better comprehend these transformations, it is beneficial to introduce some paramount concepts presented in [14]. Let $\mathbf{J}$ be the Jacobian matrix that maps joint velocities $\dot{\mathbf{q}}$ to task velocities $\dot{\mathbf{x}}$ (i.e., $\dot{\mathbf{x}} = \mathbf{J}\,\dot{\mathbf{q}}$). Points of the SMD where $\mathbf{J}$ is not full rank are *singularities*. In the SMD under consideration, the joint configurations that are singularities are represented by the red points $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$ in Fig. 2(c). The projections of these singular points onto the time axis t correspond to the values $\{a, b, c, d\}$. The corresponding task values of each singular point (in this case $\{\mathbf{x}(a), \mathbf{x}(b), \mathbf{x}(c), \mathbf{x}(d)\}$ in Fig. 2(a)) are known as *critical values*. Fig. 2(a) also depicts *w-sheets* in purple, which are continuous regions of the task space between critical values.

From critical values, two more important concepts can be derived that play a significant role in our presented redundancy resolution method. Preimages of critical values (i.e., the set of $\mathbf{q}$ that satisfy Eq. (1) for a critical value $\mathbf{x}$) are r-dimensional sets, where a *co-regular surface* exists. Indeed, co-regular surfaces can be interpreted as sets where singularities coexist with regular points, thus inspiring the term 'co-regular' (see co-regular surface at $t = b$ in Fig. 2(c)). Note that the term 'surface' may be misleading, as they can be of any dimension (e.g., curves ($r = 1$) in this example). Co-regular surfaces may also correspond to a single singular point, instead of coexisting with regular points, as is the case for singularities A, C and D. A paramount concept of co-regular surfaces is that they divide the SMD into disjoint components, referred to as *c-bundles*, denoted as $\mathcal{R}$, $\mathcal{S}$ and $\mathcal{T}$ in Fig. 2(c). A key characteristic of c-bundles is that they are sets of stacked SMMs that share the same topology, which means that no topological transformations in the SMMs occur within a c-bundle. Therefore, it can be assured that the topological transformations of the SMMs are limited to the boundaries of the c-bundles, which are the co-regular surfaces.

3.1. Transformations of self-motion manifolds

Studying the transformations of SMMs is crucial for our proposed redundancy resolution method. The SMD depicted in Fig. 2(c) is composed of three c-bundles: $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$. Several types of transformations can be extracted when studying how the SMMs within the c-bundles evolve.

Let us analyse the transformations that occur in the SMMs when positively traversing the time axis t. The first critical value encountered is $\mathbf{x}(a)$, whose preimage is the singular point A. For values $t < a$, there are no feasible joint configurations that map to $\mathbf{x}(t)$ (i.e., no SMMs exist). At $t = a$, the *creation* of an SMM occurs, which is stated by the appearance of the singularity. This singularity marks the beginning of the c-bundle $\mathcal{R}$. Between instants $t = a$ and $t = b$, the SMMs within the c-bundle $\mathcal{R}$ remain topologically unchanged.

When the critical value $\mathbf{x}(b)$ is reached, intersecting the SMD with the plane $t = b$ returns a co-regular surface. When positively crossing (in the direction of t) this co-regular surface, the single SMM that existed in c-bundle $\mathcal{R}$ splits into two disjoint SMMs, which also produces the splitting of the c-bundle $\mathcal{R}$ into two c-bundles: $\mathcal{S}$ and $\mathcal{T}$. Note that the *splitting* operation is of paramount importance for redundancy resolution: if the joint configuration ends up lying in the c-bundle $\mathcal{S}$ when traversing the co-regular surface at $t = b$, the robot will not be able to reach the time instant $t = d$ without violating Eq. (1) (i.e., the task will not be feasible). This phenomenon occurs as the SMMs within c-bundle $\mathcal{S}$ will progressively shrink as they approach the singularity C, which will eventually lead to the *vanishing* of the SMM when crossing $t = c$, thus preventing the robot from reaching the time instant $t = d$. On the other hand, the task would be feasible if the joint configuration was planned to lie in the c-bundle $\mathcal{T}$, as it would be able reach the time instant $t = d$, where the SMM shrinks to the singular point D and then vanishes.

An additional operation can be identified when the SMD is traversed in the opposite direction. When crossing the co-regular surface at $t = b$ in the negative direction of t, the two disjoint SMMs merge into a single SMM. Although the *merging* operation is not as relevant for redundancy resolution as the splitting operation, as it does not affect the feasibility of the task, it is still important to consider it, as it dictates the connectivity between different c-bundles, which will be crucial for the spatial analysis of the c-bundles that will be presented later in the paper.

3.2. Computation of self-motion manifolds

In this section, we provide an overview of the methods that have been proposed to compute SMMs.

3.2.1. Continuation methods

Along with the introduction of the SMM concept, Burdick [14] proposed the basis for a method to compute them. The method, which we will refer to as the *continuation method*, is based on the continuation in the Jacobian's null-space, which is tangent to the SMM, as is illustrated in Fig. 3(b), where $\mathbf{NS}$ is a null-space basis.

Starting from an initial joint configuration $\mathbf{q}_0$, the method iteratively computes the SMM by displacing the joint configuration a small distance ρ along the null-space of the Jacobian. In Fig. 3(a), a one-dimensional SMM embedded in a 2-DoF joint space is exemplified, where the theoretical SMM is depicted in blue, and the computed points are represented by the black dots. Fig. 3(b) provides a detailed view of the computation process to form the SMM.

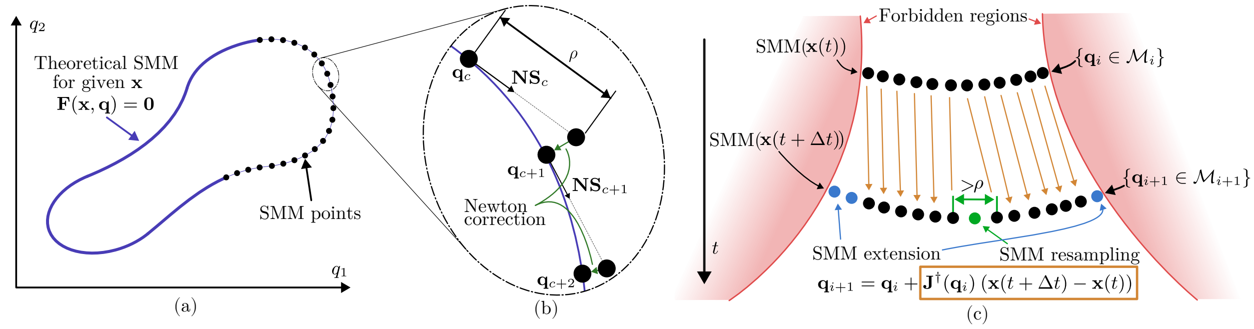
q_2
Theoretical SMM for given $\mathbf{x}$
$\mathbf{F}(\mathbf{x},\mathbf{q})=\mathbf{0}$
SMM points
q_1
(a)
ρ
$\mathbf{q}_c$
$\mathbf{NS}_c$
$\mathbf{q}_{c+1}$
$\mathbf{NS}_{c+1}$
Newton correction
$\mathbf{q}_{c+2}$
(b)
Forbidden regions
SMM($\mathbf{x}(t)$)
$\{\mathbf{q}_i \in \mathcal{M}_i\}$
SMM($\mathbf{x}(t+\Delta t)$)
$>\rho$
$\{\mathbf{q}_{i+1} \in \mathcal{M}_{i+1}\}$
t
SMM extension
SMM resampling
$\mathbf{q}_{i+1} = \mathbf{q}_i + \boxed{\mathbf{J}^{\dagger}(\mathbf{q}_i)\,(\mathbf{x}(t+\Delta t)-\mathbf{x}(t))}$
(c)

Figure 3: (a)One-dimensional SMM (curve). (b) Continuation method. (c) Task propagation.

In Fig. 3(b), $\mathbf{NS}_c$ corresponds to an orthonormal basis of the null-space of the Jacobian at the c-th computed point $\mathbf{q}_c$. Adjacent points are computed by displacing the joint configuration $\mathbf{q}_c$ along the null-space basis $\mathbf{NS}_c$ by a distance ρ. However, since the Jacobian's null space is a linear approximation of the SMM at each point, the computed adjacent points, which should pertain to the manifold, may not be accurate due to this approximation, especially in curved regions of the manifold, as depicted in Fig. 3(b). A correction must be applied to the computed points to ensure that they lie on the SMM. For this purpose, a Newton-based correction is employed, which is illustrated by the green arrows in Fig. 3(b). The correction is performed by iteratively solving the equation $\Delta\mathbf{x} = \mathbf{J}(\mathbf{q})\,\Delta\mathbf{q}$, where $\Delta\mathbf{x}$ is the difference between the task values of the computed point and the SMM, and $\Delta\mathbf{q}$ is the correction to be applied to the joint configuration. Note that by solving this equation via the Moore-Penrose pseudoinversion of $\mathbf{J}$, the correction $\Delta\mathbf{q}$ is applied perpendicularly to the SMM, which ensures a faster convergence.

Due to the heavy computational operations involved at each iteration, the method is computationally expensive. More importantly, its major limitation is that only the SMM to which the initial joint configuration $\mathbf{q}_0$ belongs will be computed. For task values yielding multiple disjoint SMMs (e.g., the SMMs in Fig. 2(b)), one should provide a joint configuration that acts as a seed for each SMM. Therefore, for applications where every SMM is required, the method may be impractical as it demands the generation of successive joint configurations until all SMMs are identified, which is complex due to the uncertainty of the exact number of SMMs. Yet, even with this approach, completely identifying all SMMs is not guaranteed unless the number of SMMs equals the maximum number of solutions for the IKP of a non-redundant manipulator of the same class (spherical, regional, spatial, etc.) [14].

In [29], the author proposed a generalization of the original method that allows for the computation of manifolds of higher dimensions. The method is based on the concept of an atlas, which is a collection of charts that cover the manifold. Each chart defines a local coordinate system that is used to parameterize the manifold in a neighbourhood of a point. Similar to the original method, the atlas is built by iteratively computing the neighbouring charts of a given chart until the entire manifold is covered. While it is a generalized approach for any dimension of the manifold, the method shares the same limitations as the original method. Furthermore, with higher degrees of redundancy, the dimensionality of the SMMs increases, and so does the computation time, which makes these methods not suitable for applications where the entirety of the SMD must be generated quickly.

3.2.2. Sampling methods

Another family of methods to compute SMMs are based on sampling. These methods typically involve two stages: the sampling of the joint space to obtain a point cloud that approximates the SMMs, and a clustering stage to identify the disjoint SMMs from the point cloud.

DeMers and Kreutz-Delgado [30] proposed generating the SMMs by randomly sampling the entire joint space, and selecting the joint configurations that are mapped (by forward kinematics) sufficiently close to the task-space value of interest. Then, all joint samples contained in the same task region are clustered by solving a Minimum Spanning Tree to identify the disjoint SMMs. The method is computationally demanding, as it requires the dense sampling of the entire joint space, which is not efficient, since SMMs are lower-dimensional manifolds embedded in the joint space, meaning that the vast majority of the sampled points will not belong to the SMM of interest.

Following a similar approach, Peidró et al. [13] presented a more efficient method. Instead of sampling the entire joint space, the authors generate the point cloud by sweeping every redundant joint that exceeds the number of task dimension and solving the IKP to obtain the remaining joint variables. This produces a densely sampled point cloud, whose points actually pertain to the SMMs, and not the entire joint space. For the clustering stage, the authors propose a recursive method that employs k-d trees to efficiently identify the disjoint SMMs. The specific challenge of the method is the need to express $\mathbf{F}(\mathbf{x}, \mathbf{q})$ in terms of every combination of r redundant variables, which is not always possible (i.e., solving $\mathbf{q}$ from $\mathbf{F} = \mathbf{0}$ is not always possible). The method presents several advantages, including computational efficiency, as it directly samples the SMMs rather than the entire joint space; and the guaranteed identification of all manifolds. It outperforms continuation methods in terms of speed, making it suitable for applications where SMMs must be generated in a short time, at least up to a certain degree of redundancy.

The detailed explanation of this family of methods will be left for Section 4.1, where we will present a method to compute SMDs that is based on the sampling approach and is closely related to the method proposed in [13].

3.2.3. Other methods

Other methods to compute SMMs that do not fit into the two most common categories have been presented in the literature. We will briefly discuss these methods, without going into detail, as they are not directly related to our proposed approach.

In [31], the authors propose a novel method to compute SMMs based on the concept of cellular automata. However, the presented computation times displayed are prohibitive for real-time applications, even when computing SMMs at a single task point, let alone the entire SMD.

Zhou et al. [16] presented a continuation approach to compute SMMs by implementing the Artificial Bee Colony Algorithm. This way, the authors are able to compute the SMMs without the need for the Jacobian matrix. Nevertheless, similarly to the continuation methods, it requires a seed joint configuration for each SMM, which is not practical for applications where the entire SMD must be generated, as the exact number of SMMs is unknown (Section 3.2.1).

Similarly, the authors of [32] proposed a method to compute SMMs based on multi-objective optimization. Finally, [28] presented a method to compute SMMs based on interval branch-and-bound. The methods are capable of successfully computing the SMMs for a given task point, but they are computationally expensive.

4. Proposed method for global redundancy resolution

As already mentioned, given a defined task trajectory $\mathbf{x}(t)$, the goal of the redundancy resolution problem is to find an optimal joint trajectory $\mathbf{q}(t)$ that satisfies $\mathbf{F}(\mathbf{x}(t), \mathbf{q}(t)) = \mathbf{0}$ in Eq. (1) and the imposed kinematic constraints. The kinematic constraints considered in this work include joint limits and the avoidance of collisions with obstacles.

We propose a three-stage novel framework for global redundancy resolution. The first stage involves the simultaneous generation of the SMD for the given task trajectory, and a graph that shows the connectivity of c-bundles along the SMD. Then, the graph is explored to establish every possible c-bundle chain (sequence of c-bundles) that satisfies the task, deriving a raw joint trajectory for each c-bundle chain. Finally, the raw joint trajectories are optimized by iteratively solving a constrained quadratic programming (QP) optimization problem that moves each discretized point along the SMM they belong to.

Furthermore, we introduce a range of variants for each stage of the algorithm, which allow users to select from various alternatives, depending on the specific requirements of their application.

4.1. SMD and graph generation

The first step of the proposed method is the generation of the SMD and graph for the given task trajectory. The SMD is computed using a sampling-based approach, which is closely related to the method proposed by Peidró et al. [13]. The steps are outlined in Algorithm 1. The graph $\mathcal{G}$ is composed of nodes that represent the c-bundles of the SMD for the task trajectory, and edges that represent the transformations that occur in the SMMs when transitioning between adjacent c-bundles.

First, the task trajectory $\mathbf{x}(t)$ is discretized into k task-space points, resulting in the sequence $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$, distributed along the trajectory with a step size Δt (i.e., $\mathbf{x}_i = \mathbf{x}(i\,\Delta t)$, with $\Delta t = \frac{t_f}{k-1}$). The number of points k is a user-defined parameter that determines the granularity of the SMD. If k is too small, the SMD may not be accurately represented, while if k is too large, the computational cost will increase. We recommend selecting a value of k that results in time steps Δt of approximately 0.05 seconds, which is an acceptable refresh rate for real-time applications.

Similarly to the methods for computing SMMs discussed in Section 3.2.2, the manifolds are sampled and subsequently clustered to identify the set of disjoint SMMs for each $\mathbf{x}_i$. In the process of generating the SMD, we loop over each task-space point $\mathbf{x}_i$ in $\mathcal{X}$, computing the SMMs at each task value.

Algorithm 1 SMD generation

```
Initialize: $\mathcal{X} \leftarrow$ Ordered set $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k\}$,    $\mathcal{G} \leftarrow$ Empty graph
for $\mathbf{x}_i \in \mathcal{X}$ do
    $\mathcal{P} \leftarrow$ SampleSMMs($\mathbf{x}_i$)    ▷ Sample a point cloud that approximates the SMMs at $\mathbf{x}_i$ (Fig. 4(a))
    $\mathcal{M} \leftarrow$ ClusterSMMs($\mathcal{P}$)    ▷ Cluster the point cloud into a set of disjoint SMMs (Fig. 4(b))
    $\mathcal{G} \leftarrow$ MatchSMMs($\mathcal{M}, \mathcal{G}$)    ▷ Match (or glue) the set of disjoint SMMs to the SMD (Figs. 4(c-d))
return $\mathcal{G}$
```

The SampleSMMs function generates a densely populated point cloud $\mathcal{P}$ that approximates the SMMs at task point $\mathbf{x}_i$. The procedure is similar to [13]. An outline of the steps is provided in Algorithm 2, but the reader is referred to the original work for further explanation.

Algorithm 2 Sampling of the SMMs at $\mathbf{x}_i$

```
1: procedure SampleSMMs($\mathbf{x}_i$)
2:     $\mathcal{P} \leftarrow \emptyset$
3:     for every different combination of $r$ redundant joints $\overbrace{(q_i, q_j, \cdots, q_k)}^{r \text{ coordinates}} \mid i, j, \cdots, k \in \{1, \cdots, n\}$ do
4:         Let $\mathcal{Q}_v = [q_{v_{\min}}, q_{v_{\max}}]$ be the valid interval of joint $q_v$ between its joint limits ($\forall\, v \in \{i, j, \cdots, k\}$)
5:         Let $\mathbf{Q}_v = \{q_{v_{\min}}, q_{v_{\min}} + \Delta q_v, \cdots, q_{v_{\max}}\}$ be a discretization of $\mathcal{Q}_v$ with step $\Delta q_v$
6:         The Cartesian product $\mathbf{Q}_{red} = \mathbf{Q}_i \times \mathbf{Q}_j \times \cdots \times \mathbf{Q}_k$ is an $r$-dimensional grid of joint coordinates
7:         for every node $\mathbf{q}_{red}$ of the $r$-dimensional grid $\mathbf{Q}_{red}$ do
8:             Substitute $\mathbf{q}_{red} = [q_i, q_j, \cdots, q_k]^{\mathrm{T}}$ and $\mathbf{x} = \mathbf{x}_i$ into Eq. (1) and solve $\mathbf{q}$
9:             if $\mathbf{q}$ is a feasible joint configuration then    ▷ e.g., joint limits, collision avoidance
10:                Add $\mathbf{q}$ to the point cloud $\mathcal{P}$
11:    return $\mathcal{P}$
```

The algorithm will store the feasible sampled points in a point cloud $\mathcal{P}$, that is initialized empty. A loop (outer loop) is then started to select every *different* combination of r joint coordinates among the n available. Note that different orderings of the same joint coordinates are not considered, i.e., for $r = 2$, the combination (q_1, q_2) is considered the same as (q_2, q_1). For every said combination, the valid interval $\mathcal{Q}_v$ of each joint coordinate is determined in line 4. The valid interval $\mathcal{Q}_v$ is defined by the joint limits of the robot, and the discretization $\mathbf{Q}_v$ is obtained in line 5 by dividing the interval $\mathcal{Q}_v$ into a set of discrete values with a step Δq_v. The Cartesian product of the discretized joint coordinates $\mathbf{Q}_{red}$ is then computed in line 6 to generate an r-dimensional grid of joint coordinates. Finally, the grid is swept in lines 7-10 (inner loop), where $\mathbf{q}$ is solved for the remaining m joint coordinates that are not in the r-dimensional grid. The m remaining joint coordinates in $\mathbf{q}$ are solved by substituting the r joint coordinates $\mathbf{q}_{red}$ into Eq. (1), along with the task point $\mathbf{x}_i$, obtaining every possible solution of $\mathbf{q}$ that maps to the task point $\mathbf{x}_i$. If the resulting joint configuration $\mathbf{q}$ is feasible (e.g., joint limits are met, and no collision is produced), it is added to the point cloud $\mathcal{P}$. Otherwise, the joint configuration is stored in a separate point cloud $\bar{\mathcal{P}}$, for its later use in Section 4.3. This procedure is computationally expensive, as it requires solving the IKP (preferably via algebraic elimination, obtaining all solutions) for every combination of r joint coordinates, producing in practice a number of outer loops (those produced by line 3) equal to the binomial coefficient $\binom{n}{r}$. Although the inner loop (line 7) is presented sequentially in both the presented algorithm and the original work [13], *vectorization* can be employed to bypass it, significantly accelerating the computation. Instead of iterating over each node in the grid $\mathbf{Q}_{red}$ and solving the IKP for each node, vectorization allows for the simultaneous solving of all nodes in the grid.

Next, back in Algorithm 1, the ClusterSMMs function clusters the point cloud $\mathcal{P}$ into a set $\mathcal{M}$ of disjoint SMMs. For this purpose, we utilize DBSCAN [33], a density-based clustering algorithm. Notably, this algorithm does not require specifying the number of clusters, aligning with our earlier discussions about only knowing the maximum number of potential disjoint SMMs, rather than the exact number (Section 3.2.1). The only parameters that must be defined are the maximum distance between two points to be considered part of the same cluster, and the minimum number of points required to form a cluster, which we set to 1. We denote the maximum distance as h, derived from the sampling process as follows: $h = \sqrt{\sum_{j=1}^{n} \Delta {q_j}^2}$, where Δq_j is the step in joint j during the discretization of its interval as indicated in line 5 of Algorithm 2. The DBSCAN algorithm returns a set of integer labels that indicate the SMM (cluster) to which each point in $\mathcal{P}$ belongs. Utilizing these labels, the point cloud is partitioned into a set of disjoint SMMs, which are stored and returned as the set $\mathcal{M}$.

Finally, the MatchSMMs function will match each disjoint SMM in $\mathcal{M}$ to the SMD generated up to that point (stored in graph $\mathcal{G}$). The matching is performed by comparing the newly generated $\mathcal{M}_t$ with the last computed set of SMMs in the SMD (i.e., SMMS in $\mathcal{M}_{t-\Delta t}$).

Before explaining the matching procedure, we have considered appropriate to exemplify the SMD and graph generation process with a simple example, in order to provide a better understanding of the method, and to introduce a couple of important concepts that will be used throughout the explanation of the last stage of the algorithm.

Consider the same robot and task trajectory $\mathbf{x}(t)$ used in Section 3 to explain SMMs (i.e., $n = 2$, $m = 1$, and $r = 1$), which produces the SMD depicted in Fig. 2(c). As we traverse the axis t in a positive direction, for each instant t between $t = a$ and $t = b$, there is a single SMM. Given that no topological transformations occur in the SMMs (as discussed in Section 3.1), the matching stage should generate the c-bundle $\mathcal{R}$, associating each subsequently generated SMM at t with the single SMM at the previous time instant $t - \Delta t$.

Upon surpassing the critical value $\mathbf{x}(b)$ (i.e., at $t = b + \Delta t$), the sampling stage generates a point cloud $\mathcal{P}$, approximating the two disjoint SMMs resulting from the splitting operation, as illustrated in Fig. 4(a). The clustering stage, depicted in Fig. 4(b), identifies the set $\mathcal{M}_{b+\Delta t} = \{\mathbf{M}^1_{b+\Delta t}, \mathbf{M}^2_{b+\Delta t}\}$ of disjoint SMMs at $t = b + \Delta t$. Subsequently, the matching stage should associate the disjoint SMMs with the SMD stored in $\mathcal{G}$, as shown in Fig. 4(c). The procedure should identify two associations, each corresponding to an SMM at $t = b + \Delta t$, which should be matched to the same SMM $\mathbf{M}_b$ (strictly speaking, $\mathbf{M}_b$ is a co-regular surface, and not a manifold because it self-intersects), as they originate from the splitting operation. This results in the creation of two new c-bundles, $\mathcal{S}$ and $\mathcal{T}$, which are incorporated into the graph $\mathcal{G}$ with edges extending from the c-bundle $\mathcal{R}$, as depicted in Fig. 4(d).

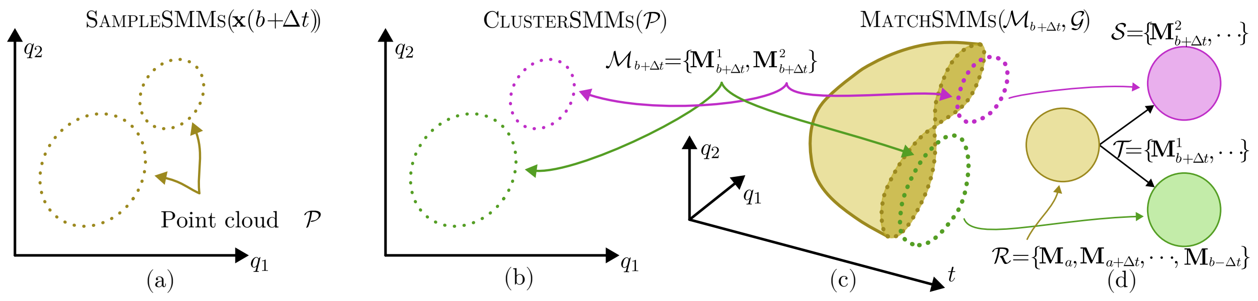
SampleSMMs($\mathbf{x}(b+\Delta t)$)
q_2
q_1
Point cloud $\mathcal{P}$
(a)
ClusterSMMs($\mathcal{P}$)
q_2
q_1
$\mathcal{M}_{b+\Delta t}$=$\{\mathbf{M}^1_{b+\Delta t}, \mathbf{M}^2_{b+\Delta t}\}$
(b)
MatchSMMs($\mathcal{M}_{b+\Delta t}, \mathcal{G}$)
q_2
q_1
t
(c)
$\mathcal{S}$=$\{\mathbf{M}^2_{b+\Delta t}, \cdots\}$
$\mathcal{T}$=$\{\mathbf{M}^1_{b+\Delta t}, \cdots\}$
$\mathcal{R}$=$\{\mathbf{M}_a, \mathbf{M}_{a+\Delta t}, \cdots, \mathbf{M}_{b-\Delta t}\}$
(d)

Figure 4: (a) Sampling. (b) Clustering. (c) Matching in the SMD. (d) Matching in the graph.

In the matching stage, associations between the newly-created SMMs and the already-computed part of the SMD are stored in a mapping, which we denote as *MatchTo*. The argument of the *MatchTo* mapping are individual SMMs at the current instant, and the value returned is the set of SMMs at the previous instant that are close to the argument SMM (i.e. those SMMs at $t - \Delta t$ that transform into the current SMM at t). Table 1 provides a summary of the transformations observed in the SMD of Fig. 2(c) when traversing the time axis t in both directions, as well as the corresponding values of the *MatchTo* mapping. Note that the creation of manifold $\mathbf{M}^2_c$ (shown in Fig. 2(c)) is a vanishing when traversing $t = c$ in the negative direction. While vanishings are not captured by the *MatchTo* mapping, they are implicitly contained in the final graph $\mathcal{G}$, as c-bundles without successors (nodes without successor edges) that do not reach the final instant of the task are considered vanishings.

The matching stage is responsible for the creation of new c-bundles and the establishment of edges between them. C-bundles are stored in the graph $\mathcal{G}$ as nodes, and their internal structure is represented as an ordered set of stacked SMMs identified as part of the same c-bundle. We denote the c-bundle that contains the SMM $\mathbf{M}$ as $\mathcal{C}(\mathbf{M})$.

After exemplifying the generation and matching procedure of SMD of Fig. 2(c), we will now explain the matching algorithm in detail. Algorithm 3 displays the steps of the matching stage, which takes the set of disjoint SMMs $\mathcal{M}_t$ at instant t as input, in order to match them to the existing SMMs of the SMD at the previous instant.

The matching procedure can be divided into two parts for better comprehension: the spatial analysis (lines 2-7) and the matching process (lines 8-17). The algorithm is started by retrieving the set of disjoint SMMs $\mathcal{M}_{t-\Delta t}$ at the previous task instant from the graph $\mathcal{G}$ in line 2. Then, the spatial analysis is performed to determine which SMMs at instant $t - \Delta t$ are close to the newly computed SMMs $\mathcal{M}_t$. This is achieved by a nested loop that iterates over each pair of SMMs, one from $\mathcal{M}_t$ and one from $\mathcal{M}_{t-\Delta t}$, and checks if they are close to each other. The closeness between two SMMs is determined by d, which is a parameter similar to h, but that also considers the time step to incorporate the extra dimension: $d = \sqrt{\sum_{j=1}^{n} \Delta q_j{}^2 + \Delta t^2}$, where Δq_j is the step in joint j during the discretization

Table 1: Transformations when traversing the SMD of Fig. 2(c) in both directions of axis t.

Traversed critical point	$MatchTo$ mapping		Transformation
	$\mathbf{M}$	$MatchTo(\mathbf{M})$	
$t = b$	$\mathbf{M}^1_{b+\Delta t}$	$\{\mathbf{M}_b\}$	Splitting
	$\mathbf{M}^2_{b+\Delta t}$	$\{\mathbf{M}_b\}$	
$t = c$	$\mathbf{M}_{c+\Delta t}$	$\{\mathbf{M}^1_c\}$	-
	MatchTo does not capture vanishings		Vanishing
If time were traversed in the negative direction:			
$t = b$	$\mathbf{M}_b$	$\{\mathbf{M}^1_{b+\Delta t}, \mathbf{M}^2_{b+\Delta t}\}$	Merging
$t = c$	$\mathbf{M}^1_c$	$\mathbf{M}_{c+\Delta t}$	-
	$\mathbf{M}^2_c$	$\emptyset$	Creation

of its interval in line 5 of Algorithm 2, and Δt is the step resulting from the discretization of the task trajectory $\mathbf{x}(t)$ in line 1 of Algorithm 1.

There exist a variety of approaches to distance computation between point clouds in an n-dimensional space. We have found that a two-step approach is the most efficient: first, we perform a broad check by evaluating the distance between the bounding boxes of both point clouds, and if they are close enough (i.e., closer than d), we proceed to the second step, which involves a more detailed check that computes the distance between every pair of points in both point clouds. The pairwise distances are computed by efficiently querying the k-d tree of the point cloud. If any pair of points (each belonging to a different point cloud) is closer than d, the SMMs are considered to be close, reaching line 7 of Algorithm 3. If this is the case, the SMM $\mathbf{M}^j_{t-\Delta t}$ is added to the set of SMMs close to $\mathbf{M}^i_t$, which is stored in $MatchTo(\mathbf{M}^i_t)$. Upon completion of the nested loop, the mapping *MatchTo*, initialized in line 3, will collect the set $\mathcal{MT}$ of SMMs from the previous instant that are close to each SMM $\mathbf{M}^i_t$ at the current instant.

When the spatial analysis for each SMM is completed, the actual matching process is executed between lines 8-17. For every SMM $\mathbf{M}^i_t$ at the current instant, the set $\mathcal{MT}$ of close SMMs is retrieved from $MatchTo(\mathbf{M}^i_t)$. The rest of the iteration will serve to match $\mathbf{M}^i_t$ to the manifolds in $\mathcal{MT}$. In line 10, the algorithm verifies whether the SMM $\mathbf{M}^i_t$ has come from a single manifold at the preceding instant (i.e., the cardinality of $\mathcal{MT}$ is 1). Should this be true, the SMM $\mathbf{M}_{t-\Delta t}$ is retrieved from the set $\mathcal{MT}$, and the algorithm checks if $\mathbf{M}_{t-\Delta t}$ has not been matched to any other SMM at the current instant (line 12). If such condition is met, the SMM $\mathbf{M}^i_t$ is incorporated into the graph $\mathcal{G}$ as a direct extension (i.e., no topological transformation occurs) of the c-bundle $\mathcal{C}(\mathbf{M}_{t-\Delta t})$ to which $\mathbf{M}_{t-\Delta t}$ pertains. When line 13 is reached, the algorithm skips the next lines and continues to the next SMM in $\mathcal{M}_t$.

Otherwise, when $\mathbf{M}^i_t$ matches to multiple SMMs at the previous instant (condition at line 10 returns false) or the single SMM in $\mathcal{MT}$ belongs to $MatchTo(\mathbf{M}^j_t)$ for any other manifold $\mathbf{M}^j_t$ different from $\mathbf{M}^i_t$ (condition at line 12 is not met), the algorithm initializes a new c-bundle $\mathcal{C}_{new}$, as an ordered set of SMMs, with $\mathbf{M}^i_t$ as the only initial component (line 15). Then, the algorithm iterates over every manifold $\mathbf{M}^j_{t-\Delta t}$ in the set $\mathcal{MT}$, adding an edge in the graph $\mathcal{G}$ from the c-bundle $\mathcal{C}(\mathbf{M}^j_{t-\Delta t})$ to the new c-bundle $\mathcal{C}_{new}$.

Finally, the algorithm returns the graph $\mathcal{G}$, which is composed of nodes that correspond to the c-bundles, and edges that represent portions of the co-regular surfaces that connect these c-bundles. Note that the SMD can subsequently be retrieved at any time from the graph $\mathcal{G}$ by extracting the ordered set of SMMs stacked within each c-bundle.

4.2. Generation of raw joint trajectories

The second stage of the proposed method involves the generation of the raw joint trajectories that will be optimized in the final stage. By tracing a continuous path along the generated SMD, from the initial to the final time instant, a joint trajectory that satisfies the task trajectory and kinematic constraints can be generated. We propose generating raw trajectories along the SMD, which will serve as initial trajectories for the subsequent optimization stage, bypassing the need for direct tracing of the globally optimal path in the SMD.

In the previous stage, we constructed a graph $\mathcal{G}$ that represents the SMD, where each node corresponds to a c-bundle. By combining the c-bundles in a specific order, where the initial c-bundle contains the SMM at the initial instant and the final c-bundle contains an SMM at the final instant, a *c-bundle chain* is formed. A c-bundle chain

Algorithm 3 Matching the set of disjoint SMMs in $\mathcal{M}_t$ to the SMD in $\mathcal{G}$

```
1: procedure MatchSMMs($\mathcal{M}_t, \mathcal{G}$)
2:     $\mathcal{M}_{t-\Delta t} \leftarrow$ Retrieve the set of SMMs at $t - \Delta t$, stored in $\mathcal{G}$
3:     $MatchTo \leftarrow \emptyset$          ▷ Mapping to store the future matching between SMMs
4:     for $\mathbf{M}_t^i \in \mathcal{M}_t$ do
5:         for $\mathbf{M}_{t-\Delta t}^j \in \mathcal{M}_{t-\Delta t}$ do
6:             if $\mathbf{M}_{t-\Delta t}^j$ is close to $\mathbf{M}_t^i$ then
7:                 Add $\mathbf{M}_{t-\Delta t}^j$ to the set of SMMs close to $\mathbf{M}_t^i$, stored in $MatchTo$ for the argument $\mathbf{M}_t^i$
8:     for $\mathbf{M}_t^i \in \mathcal{M}_t$ do          ▷ Now, actually process the matching
9:         $\mathcal{MT} \leftarrow MatchTo(\mathbf{M}_t^i)$          ▷ Retrieve from $MatchTo$ the set of SMMs that are close to $\mathbf{M}_t^i$
10:        if $|\mathcal{MT}| = 1$ then          ▷ if the cardinality of $\mathcal{MT}$ is 1, $\mathbf{M}_i$ matches to a single SMM
11:            $\mathbf{M}_{t-\Delta t} \leftarrow$ the single SMM in $\mathcal{MT}$
12:            if $\mathbf{M}_{t-\Delta t} \notin MatchTo(\mathbf{M}_t^j) \forall j \neq i$ then          ▷ $\mathbf{M}_{t-\Delta t}$ is not close to any other SMM
13:                Add $\mathbf{M}_t^i$ to the c-bundle $\mathcal{C}(\mathbf{M}_{t-\Delta t})$
14:                Continue          ▷ Continue iterating (skip next lines)
15:        Initialize new c-bundle $\mathcal{C}_{new} = \{\mathbf{M}_t^i\}$ and add it to $\mathcal{G}$
16:        for $\mathbf{M}_{t-\Delta t}^j \in \mathcal{MT}$ do
17:            Add edge in $\mathcal{G}$ from $\mathcal{C}(\mathbf{M}_{t-\Delta t}^j)$ to $\mathcal{C}_{new}$
18:    return $\mathcal{G}$
```

$\mathcal{CBC}$ is a sequence of nodes in the graph $\mathcal{G}$ that connects the initial and final instants: $\mathcal{CBC} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_f\}$, where $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$ are adjacent and timewise successive nodes in the graph $\mathcal{G}$, f is the number of c-bundles in the chain, $\mathcal{C}_1$ contains an SMM at the initial instant and $\mathcal{C}_f$ contains an SMM at the final instant ($\mathcal{C}_f$ extends until the final time value). Note that since the c-bundle chains are simple paths (i.e., a path with no repeated nodes) of the graph $\mathcal{G}$, the number of c-bundle chains is finite. Three possible c-bundle chains for a hypothetical graph $\mathcal{G}$ are illustrated in Fig. 5(a). Fig. 5(b) shows the corresponding SMD, dismembered to show the c-bundles in one of the c-bundle chains.

While an infinite number of joint trajectories can be generated from the SMD, our focus is on the extraction of a finite number of joint trajectories, each corresponding to a c-bundle chain. To establish every c-bundle chain, we employ a depth-first search algorithm, as detailed in [34]. The algorithm identifies all simple paths from a source node and a target node within a graph. Consequently, it must be executed for each pair of source and target nodes in the graph $\mathcal{G}$. Here, the source nodes represent the c-bundles to which the SMMs generated at the initial instant belong ($\mathcal{C}_1$ in $\mathcal{CBC}$), and the target nodes correspond to the c-bundles containing every SMMs computed at the final instant ($\mathcal{C}_f$ in $\mathcal{CBC}$). If the initial joint configuration is restricted, as is most often the case in real-time applications where no prior self-motion is desired, the source node will be the c-bundle that includes the SMM that contains the initial joint configuration. For the purpose of this explanation, we assume that the initial joint configuration $\mathbf{q}_0$ is restricted (only one source node is considered), noting that extension to the unrestricted case is trivial.

Next, we propose generating raw joint trajectories along each c-bundle chain. While various methods can be utilized for this purpose, as we will discuss in Section 4.4.2, we present a simple, yet highly efficient approach based on nearest-neighbour queries. Given that the quality of the raw joint trajectories is not crucial, as they will undergo optimization in the final stage of the framework, this method is sufficient. It leverages the same k-d trees used in the matching stage of the graph construction (Section 4.1) to efficiently query the closest point in an SMM. The details of this algorithm are provided in Algorithm 4.

Before getting into the details of the algorithm, we must define the concept of a *manifold portion*. For every adjacent pair of c-bundles $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$ in the c-bundle chain $\mathcal{CBC}$, the manifold portion corresponds to a portion of the co-regular surface that connects the c-bundles. As we reviewed in Section 3, every c-bundle is delimited by co-regular surfaces, which are sets that contain regular and singular points. However, a co-regular surface does not only delimit the c-bundles $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$, but also any other c-bundle that results from a topological transformation (see how $\mathcal{C}_3$ is connected to $\mathcal{C}_a$, $\mathcal{C}_4$ and $\mathcal{C}_b$ in Fig. 5(b)). Co-regular surfaces are split by singularities into what we call "manifold portions", as exemplified in Fig. 5(c). Fig. 5(c) shows a co-regular surface that is a triple-eight curve with two self-intersections. This is where c-bundle $\mathcal{C}_3$ ends and splits into new c-bundles $\mathcal{C}_a$, $\mathcal{C}_4$ and $\mathcal{C}_b$. By removing the self-intersections from this co-regular surface, manifold portions that connect adjacent c-bundles are obtained:

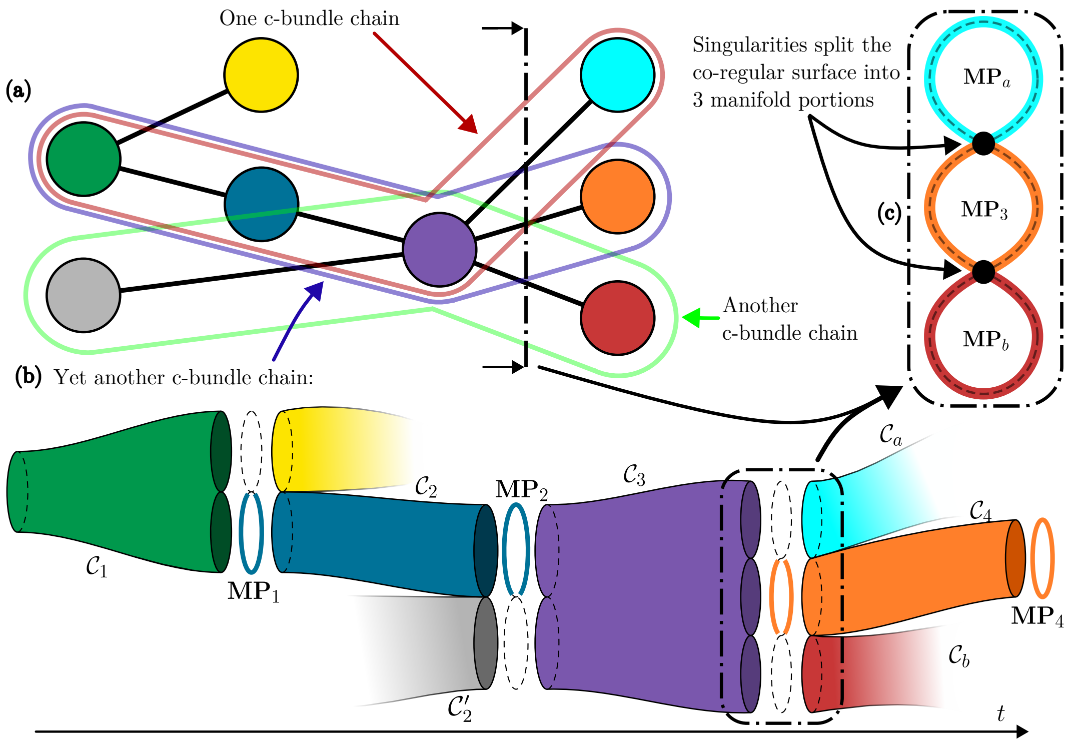

Figure 5: (a) Graph $\mathcal{G}$ and three possible c-bundle chains. (b) Corresponding dismembered SMD for one of the c-bundle chains. (c) Splitting of the co-regular surface into manifold portions.

manifold portion $\mathbf{MP}_a$ is an open curve that connects c-bundle $\mathcal{C}_3$ with $\mathcal{C}_a$, $\mathbf{MP}_3$ is a pair of open curves that connect $\mathcal{C}_3$ with $\mathcal{C}_4$, and $\mathbf{MP}_b$ is another open curve that joins $\mathcal{C}_3$ with $\mathcal{C}_b$. Regarding the notation employed in Algorithm 4, $\mathbf{MP}_i$ is the manifold portion that joins c-bundle $\mathcal{C}_i$ with $\mathcal{C}_{i+1}$ in the considered c-bundle chain. A special case occurs when the last c-bundle in the chain is reached, since there is no next c-bundle, the manifold portion $\mathbf{MP}_f$ is the last SMM in the c-bundle $\mathcal{C}_f$ ($\mathbf{MP}_4$ in Fig. 5(b)).

The algorithm receives as input the c-bundle chain $\mathcal{CBC}$ and the initial joint configuration $\mathbf{q}_0$, which initializes the raw joint trajectory $\mathcal{Q}$ in line 1. Then, the algorithm iterates over every c-bundle in the c-bundle chain $\mathcal{CBC}$ (lines 2-3). For each c-bundle $\mathcal{C}_i$, the algorithm retrieves the manifold portion $\mathbf{MP}_i$ that connects the c-bundles $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$ in the c-bundle chain $\mathcal{CBC}$ (line 4). If $\mathcal{C}_i$ is the last c-bundle in the chain, $\mathcal{C}_{i+1}$ does not exist, and $\mathbf{MP}_i$ simply corresponds to the last SMM in $\mathcal{C}_i$.

During each iteration, the algorithm retrieves the last point (up to that moment) in the raw trajectory $\mathcal{Q}$, denoted as $\mathbf{q}_a$. Then, the nearest point in $\mathbf{MP}_i$ is queried and stored as $\mathbf{q}_b$. At this point of the iteration, both $\mathbf{q}_a$ and $\mathbf{q}_b$ represent points located in limiting SMMs of the currently processed c-bundle $\mathcal{C}_i$. If a straight line is drawn between $\mathbf{q}_a$ and $\mathbf{q}_b$, it will represent the shortest path that traverses the c-bundle $\mathcal{C}_i$, starting from a fixed initial point ($\mathbf{q}_a$). This straight-line path, however, does not accurately represent the actual trajectory, as the straight-line segment that connects $\mathbf{q}_a$ and $\mathbf{q}_b$ will not belong to the SMD in general. Therefore, this straight line serves merely as a reference and is not expected to depict a realistic joint trajectory. To produce this guiding path, the algorithm linearly interpolates between $\mathbf{q}_a$ and $\mathbf{q}_b$, producing n_i points, where n_i corresponds to the number of discretized SMMs comprising the c-bundle $\mathcal{C}_i$. To correct this line segment and "project" it onto the SMD, lines 7-13 are executed. First, recall that, for Algorithm 4, c-bundle $\mathcal{C}_i$ is an ordered collection of n_i homotopic SMMs stacked along the time dimension. Thus, the straight segment between $\mathbf{q}_a$ and $\mathbf{q}_b$ is discretized into a set $\mathcal{Q}_l$ of n_i points, one per each SMM $\mathbf{M}_j$ (line 8). A final loop then iterates over each SMM $\mathbf{M}_j$ in the c-bundle $\mathcal{C}_i$ (lines 9-13). For each SMM $\mathbf{M}_j$, the algorithm retrieves the point $\mathbf{q}_l$ in the linearly interpolated path $\mathcal{Q}_l$ that corresponds to the same instant as $\mathbf{M}_j$. The algorithm queries the nearest point $\mathbf{q}_c$ in the SMM $\mathbf{M}_j$ to the interpolated point $\mathbf{q}_l$ and incorporates it into the raw trajectory $\mathcal{Q}$. This query is executed efficiently using the k-d tree of the SMM, which was computed during the matching stage of the graph construction (Section 4.1).

Finally, the algorithm returns the raw joint trajectory $\mathcal{Q}$, which is formed by the points that approximate a joint

Algorithm 4 Generation of a raw joint trajectory along a c-bundle chain $\mathcal{CBC}$

```
1: $\mathcal{Q} = \{\mathbf{q}_0\}$    ▷ Initialize the raw trajectory with the initial joint configuration
2: for $i = 1, 2, \ldots, f$ do
3:     $\mathcal{C}_i \leftarrow i$-th c-bundle in $\mathcal{CBC}$
4:     $\mathbf{MP}_i \leftarrow$ Manifold portion that connects $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$ in $\mathcal{CBC}$
5:     $\mathbf{q}_a \leftarrow$ Last point in $\mathcal{Q}$
6:     $\mathbf{q}_b \leftarrow$ Closest point in $\mathbf{MP}_i$ to $\mathbf{q}_a$
7:     $n_i \leftarrow$ Number of SMMs stacked in $\mathcal{C}_i$
8:     $\mathcal{Q}_l \leftarrow$ Linear interpolation from $\mathbf{q}_a$ to $\mathbf{q}_b$, generating $n_i$ points
9:     for $j = 1, 2, \ldots, n_i$ do
10:        $\mathbf{M}_j \leftarrow j$-th SMM stacked in $\mathcal{C}_i$
11:        $\mathbf{q}_l \leftarrow$ Point in $\mathcal{Q}_l$ corresponding to the same instant as $\mathbf{M}_j$
12:        $\mathbf{q}_c \leftarrow$ Closest point in $\mathbf{M}_j$ to $\mathbf{q}_l$
13:        Add $\mathbf{q}_c$ to $\mathcal{Q}$
14: return $\mathcal{Q}$
```

trajectory along the specified c-bundle chain $\mathcal{CBC}$ that satisfies the task trajectory.

4.3. Trajectory optimization

The final stage of the proposed method involves the optimization of the joint trajectory generated for each c-bundle chain. The optimization is performed by iteratively solving a constrained QP problem that moves each joint trajectory point along the SMM of the associated instant. Starting from an initial joint trajectory $\mathcal{Q}$, the aim is to minimize a cost function. In line with the popular methods for general purpose (not SMM-restricted) trajectory optimization [10, 11], we employ a cost function that quantifies the sum of the squared velocities along the trajectory:

$$\mathbf{f}(\varphi) = \frac{1}{2}\|\mathbf{V}\varphi + \mathbf{e}\|^2 \tag{2}$$

where φ is a column vector that corresponds to the flattened joint trajectory $\mathcal{Q}$, $\mathbf{V}$ is the finite difference matrix that computes the velocities from the joint trajectory, and $\mathbf{e}$ is a vector that contributes with the invariant initial joint configuration:

$$\begin{aligned}
\varphi &= [q_{11}, q_{12}, \cdots, q_{1n}, q_{21}, q_{22}, \cdots, q_{2n}, \cdots, q_{k1}, q_{k2}, \cdots, q_{kn}]^{\mathrm{T}} \\
\mathbf{V} &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}_{k\times k} \otimes \mathbf{I}_{n\times n} \\
\mathbf{e} &= \begin{bmatrix} -\mathbf{q}_0^{\mathrm{T}} & \mathbf{0}_{n(k-1)\times 1}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}
\end{aligned} \tag{3}$$

where q_{ij} denotes the j-th joint coordinate at the i-th point in the joint trajectory, $\otimes$ denotes the Kronecker product, k is the number of points in the joint trajectory $\mathcal{Q}$, $\mathbf{q}_0$ is the initial joint configuration, and the subscript $p \times q$ denotes the dimensions of the matrix to which it accompanies.

The parent problem (minimization of $\mathbf{f}(\varphi)$) is solved sequentially by a local approximation. This approach is analogous to other common optimization techniques, such as the ones employed in Sequential Quadratic Programming (SQP) or Sequential Convex Programming (SCP). A first-order Taylor expansion is used to locally approximate $\mathbf{f}(\varphi)$:

$$\mathbf{f}(\varphi) \approx \mathbf{f}(\varphi_{current}) + \nabla\mathbf{f}(\varphi_{current})^{\mathrm{T}}(\varphi - \varphi_{current}) \tag{4}$$

where $\varphi_{current}$ is the current joint trajectory, and $\nabla\mathbf{f}(\varphi_{current})$ is the gradient of the cost function with respect to the joint trajectory, which is computed as $\nabla\mathbf{f}(\varphi_{current}) = \mathbf{V}^{\mathrm{T}}(\mathbf{V}\varphi_{current} + \mathbf{e})$.

The local approximation is then minimized by solving the following QP problem:

$$\underset{\Delta\varphi}{\text{minimize}} \quad \frac{s}{2}\|\Delta\varphi\|^2 + \nabla\mathbf{f}^{\mathrm{T}}\Delta\varphi \tag{5}$$

where s is a regularization constant that determines the penalization of large changes in $\Delta\varphi$ [35]. We have set $s = 10$ in our experiments, as it provides a good balance between convergence speed and solution quality. Finally, the solution is added to the current joint trajectory following $\varphi = \varphi_{current} + \Delta\varphi$.

The QP problem can be further constrained to ensure that, for each instant i, the joint coordinates $\mathbf{q}_i$ remain in the corresponding r-dimensional SMM, instead of allowing free movement in the n-dimensional joint space. This constraint reduces the number of decision variables from nk to rk. The restriction is imposed following the equality:

$$\Delta\varphi = \mathbf{NS}^{\mathrm{T}}\,\mathbf{d} \tag{6}$$

where $\mathbf{d} = [d_{11}, d_{12}, \cdots, d_{1r}, \cdots, d_{k1}, \cdots, d_{kr}]^{\mathrm{T}} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_k]^{\mathrm{T}}$ is the rk-dimensional decision vector that signifies movement along the axes of the null-space basis of the Jacobian matrix at each joint configuration. These bases are contained in $\mathbf{NS}$ as follows:

$$\mathbf{NS} = \begin{bmatrix} \mathbf{NS}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{NS}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{NS}_k \end{bmatrix}_{rk \times nk} \tag{7}$$

where $\mathbf{NS}_i$, of size $r \times n$, is an orthonormal basis of the null-space of the Jacobian matrix $\mathbf{J}(\mathbf{q}_i)$ at the i-th joint trajectory point $\mathbf{q}_i$. As depicted in Figs. 3 and 6, the null-space basis comprises r linearly independent n-dimensional vectors that are tangent to the SMM at the joint configuration $\mathbf{q}_i$. Movement along the null-space basis approximates movement along the SMM. However, moving along tangent vectors does not guarantee that the trajectory points will remain on the SMM. To ensure that trajectory points precisely align with the SMM, a Newton-based correction is necessary, as discussed in Section 3.2.1. This step does not significantly impact computation, as the method typically converges in a single iteration, since the restrictions R1 and R2 that will be introduced in the final QP problem ensure that points do not deviate significantly from the current joint trajectory.

The final QP problem that is solved at each iteration is then:

$$\begin{aligned} \underset{\mathbf{d}}{\text{minimize}} \quad & \frac{s}{2}\|\mathbf{NS}^{\mathrm{T}}\,\mathbf{d}\|^2 + \nabla\mathbf{f}^{\mathrm{T}}\,\mathbf{NS}^{\mathrm{T}}\,\mathbf{d} \\ \text{subject to} \quad & \mathbf{d} \in \mathcal{A} \quad \text{(R1)} \\ & \mathbf{d} \in \mathcal{B} \quad \text{(R2)} \end{aligned} \tag{8}$$

and the solution is added to the current joint trajectory following $\varphi = \varphi_{current} + \mathbf{NS}^{\mathrm{T}}\,\mathbf{d}$.

The restriction R1, which establishes a trust region around the current joint trajectory point, ensures the validity of the approximation in Eq. (4). This trust region is an r-dimensional hypercube centred at $\mathbf{q}_i$ and has its axes aligned with basis $\mathbf{NS}_i$. It is represented in Fig. 6 by its blue boundaries. This region also prevents excessive null-space movement, thus reducing the number of iterations required for the Newton-based correction required by Eq. (6). The side length of the hypercube is 2λ. A smaller λ enhances the accuracy of the cost function approximation in Eq. (4) and the validity of the null-space basis, tangent to the SMM, as an approximate movement along the SMM, but increases the number of iterations needed to minimize the parent problem $\mathbf{f}(\varphi)$. Conversely, a larger λ reduces the number of iterations but compromises the approximation accuracy, leading to more iterations in the Newton-based correction and potential convergence issues, such as skipping over minima of $\mathbf{f}(\varphi)$. Our experiments indicate that a value of $\lambda = 0.1$ exhibits a good balance between accuracy and convergence speed. The trust region depicted in Fig. 6 is exaggerated for illustrative purposes.

The restriction R2 further constraints the movement of the joint trajectory points, to ensure that they remain within the boundaries of their associated SMM. Without this restriction, the points could potentially drift into unfeasible regions of the joint space (i.e., out of the SMM), such as those violating joint limits or causing collisions. A previous process must be performed in order to determine the boundaries of the SMM at the graph construction stage (Section 4.1). Specifically, during the sampling phase, any joint configurations determined as unfeasible (i.e., pertain to the SMM but produce collisions or joint limit violations) are stored in a point cloud $\bar{\mathcal{P}}_i$ that represents the invalid region of the SMM associated with the i-th task trajectory point. Note that this process does not require additional computational effort, as these points are already computed during the graph construction stage, independently of this restriction, and the overhead of simply storing them is negligible.

In Fig. 6, the SMM and the unfeasible region of the SMM are represented in grey and red, respectively. Since we have used a small yet finite step in the discretization process, the actual boundary of the SMM is unknown and will lie somewhere within the white stripe, situated between the grey and red regions. Therefore, we can only rely

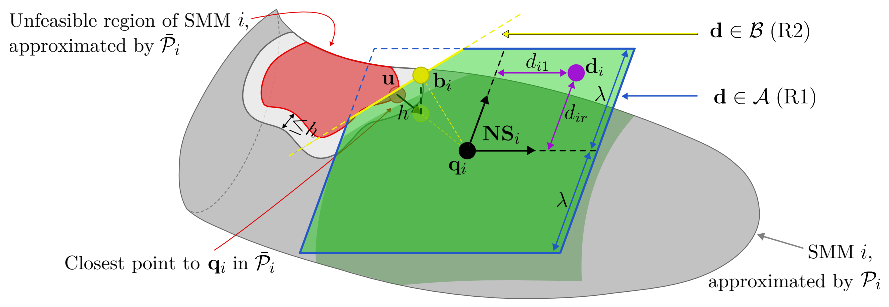
Unfeasible region of SMM i,
approximated by $\bar{\mathcal{P}}_i$
$\mathbf{d} \in \mathcal{B}$ (R2)
$\mathbf{u}$
$\mathbf{b}_i$
d_{i1}
$\mathbf{d}_i$
λ
$\mathbf{d} \in \mathcal{A}$ (R1)
h
h
d_{ir}
$\mathbf{NS}_i$
$\mathbf{q}_i$
λ
Closest point to $\mathbf{q}_i$ in $\bar{\mathcal{P}}_i$
SMM i,
approximated by $\mathcal{P}_i$

Figure 6: QP problem restrictions in a hypothetical ($r = 2$)-dimensional SMM embedded in an ($n = 3$)-dimensional joint space. Trust region in blue (R1), SMM boundary in yellow (R2).

on a conservative estimation of the boundary. For this purpose, we propose a simple three-step process to determine the conservative boundary point $\mathbf{b}_i$ for each joint trajectory point $\mathbf{q}_i$.

First, the nearest point in the unfeasible point cloud $\bar{\mathcal{P}}_i$ for each joint trajectory point $\mathbf{q}_i$ is queried, marked as a red point $\mathbf{u}$ in Fig. 6. Given that the maximum width of the white stripe is determined by the maximum distance between discretized points during the graph generation stage (denoted as h in Section 4.1), point $\mathbf{u}$ will be at most a distance of h from the actual SMM boundary. Note that the representation of h has also been exaggerated in Fig. 6 for illustrative purposes. In practice, h is a small value, and such stripe is extremely narrow. The second step is to shift point $\mathbf{u}$ towards $\mathbf{q}_i$ by a distance of h, and projecting the result to the r-dimensional subspace spanned by $\mathbf{NS}_i$ (green plane in Fig. 6) to obtain $\mathbf{b}_i$. This adjustment, although over-conservative in most cases, ensures that the joint trajectory points do not invade the forbidden regions $\bar{\mathcal{P}}$ of the SMMs. Lastly, the following inequality imposes restriction R2:

$$\mathbf{b}_i^{\mathrm{T}}\,\mathbf{d}_i \leq \|\mathbf{b}_i\|^2 \tag{9}$$

This inequality is obtained by linearizing an r-dimensional sphere that is centred at $\mathbf{q}_i$ and passes through $\mathbf{b}_i$, which produces a hyperplane in the subspace spanned by $\mathbf{NS}_i$, depicted in Fig. 6 in yellow. Note that such linearization is done to be able to express the restriction as a linear inequality, which is a requirement for QP problems.

The minimization of the cost function is achieved through an iterative process, as detailed in Algorithm 5. In every iteration, the restrictions R1 and R2 are computed first. Then, the QP problem in Eq. (8) is solved subject to R1 and R2, by employing a QP solver such as [36] or [37]. The solution is added to the current joint trajectory, and the process is repeated until the cost function converges. The convergence is dictated by the maximum magnitude of $\mathbf{d}$, which must be less than a small value ε (e.g., a value of 10^{-3} is used for the experiments of this work).

Algorithm 5 Trajectory optimization

```
1: while $\max\left(|d_{ij}|\right) > \varepsilon$ do
2:     Compute restrictions (R1) and (R2)
3:     $\mathbf{d} \leftarrow$ Solve QP problem in Eq. (8) subject to (R1) and (R2)
4:     $\varphi \leftarrow \varphi + \mathbf{NS}^{\mathrm{T}}\,\mathbf{d}$
5: return $\varphi$
```

4.4. *Variants*

The proposed method can be adapted to satisfy different requirements in terms of computational speed, optimality or generalization. In this section, we present several variants for every stage of the presented framework, which can be combined to satisfy the specific requirements of the application. Furthermore, we discuss the selection of the most suitable variant for each stage, based on the application's requirements. Table 2 summarizes the proposed variants for each stage of the framework.

4.4.1. *SMD and graph generation*

Even though any method presented in Section 3.2 can be employed to generate the SMMs in the graph construction stage, we have selected a *sampling*-based method due to its efficiency. However, our proposed method for SMD generation, extended from the method presented in [13] for SMM generation, requires solving the IKP by algebraic

Table 2: Proposed variants, briefly summarized, and their sections for the different stages of the proposed framework.

SMD and graph generation	**Generation of raw joint trajectories**	**Trajectory optimization**
Sampling (Section 4.1) Sweeping every combination of r joints, and solving the IKP for the remaining.	**Nearest Neighbour (NN)** (Section 4.2) Querying nearest neighbours (k-d trees) to efficiently traverse the $\mathcal{CBC}$.	**Standard** (Section 4.3) Completely optimizing (until convergence) every raw joint trajectory.
Continuation (Section 4.4.1) [14, 29] for computing the SMM at $\mathbf{x}(0)$. Task propagation, resampling and expansion for the subsequent SMMs.	**Pathfinding Algorithms (PA)** (Section 4.4.2) Running pathfinding algorithms (e.g., A* or RRT) in the entire $\mathcal{CBC}$.	**Continuation-friendly** (Section 4.4.3) Substituting the SMM boundaries restriction R2 by a joint-limits restriction (RV1) and a local approximation of collisions (RV2).
Other (Section 3.2.3) Employing any method for SMM generation for every value of $\mathbf{x}(t)$.	**Hybrid NN-PA** (Section 4.4.2) NN method to find the points in every **MP**, PA to establish the trajectory along every c-bundle.	**Online** (Section 4.4.3) Simultaneous optimization and following of the trajectory.

elimination for every combination of r joint coordinates, which may not always be possible, especially in complex kinematic chains. In such cases, we propose employing a *continuation*-based method to generate the SMD, which is more computationally expensive, but is generally applicable to any kinematic chain.

One could simply adapt the algorithm presented in Algorithm 1, substituting the SampleSMMs and ClusterSMMs functions with any continuation-based method from the literature, and leaving the rest of the algorithm unchanged. Nevertheless, we present a more efficient alternative that leverages the Jacobian matrices that are computed in the continuation-based method, in order to propagate the already computed SMMs to generate the graph.

Starting from the initial joint configuration $\mathbf{q}_0$, conventional continuation methods, such as [14] ($r = 1$) or [29] (when $r > 1$), can be employed to compute the SMM at the initial task value $\mathbf{x}(0)$. Note that this will only compute the SMM to which the initial joint configuration belongs, and not every disjoint SMM. If the entire SMD is required, numerous seed joint configurations should be sampled to ensure that all disjoint SMMs are identified. However, as there is no guarantee that every existing disjoint SMM will be identified (Section 3.2.1), we recommend working with the initial SMM only, and expand the SMD from there. This is acceptable when using our method for single-query applications, as the other SMMs for $\mathbf{x}(0)$ that are not identified would belong to nodes (c-bundles) of the graph that will not be visited during the execution of the task because this would require traversing time backwards (e.g., like starting in c-bundle $\mathcal{C}_2$ in Fig. 5(b), advancing in time to c-bundle $\mathcal{C}_3$ in the same figure, and then going back in time to c-bundle $\mathcal{C}_2'$).

Once the initial SMM has been generated, since the Jacobian matrices are already computed for every joint configuration sampled in the SMMs, the evolution of the SMMs over time can be efficiently calculated by solving:

$$\mathbf{q}_{i+1} = \mathbf{q}_i + \mathbf{J}^\dagger(\mathbf{q}_i)\,(\mathbf{x}(t+\Delta t) - \mathbf{x}(t)) \tag{10}$$

which effect propagating points from the SMM at $\mathbf{x}(t)$ to the SMM at $\mathbf{x}(t + \Delta t)$ is exemplified in Fig. 3(c) by the orange arrows.

However, similarly to the continuation method, the obtained points do not exactly belong to the SMM at $\mathbf{x}(t+\Delta t)$, as this approximation is only valid for infinitesimal displacements. A Newton-based correction is required to ensure that the points lie exactly on their SMM, as in Section 3.2.1. Fortunately, both of these computations can be accelerated by vectorizing the operation, which allows for the simultaneous calculation and subsequent correction of all points within the SMM, instead of performing these operations point by point.

Then, the resulting SMMs must be resampled to guarantee a dense representation within the graph (as shown in Fig. 3(c), when propagating each joint-space point $\mathbf{q}_i$ with Eq. (10), there may appear low-density regions in the SMM that must be filled with the depicted green points). Moreover, the boundaries of the SMMs should be expanded to account for the potential expansion of the SMMs from one instant to the next (again, as shown in 3(c), propagation with Eq. (10) may result in low-density regions that must be populated by the blue points to determine where the SMM ends due to entering a forbidden region). The process of resampling and boundary expansion is analogous to the continuation method under discussion.

The SMD is generated by applying Eq. (10) iteratively to every joint configuration $\mathbf{q}_i$ at every instant. This process yields the SMMs for the subsequent instant, which are then resampled, and their boundaries extended. Furthermore, the matching stage presented in Section 4.1 is not necessary, since the neighbourhood of each SMM is determined by the SMMs from which the propagation originated. That is, the neighbour of SMM $\mathcal{M}_{i+1}$ is the SMM $\mathcal{M}_i$, which is the SMM containing the points $\{\mathbf{q}_i \in \mathcal{M}_i\}$ that were propagated using Eq. (10) to generate the points $\{\mathbf{q}_{i+1} \in \mathcal{M}_{i+1}\}$.

Note that Eq. (10) is not a direct replacement for the matching stage presented in Algorithm 3, as the latter operates without prior knowledge of the SMMs (i.e., $\mathbf{q}_i$ that lead to $\mathbf{q}_{i+1}$ are not known), and they both serve different purposes. In summary, the sampling method presented in Section 4.1 and the continuation method are not fully interchangeable, in the sense that one method does not perfectly substitute the other. Moreover, as said previously, continuation can only generate SMMs that are expanded from the current ones. If there are new manifolds that are disjoint from the current ones, they would not be generated by continuation, whereas the sampling method of Section 4.1 would generate them.

4.4.2. Generation of raw joint trajectories

In the generation of raw joint trajectories, we introduced a method that, while not optimal, is significantly efficient. This method, which we will refer to as the *nearest neighbour* (NN) approach, leverages the same k-d trees used in the matching stage of graph construction to efficiently query the closest points in the SMMs. We opted for this method due to its efficiency, sacrificing the optimality of these trajectories, since it is not crucial at this stage, as they will be optimized in the subsequent stage. However, if the application, due to generalization demands, requires the use of the graph construction method discussed in Section 4.4.1, or a short window of time is allocated for the trajectory optimization stage, it may be more beneficial to invest the extra computational effort in generating the raw joint trajectories more optimally at this stage.

Since the discretized points that form the SMD can be easily converted into a graph, any classical *pathfinding algorithm* (PA) can be employed to generate the raw joint trajectories. The A* algorithm, for example, is a popular choice for path planning in graphs, and can be employed to generate the raw joint trajectories optimally. However, the computational cost of employing such algorithms is significantly higher than the nearest neighbour method, and may not be suitable for real-time applications. The RRT algorithm, on the contrary, is a more efficient alternative that can be employed to generate the raw joint trajectories in a more optimal manner than the nearest neighbour method, but with a lower computational cost than A*. While utilizing such pathfinding algorithms should be the most rigorous and general method for connecting a point in $t = 0$ to a point in $t = t_f$ (final time), the computational effort required for their generation may be prohibitive when planning on the entire $\mathcal{CBC}$.

To mitigate this issue, we propose a *hybrid* combination of the two methods, employing the NN-based method to generate the points that belong to manifold portions that separate c-bundles of the c-bundle chain, and then employing a PA to generate the intermediate points along the c-bundles that form the c-bundle chain. This would mean the replacement of lines 7-13 in Algorithm 4 with the PA of choice (e.g., any RRT-based algorithm), which returns a path between the points $\mathbf{q}_a$ and $\mathbf{q}_b$. This approach would be a middle ground between the efficiency of the NN method and the optimality of the PA method, and would be particularly useful when the application required that the raw joint trajectories were near-optimal, but computational speed were also a concern.

4.4.3. Trajectory optimization

The optimization of the raw joint trajectories is performed by iteratively solving a constrained QP problem that moves the trajectory along the SMD, but restricting every point to remain in its corresponding SMM. We will refer to the method presented in Section 4.3 as the *standard* optimization procedure, since it optimizes the entire joint trajectory until convergence. However, this method can be adapted to satisfy different requirements in terms of computational speed or generalization.

When selecting the continuation-based method presented in Section 4.4.1 as a more general variant of the graph construction, it will not be possible to store the point cloud $\bar{\mathcal{P}}$ of unfeasible joint configurations that represent the forbidden regions of the SMM during the graph construction stage, rendering the restriction R2 in the standard optimization stage not applicable. For such cases, we present the *continuation-friendly* variant of the method, in which the restriction R2 is replaced by two new restrictions, RV1 and RV2, that ensure that the joint trajectory points remain within the permitted boundaries of the SMM without the need for the point cloud $\bar{\mathcal{P}}$.

Boundaries of SMMs appear as a result of entering a region that does not satisfy the kinematic constraints of the robot, such as joint limits or collisions. Such boundaries can be approximated by two new restrictions. The first restriction RV1 ensures that the joint trajectory points do not violate the joint limits, and is imposed by the following inequalities at every i-th trajectory point:

$$\begin{aligned} \mathbf{q}_i + \mathbf{NS}_i^{\mathrm{T}}\,\mathbf{d}_i &\geq \mathbf{q}_{\min} \qquad \text{(RV1)} \\ \mathbf{q}_i + \mathbf{NS}_i^{\mathrm{T}}\,\mathbf{d}_i &\leq \mathbf{q}_{\max} \end{aligned} \tag{11}$$

where $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$ are the vectors that contain the lower and upper joint limits, respectively, and the inequalities are coordinate-wise.

The second restriction RV2 ensures that the joint trajectory points do not violate the collision constraints. It is based on a local approximation of the distance between the robot and the obstacles, as in [38]. In every iteration of the optimization process, the points $\mathbf{p}$ of the robot and $\mathbf{o}$ of the obstacle that are closest are queried. Then, the following restriction is imposed:

$$\mathbf{n}_i^{\mathrm{T}}\,\mathbf{J}_{\mathbf{p}_i}(\mathbf{q}_i)\,\mathbf{NS}_i^{\mathrm{T}}\,\mathbf{d}_i \leq \|\overrightarrow{\mathbf{p}_i\mathbf{o}_i}\| - d_{\mathrm{safe}} \quad \text{(RV2)} \tag{12}$$

where the subindex i denotes the i-th trajectory instant; $\mathbf{n}_i = \overrightarrow{\mathbf{p}_i\mathbf{o}_i}/\|\overrightarrow{\mathbf{p}_i\mathbf{o}_i}\|$ is the unit common normal from robot towards obstacle; $\mathbf{J}_{\mathbf{p}_i}(\mathbf{q}_i)$ is the Jacobian matrix that maps the joint velocities to the velocities of the point $\mathbf{p}_i$; and d_{safe} is the safety distance that the robot must maintain with the obstacles, which is usually set to zero or a small distance. The QP problem in Eq. (8) should be updated to include these two new restrictions RV1 and RV2, replacing in practice restriction R2.

Another variant is the *online* optimization, which is particularly useful when the application requires real-time operation. In contrast to the standard optimization, which optimizes the entire joint trajectory φ until convergence, and then the robot starts moving, the online optimization method optimizes the joint trajectory concurrently with the robot's movement. The principle behind this variant is to optimize the trajectory as much as possible within each sampling period of the controller, interrupting 5 at the end of the sampling period even before it has converged, and then apply the first partially-optimized joint increment to move the robot.

For example, assume that the robot's controller operates at a sample time of 50 ms, and a hypothetical raw joint trajectory φ of 100 points must be optimized. The online optimization method would execute as many iterations of 5 as possible in the 50 ms time window, obtaining a partially optimized trajectory φ^*. Then, the first joint increment $\mathbf{q}_1$ of φ^* would be applied to move the robot, and $\mathbf{q}_1$ would be removed from φ^*, shortening the trajectory to 99 points. This process would be repeated in the following sampling periods, optimizing the shortened trajectory as much as possible within each period, applying the first sample to move the robot and removing it from the trajectory, until the final point is reached or the trajectory has been shortened so much that it can be completely optimized within the sampling period (i.e., when 5 converges due to $\max(|d_{ij}|) \leq \varepsilon$.

Due to its online nature, the online optimization variant should be applied only to the raw trajectory with the lowest cost (instead of applying it to all raw trajectories). Therefore, it is advised to combine this method with the hybrid NN-PA method for the generation of raw joint trajectories, to ensure that the chosen raw trajectory is as close as possible to the globally optimal one.

4.4.4. Variant selection guidelines

Given the high flexibility of this framework, users might be uncertain of which variant combination to select. In this subsection, we provide a set of guidelines to assist users in identifying the most suitable combinations for their specific applications.

The framework is composed of three stages: 1) SMD and graph generation, 2) search of raw joint trajectories, and 3) trajectory optimization. The paper has been structured by proposing a baseline approach for each one of the stages in Sections 4.1, 4.2, and 4.3, respectively. However, there exist situations where these baseline methodologies are not the most appropriate, or simply cannot be applied. For these cases, we have proposed alternative variants for each stage, in order to address different requirements (limited hardware resources, very complex kinematic chains, etc.). Some of these alternatives are interchangeable, while others are complementary. We trust that the explanation of the variants throughout the present section, and the examples shown below, will provide the necessary guidance for understanding how to combine them to achieve the desired results. In Table 3, we propose combinations of variants depending on the degrees of redundancy, the complexity of the kinematic chain, and the desired trade-off between computational cost and optimality.

Table 3: Variant selection guidelines. Please check Table 2 for an overview of all variants and the sections in which they are defined. *: or when solving Eq. (1) for the remaining values of $\mathbf{q}$ (via algebraic elimination) is not simple and continuation is the only option (complex kinematic chains).

Main objective $\rightarrow$	**Global optimality of the solution**	**Real-time performance**
$r = 1$ **with collisions** *	**Continuation**-based SMD computation **Hybrid PA-NN** raw trajectory generation **Continuation-friendly** + **standard** + **parallelized** optimization	**Continuation**-based SMD computation **NN**-based raw trajectory generation **Continuation-friendly** + **online** optimization
$r > 1$ **(or** $r = 1$ **without collisions)**	**Sampling**-based SMD computation **Hybrid PA-NN** raw trajectory generation **Standard** + **parallelized** optimization	**Sampling**-based SMD computation **NN**-based raw trajectory generation **Online** optimization

In terms of generalization, when a complex kinematic chain is involved, which may not permit solving the IKP by algebraic elimination as the sampling method described in section 4.1 requires, we recommend employing the continuation-based variant presented in Section 4.4.1, as it is directly applicable to any robot, regardless of the complexity of its kinematic chain, since it only requires the computation of the Jacobian matrix at each joint configuration.

The dimensionality of the SMMs and, consequently, the SMD, is influenced by the degrees of redundancy inherent in the task and robot. This dimensionality primarily impacts the graph generation stage. For a redundancy of $r = 1$, the SMMs within the SMD are one-dimensional manifolds. In this case, the original continuation-based method for SMM generation, presented by Burdick [14], can be utilized. However, for $r > 1$, the higher-dimensional continuation alternative, introduced in [29], becomes necessary. Despite its successful extension to higher-dimensional SMMs, this method is computationally demanding and unsuitable for real-time applications. In such scenarios, the sampling-based method for graph generation, which offers greater computational efficiency and scalability, is recommended. The sampling-based method is also recommended for redundancy $r = 1$ where collisions are not involved. This will be justified by means of the examples of Section 5.

Regarding the generation of raw joint trajectories, using the hybrid NN-PA approach reduces the time required for post-processing optimization, but it also increases the initial computation time. This trade-off makes hybrid NN-PA more suitable for scenarios where higher-quality final trajectories are preferred. These are the scenarios that benefit from using hybrid NN-PA: if trajectories are going to be fully optimized, hybrid NN-PA is more efficient overall. Otherwise, if a fast solution is needed, and the trajectories are not going to be fully optimized, NN is faster.

Regarding the balance between computational cost and optimality, the choice of which optimization stage to implement presents a trade-off between these two factors. At one end, when optimality is paramount and the application permits sufficient processing time, the standard optimization procedure delineated in Section 4.3 is recommended. This method identifies the globally optimal joint trajectory in every c-bundle chain. Conversely, when computational efficiency is crucial and real-time operation is necessary, the online optimization variant discussed in Section 4.4.3 is advised. This approach enables an almost immediate commencement of movement following the graph generation stage, with the optimization process operating concurrently, refining the joint trajectory as the allocated time window permits. When a balance between computational cost and optimality is sought, a larger time window for the online optimization could be allocated, or the standard optimization could be conducted with a restricted number of iterations, gradually transitioning the objective of the optimization stage from optimizing to smoothing trajectories as the maximum number of iterations of Algorithm 5 allowed in a window decreases.

This compromise between computational cost and optimality is also present in other state-of-the-art methodologies for global redundancy resolution. For instance, the authors of [12] describe their algorithm as rather expensive in execution time. To solve this issue (and to extend it to higher task dimensionality), they propose the use of neural networks, which are always associated with some degree of approximation, thus compromising the optimality of the solution. On the contrary, works that prioritize the optimality of their solution, not compromising it for computational efficiency, such as [15, 19], are computationally expensive. In our framework, we offer the user the possibility to prioritize either computational efficiency or optimality, or to find a balance between the two, by selecting the most suitable variant for each stage.

5. Experimental results

A series of simulations have been conducted for the evaluation of the proposed framework for global redundancy resolution. For evaluating the performance of the proposed method, we have employed the cost function defined in Eq. (2), and the runtime of the different stages of the framework, as assessment metrics. The experiments were run on a machine equipped with an Intel Core i7-9700F CPU and 32 GB of RAM, under the Ubuntu 24.04 operating system, using Python 3.11. FCL 0.7 [39] was employed for collision detection, while the QP solver used was ProxSuite 0.6.2 [36].

5.1. Example 1: 3R planar manipulator

The first experiment considered corresponds to a 3R planar manipulator tasked with following a 2-dimensional end-effector trajectory. Two ellipses were placed in the workspace, representing obstacles that the entire manipulator must avoid $[(x-0.5)^2/1^2+(y-0.7)^2/0.25^2 \leq 1$ and $(x-1)^2/0.3^2+(y+1)^2/0.2^2 \leq 1]$. Due to such demanding obstacle-avoidance constraints, the joint limits were set to $[-2\pi, 2\pi]$ radians for all joints. The task trajectory, corresponding to the end-effector position, was defined as a linear path from $(2.457, -0.793)$ to $(-2.4, 1.5)$ (in meters, relative to the base of the robot), to be completed in 5 seconds, with constant speed. The starting joint configuration was set to $\mathbf{q} = \left[\frac{\pi}{8}, -\frac{\pi}{4}, -\frac{\pi}{6}\right]^{\mathrm{T}}$ radians.

For this experiment, we used the continuation-based approach for the graph generation stage, which is the best choice when the kinematic equations of the robot are difficult to solve via algebraic elimination methods as the sampling method described in Section 4.1. Note that this is not necessary for the 3R manipulator, but we start with this choice so that it can be compared later with the sampling method. In addition, we seek to completely optimize the joint trajectories regardless of the computational cost.

The task trajectory was discretized into $k = 100$ points, resulting in a time step of $\Delta t = 0.05$ seconds, which provides sufficient resolution for the task analysis. A value of $\rho = 0.3$ was used for the continuation-based generation of the SMD.

Regarding optimality, we aimed to obtain the optimal trajectory for the task. Therefore, we selected the standard optimization method, which optimizes every joint trajectory until convergence. Having employed the continuation-based graph-generating method, which does not produce the point cloud $\bar{\mathcal{P}}$ of unfeasible joint configurations (i.e., R2 is not applicable), the restrictions R1, RV1, and RV2 were imposed in the QP problem, as discussed in Section 4.4.3. We selected the nearest-neighbour-based method for the generation of raw joint trajectories, since it is significantly more efficient than the pathfinding algorithms, and the quality of the raw joint trajectories is not crucial at this stage, since they will be optimized in optimization process.

Figs. 7(a-f) illustrate the results of the simulation in the form of snapshots of a video, which is available in the supplementary material (*ex1_3r.mp4*). They display six instants of the simulation, which correspond to the initial and final moments of the task trajectory ($t = 0$ and $t = 5$), as well as four intermediate instants, which correspond to every transition between two c-bundles ($t = 0.05$, $t = 0.9$, $t = 1.55$, and $t = 3.3$). Fig. 7(g) shows the resulting graph $\mathcal{G}$, where the nodes represent the c-bundles of the SMD, and the edges represent the connections between them. Four distinct c-bundle chains that have been found for the task are highlighted in the graph. For simplicity, c-bundle graphs have been represented throughout the paper as graphs with only topological information (connections between c-bundles), representing them as circles without timespan information. However, in this example, for clarity, we have faithfully represented the time instants at which each c-bundle starts and ends, so that it is clear to the reader why some c-bundles do not complete the desired task, due to vanishing before reaching the end time.

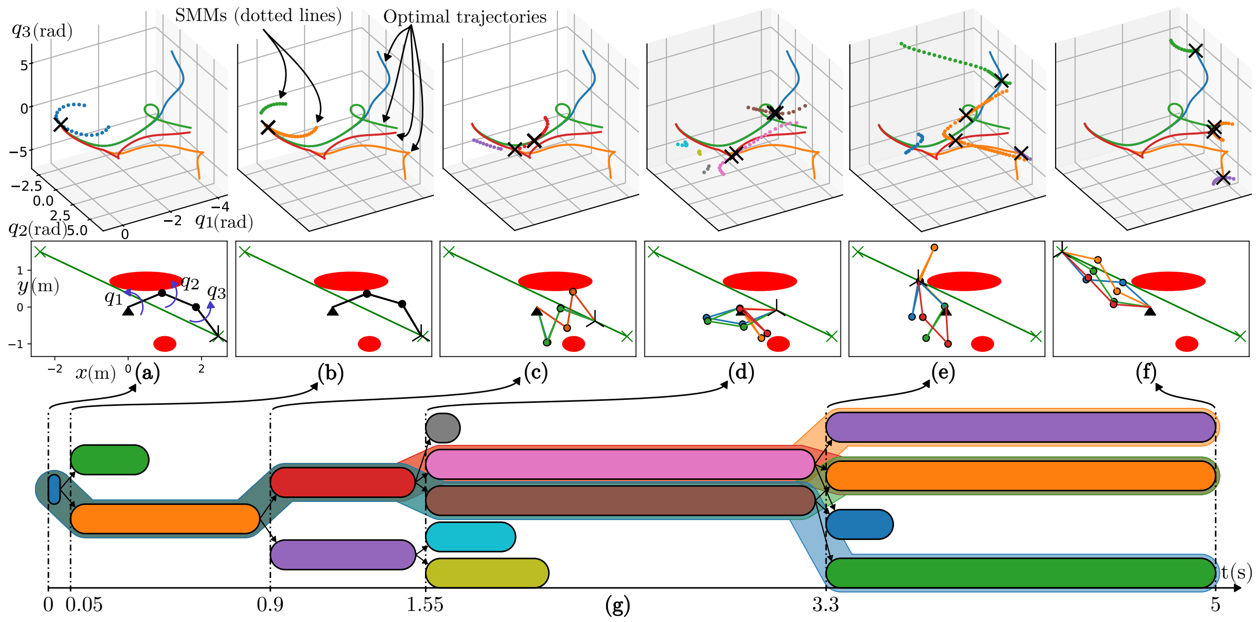

Figure 7: (a-f) Snapshots of the simulation. (g) C-bundle graph $\mathcal{G}$ and every possible c-bundle chain.

Each subfigure presents both the joint space (q_1, q_2, q_3), at the top of Figs. 7(a-f), and the task space (x, y), at the bottom. In the joint space, the optimized joint trajectories that traverse each c-bundle chain are represented along with the SMMs (dotted lines) at the corresponding instant, with the joint configuration of the optimized trajectory at each instant marked with an 'X'. In the task space, the task trajectory is represented in green, and the robot configuration when traversing each c-bundle chain is depicted.

Two different colour schemes are employed in Fig. 7. The first colouring is dictated by the nodes of the graph (c-bundles of the SMD). Every c-bundle is given a different colour in the graph, and the SMMs that are depicted in the joint space (in dotted lines) share the same colour as the c-bundle they pertain to. The other colour scheme corresponds to every obtained c-bundle chain, which are the wide lines marked in the graph. Both the optimal trajectory plotted in the joint space and the corresponding robot representation share the same colour as the c-bundle chain they traverse.

In Table 4, the computational times for each stage of the proposed framework are presented, along with the cost of the generated trajectories. The computational times are given in milliseconds, and the cost of the trajectories corresponds to the cost function in Eq. (2), which quantifies the sum of the squared joint velocities along each trajectory. The provided data corresponds to the average values obtained from 100 independent simulations.

Table 4: Computational times and costs of the generated trajectories for the 3R planar manipulator.

Graph generation	**Continuation: 2092 ms**		*Sampling: 2973 ms*	
Generation of raw trajectories	**20 ms**			
C-bundle chains calculation	*1 ms*			
	$\downarrow \mathcal{CBC}_1$ (blue)	$\downarrow \mathcal{CBC}_2$ (orange)	$\downarrow \mathcal{CBC}_3$ (green)	$\downarrow \mathcal{CBC}_4$ (red)
Raw trajectories	*4 ms*	*5 ms*	*4 ms*	*6 ms*
Optimization (standard)	**4056 ms**	**2509 ms**	**3715 ms**	**4010 ms**
Trajectory cost	**2.144**	**1.591**	**2.07**	**1.195**

Regarding the cost of the returned trajectories, the results show that the globally optimal joint trajectory corresponds to the c-bundle chain $\mathcal{CBC}_4$ (depicted in red in Figs. 7(a-f)), with a cost of 1.195. The SMD is computed in a reasonable time considering that it characterizes the infinite number of solutions to the IKP for the entire task trajectory. In this experiment, the continuation-based method outperforms the sampling-based method. We believe that this is due to the significant overhead that computing whole-body collisions introduces in the sampling-based method, where such computation must be performed for every sampled point. On the contrary, the continuation-based method will stop checking for collisions as soon as the first collision is detected at each end of the manifold (if it is a curve ($r = 1$)), producing a significant reduction in the collision-checking time, which outweighs the additional computational cost of this continuation method (which involves the Jacobian's null-space computation and Newton correction). From the results, we can conclude that, for $r = 1$, when collisions are considered, the continuation-based method is the most efficient method for the generation of the SMD. However, for $r > 1$ (or when $r = 1$ and collisions are not considered), the sampling-based method will be more efficient, as we will demonstrate in the next experiments.

The generation of the raw joint trajectories is significantly fast, as the traversal of the graph is a simple process and the nearest-neighbour search is extremely efficient due to the k-d trees. The largest time consumption is observed in the optimization stage, as the QP problem must be iteratively solved for every derived joint trajectory. It is worth noting that this stage is a good candidate for parallelization, given that the CPU employed possesses enough cores, as the optimization of each joint trajectory is independent of the others. In practice, the computational time of the optimization stage could be reduced to the time required to optimize the most demanding joint trajectory, which, in the case of Table 4, corresponds to the trajectory through $\mathcal{CBC}_1$, which results in an overall time of 6.2 seconds for the complete execution of the proposed framework.

Lastly, we have conducted an additional experiment to demonstrate the method's capability to handle link-link collisions. Specifically, we incorporated collision constraints between the first and third links, which correspond to the links that lie on the same plane in the real robot. The results of this experiment are presented in the supplementary material (*ex1_3r_collisions.mp4*). This capability was implemented by querying the closest points among the involved links, in addition to the closest points between the robot and the obstacles, and imposing the restriction RV2 on the QP problem. This demonstrates that the collision formulation of this paper can equally handle self-collisions between different parts of the robot, or collisions between the robot and the environment.

5.2. Example 2: RPRRRRPR robot

The second experiment considered corresponds to a more complex scenario, where an RPRRRRPR serial robot is assigned with following a 6-dimensional end-effector trajectory, including both position and orientation. This kinematic chain models the HyReCRo robot, which is a biped climbing robot that moves by adhering one of its feet to a surface, and moves the other foot to a new location, adhering it to the surface again. The kinematic model and equations of the RPRRRRPR robot are detailed in [40].

Such a computationally demanding scenario ($n = 8$, $r = 2$) necessitates an efficient combination of variants in order to ensure that the computational time remains within acceptable limits. The sampling-based method for graph generation was selected, as it is the most efficient method for the generation of the SMD when both the DoF and redundancy degree increase. For the optimization stage, the online optimization variant was employed, as the computational time required for the entire optimization of every joint trajectory would suppose a significant overhead, although still manageable. We will only perform a single iteration of the online optimization between discrete instants, even though more iterations could be performed, thus a better trajectory could be obtained, but we want to showcase the worst-case scenario. For the trajectory optimization stage, we chose the hybrid combination of the nearest-neighbour-based method and a pathfinding algorithm. Specifically, we implemented a Depth-first search (DFS) algorithm to generate the points along the c-bundles that connect the co-regular surfaces. This selection results in a slightly more expensive raw trajectory generation stage, but the quality of the raw joint trajectories is significantly improved. Such a choice is done to increase the chances of picking the raw trajectory that will become the globally optimal one after applying the online optimization.

The free foot of the robot is tasked with following a trajectory that has been defined as a cubic spline that interpolates 9 waypoints in 5 seconds while avoiding an obstacle, with the orientation of the free foot being kept constant, considering joint limits as kinematic constraints [40]. Similarly to the previous experiment, the task trajectory was discretized into $k = 100$ points, resulting in a time step of $\Delta t = 0.05$ seconds. The starting joint configuration was arbitrarily set to $\mathbf{q} = [-0.055, 0.18, -0.055, -1.505, -3.076, 0.273, 0.187, 0.273]^{\mathrm{T}}$, where the revolute joints are measured in radians and the prismatic joints in meters. Joint steps of $\Delta q = 0.1$ radians were employed for the revolute joints, and $\Delta q = 0.05$ meters for the prismatic joints, during the sampling-based generation of the SMD and graph.

Figures 8(a-e) illustrate different moments of the simulation, while Fig. 8(f) displays the generated graph. The video of the animated simulation is also available in the supplementary material (*ex2_ rconst.mp4*). The included snapshots correspond to the initial and final moments of the task trajectory ($t = 0$ and $t = 5$), and three intermediate instants ($t = 2.3$, $t = 2.75$, and $t = 3.95$), which correspond to the transitions between c-bundles. Although the online optimization variant was selected, meaning that only the best raw trajectory is optimized, we showcase the optimized joint trajectory resulting from every c-bundle chain in the graph, for the sake of comparison. The same colour scheme as in the previous experiment is employed, i.e.: on the one hand, the nodes of the graph in Fig. 8(f) and the SMMs (in the top of Figs. 8(a-e)) share the same colour as the c-bundle they pertain to. On the other hand, the robot (in the bottom of Figs. 8(a-e)) and the optimized joint trajectory (in the top of Figs. 8(a-e)) share the same colour as the corresponding c-bundle chains of Fig. 8(f). Green and orange trajectories correspond to the fully optimized joint trajectories. Also, the top of Figs. 8(a-e) show the trajectory optimized following the online variant; this is represented as a blue continuous line that is almost indistinguishable from the fully optimized green trajectory (see the difference in the top of Fig. 8(d)). This demonstrates that the less computationally expensive online optimization yields results that are extremely close to the fully optimized trajectory.

It is worth noting that the plotted SMMs are ($r = 2$)-dimensional manifolds embedded in the ($n = 8$)-dimensional joint space, which is why they are represented as surfaces (approximated by point clouds) in the top part of Figs. 8(a-e). Since an 8-dimensional space cannot be visualized directly, we chose to project the SMMs onto the subspace of the first, fourth and sixth joints of this robot [40], which are denoted as $(\varphi_{1A}, \theta_A, \varphi_{2B})$ in Figs. 8(a-e). This is the reason why the blue and orange c-bundles, which are present at the initial instants, seem to be connected in Fig. 8(a), when in reality they are not connected in the full 8-dimensional joint space.

Table 5 presents the computational times for each stage of the proposed framework, along with the cost of the trajectory. Although the number of DoF and redundancy have been increased, it is remarkable that the computational time of the graph generation stage remains within the same order of magnitude as the previous experiment, thanks to the efficiency of the sampling-based method. Further examples of efficiency of sampling the SMMs are given in [41]. The online optimization of the best raw joint trajectory significantly reduces the computational time of the optimization stage, as only a single iteration of the QP problem is required to start the movement of the robot. Moreover, the time taken to perform an iteration remains under 30 ms, which allows for the online optimization of the trajectory within the 50 ms between successive time steps. By comparing in the last two rows of Table 5 the cost of the online-optimized trajectory with that of its completely optimized counterpart, it is evident that the online optimization variant produces a trajectory of slightly higher cost. However, the difference in cost is not significant, as can be observed in the joint-space trajectories in Fig. 8(d) (top), due to employing an efficient pathfinding algorithm to generate the raw joint trajectories.

This experiment demonstrates the efficiency of the sampling-based method for the graph generation in scenarios

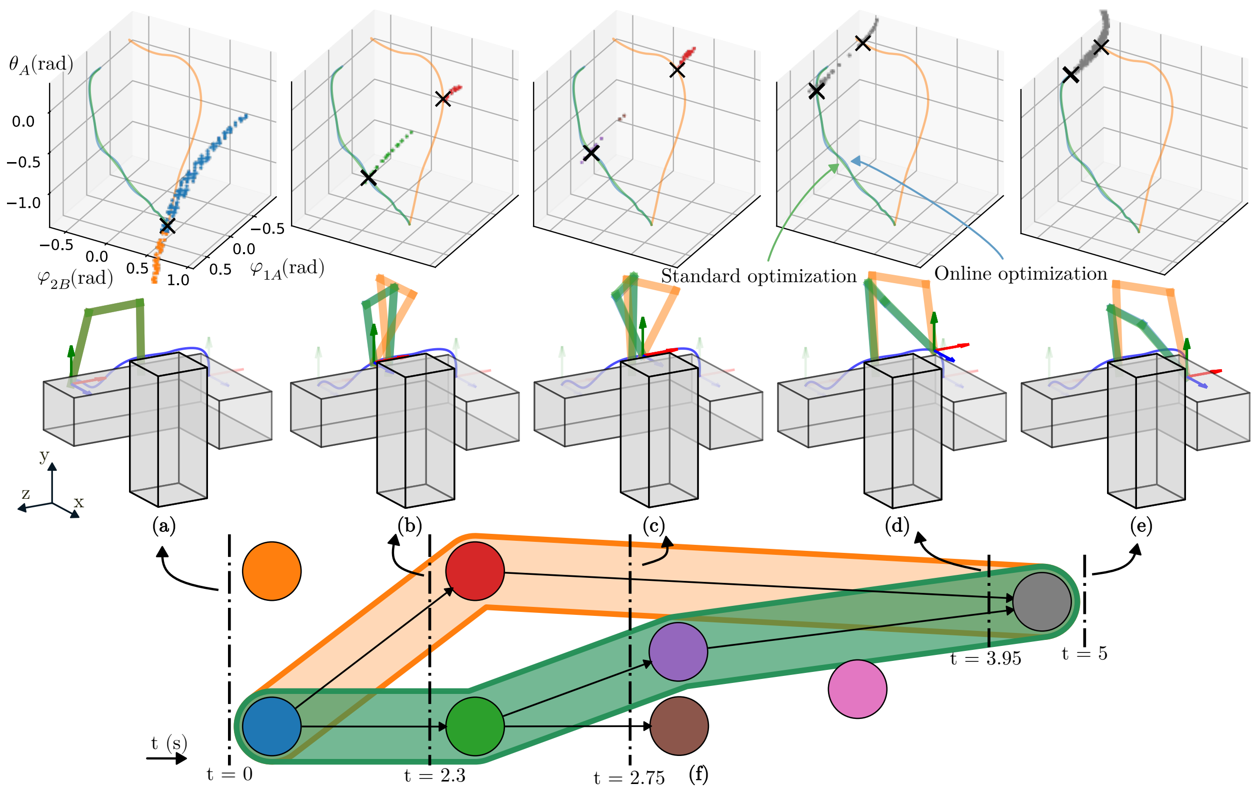

Figure 8: (a-e) Snapshots of the simulation. (f) C-bundle graph $\mathcal{G}$ and every possible c-bundle chain.

Table 5: Computational times and costs of the generated trajectory for the RPRRRRPR robot.

Graph generation	**3917 ms**
Sampling	*2957 ms*
Clustering	*896 ms*
Matching	*64 ms*
Generation of raw trajectories	**16 ms**
C-bundle chains calculation	*1 ms*
Raw trajectory	*15 ms*
Online optimization (average time per iteration)	**26 ms**
Standard optimization	*6043 ms*
Trajectory cost (online optimization)	**0.119**
Trajectory cost (standard optimization)	*0.111*

involving robots with higher DoF. As outlined in Section 4.1, the process of sweeping or scanning through every possible combination of r joint coordinates corresponds to a number of scans equal to the binomial coefficient $\binom{n}{r}$. Consequently, the number of combinations scales exponentially with the increase in DoF n. A method lacking efficiency would fail to generate the SMD for high DoF robots within a reasonable timeframe. In the case at hand, the number of combinations corresponds to $\binom{8}{2} = 28$. Resolving the SMD for an 8-DoF robot and an $r = 2$ redundancy degree in the same order of magnitude as the 3-DoF robot with redundancy $r = 1$ is a testament to the efficiency of the proposed method.

Nevertheless, some of such combinations of joint coordinates render the IKP unsolvable due to the fact that, when fixing the task $\mathbf{x}$, some joint coordinates become dependent in the kinematic equations of the robot. Specifically, the RPRRRRPR robot has 3 combinations that are unsolvable, yielding a total of 25 solvable combinations. When solving some of these 25 combinations using algebraic elimination methods, it is necessary to find the roots of polynomials of degrees higher than 4, which can increase the computational time of the graph generation stage. To mitigate this issue, these combinations were omitted from the SMD computation stage, resulting in a less densely populated SMD. However, thanks to the optimization stage, which does not rely on the previously sampled points of the SMD, the quality of the final trajectory is not affected by this omission. This is because, as we will demonstrate in Section 6.1, the paths converge to the same optimal solution independently of the density of the SMD or the initial raw trajectories. Specifically, in the results we have shown, we have omitted 10 combinations for solving the IKP, significantly reducing the computational time of the graph generation stage, while maintaining the quality of the final trajectory. A video of the simulation with every possible combination is available in the supplementary material as *ex2_ rconst_ fullparams.mp4*, replicating the simulation results presented in Fig. 8, where it can be observed that the SMD and extracted optimal joint trajectories are not affected by the omitted combinations. The only difference corresponds to a couple of very small c-bundles that are created and instantly vanish, but these can be considered noise, and do not affect the final result.

Another experiment was conducted in order to simulate a concave transition during the exploration of a vertical structure that the robot must climb. The starting pose of the robot remains the same as in the previous experiment, while the end pose corresponds to a pose located at a vertical beam that the robot must adhere to. The results are presented in the supplementary material in the form of an animation of the robot configuration and the extracted optimal joint trajectory (*ex2_ cnctrans.mp4*).

5.3. Example 3: RPR planar manipulator

This brief section introduces an additional experiment designed to demonstrate the exhaustive nature of the proposed method for global redundancy resolution. In scenarios with substantial constraints, such as those presented in the preceding experiments, the resultant graph is often sparse. This sparsity is due to the limited number of c-bundle chains that successfully complete the task, as the constraints imposed by obstacles and joint limits significantly reduce the feasible joint configurations, leading to the elimination of many previously feasible c-bundles. Conversely, in scenarios with less-restrictive constraints, the graph is denser, increasing the number of c-bundles in the SMD. This experiment aims to highlight the exhaustive capabilities of the proposed method, demonstrating its competence in identifying and managing every c-bundle and co-regular surface that arises from the task constraints.

The experiment consists of an RPR (Revolute-Prismatic-Revolute) planar manipulator (Fig. 9(a)) tasked with following a one-dimensional end-effector trajectory. The task trajectory is defined as a parabolic path of the y coordinate of the end-effector, following the equation $y = -6.662t^2 + 8.162t - 1.5$, to be completed in 1 second ($t \in [0, 1]$). This results in degree of redundancy $r = 2$, i.e., SMMs are surfaces. To enhance the complexity and realism of the task, the end-effector is required to avoid an elliptic obstacle in the workspace, defined by the inequality $(x - 1.1)^2/1^2 + (y + 0.2)^2/0.25^2 \leq 1$. Every position coordinate is given in meters, relative to the base of the robot. The joint limits are set to $[-2\pi, 2\pi]$ radians for the revolute joints, and $[0, 0.5]$ meters for the prismatic joint.

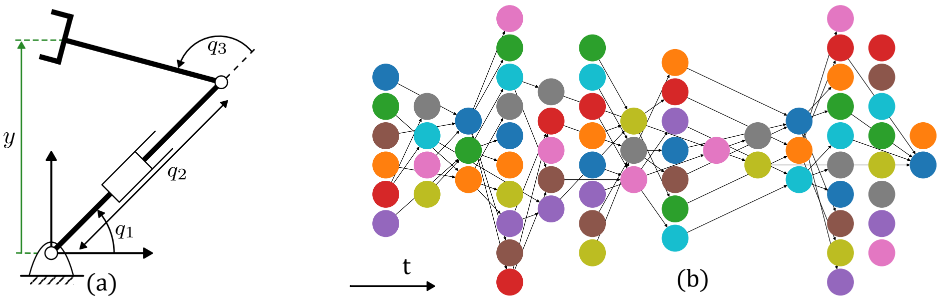

Figure 9: (a) RPR planar manipulator. (b) Generated graph.

The resulting graph is presented in Fig. 9(b). The graph is significantly denser than the previous examples, as the constraints imposed by the task are less restrictive. The graph is composed of 72 nodes, each corresponding to a c-bundle, and 68 edges, which represent the connections between the c-bundles.

The rest of the simulation results are omitted, as the focus of this experiment is the graph itself, which demonstrates the capability of our method to correctly handle SMDs with highly dense and intricate c-bundle connections.

In addition, we have conducted a simulation experiment to assess the effectiveness of the vectorization of the computation of the SMD (Section 4.1). When employing Algorithm 2 sequentially to compute the SMMs at $y = -1.5$ meters, the runtime is 82.909 milliseconds. When vectorizing the inner loop of the algorithm, the runtime is reduced to 1.675 milliseconds, which results in a speedup of 49.5 times. When extending this experiment to the entire SMD for the same task as in Fig. 9, the runtime is reduced from 10.614 seconds to 0.194 seconds, which results in a speedup of 54.7 times. From this, we can estimate that the vectorization of the SMD computation results in a speedup of approximately 50 times in this example.

6. Discussion

This section presents a discussion of some of the key aspects of the proposed framework, including its global optimality, complexity and dimensionality analysis, and comparisons with state-of-the-art methods.

6.1. Global optimality

In the context of redundancy resolution, the term *global optimality* refers to the capability of the method to identify the joint trajectory $\mathbf{q}(t)$ for which a given cost function results in the minimum value, from the infinite number of solutions that satisfy a given task trajectory and constraints.

In the proposed framework, we characterize the infinity of solutions by generating the SMD. Then, we identify the c-bundle chains that successfully complete the task in order to extract a joint trajectory for each chain. Considering that the SMD will certainly contain the globally optimal joint trajectory (since it characterizes the infinite set of possible solutions), and that we extract every possible c-bundle chain from the SMD, one should expect that the globally optimal joint trajectory will coincide with the optimal trajectory in one of the extracted c-bundle chains. While we are not able to provide a formal proof of the global optimality of the proposed method, we can offer a statistical demonstration of the optimality of the extracted joint trajectories.

Consider the experiment with the 3R planar manipulator presented in Section 5.1, where we extracted four different optimal joint trajectories, one for each c-bundle chain. To demonstrate optimized trajectories converge to the optimal solutions for each c-bundle chain, we conducted an additional experiment. Instead of generating the raw joint trajectory by any of the introduced methods, we generated completely random raw joint trajectories within each extracted c-bundle chain by modifying line 6 of Algorithm 4 by randomly selecting a point in $\mathbf{MP}_i$. This slight modification produces completely random raw joint trajectories, which are then optimized to compare the costs of the resulting trajectories. We conducted 2350 iterations of this experiment, and the histograms of the cost function values of the resulting trajectories for each c-bundle chain are presented in Fig. 10.

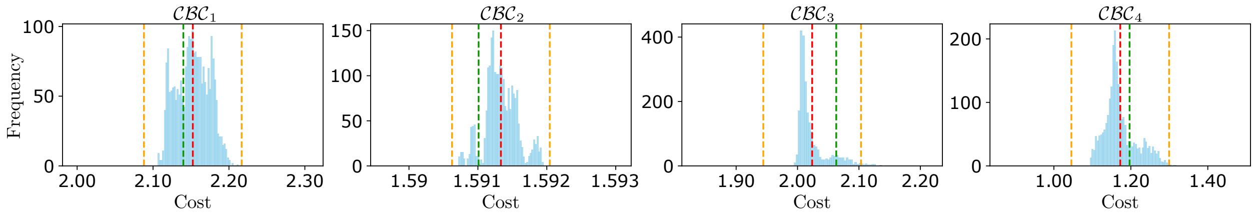

Figure 10: Global optimality analysis.

The mean cost function value is plotted as a vertical red line, while two orange lines represent three standard deviations from the mean. The results shown in Table 4 correspond to the vertical green lines. It can be observed that every optimized trajectory for each c-bundle chain converges to the same cost value within a small deviation, and that no significant outliers are present. The slight variations in the cost function values are due to the value selected for the convergence parameter ε in the optimization stage (Algorithm 5). We have selected a large-enough value in order to be able to visualize such distribution. This experiment indicates that the extracted joint trajectories converge to the optimal solutions for each c-bundle chain, which in turn demonstrates that the globally optimal joint trajectory is among the extracted trajectories. While this statistical demonstration does not provide a formal proof of the global optimality of the proposed method, it does offer a strong indication.

6.2. Complexity and dimensionality analysis

It is beneficial to analyse the complexity of the proposed algorithms. The first stage of our framework is the computation of the SMD, which is the most computationally intensive part of the process. In Algorithm 2, every combination of r of the n joint coordinates is swept, producing a number of sweeps equal to the binomial coefficient $\binom{n}{r}$. For each of these combinations, N^r points are sampled (where N is the number of points in which every joint coordinate is discretized), and for each point, $\mathbf{q}$ is solved from Eq. (1), obtaining a maximum of a different solutions. If we consider the worst-case scenario over all combinations for generating the SMD, the computational complexity of the sampling-based method is $\mathcal{O}(k\binom{n}{r}N^r a)$, where k is the number of points in which the task trajectory is discretized. The same complexity is present in the NN-based raw trajectory generation stage, as the complexity of Algorithm 4 is dictated by the nearest-neighbour search, whose worst-case complexity equals the maximum possible number of points in the SMD: $\mathcal{O}(k\binom{n}{r}N^r a)$. If the hybrid version of the raw trajectory generation stage is employed, the complexity is that of the employed PA, multiplied by the number of times it is executed, which equals to the number of c-bundles in the c-bundle chain.

It is clear that the computational complexity of the entire framework is dictated by, and grows exponentially with the redundancy degree r, and factorially with the increase in DoF n. In fact, few studies exist that address the global redundancy resolution problem for robots with a redundancy degree greater equal or greater than 2, as the curse of dimensionality renders the problem of identifying the global set of infinite solutions (e.g., SMMs or FMs) intractable in a reasonable timeframe. State-of-the-art methods for the generation of SMMs take from hundreds [28] ($r = 1$) to thousands of seconds [31] ($r = 2$) to generate the SMMs at a single task point. We have reduced the computational time of the SMM generation process to the order of milliseconds, even for robots with a redundancy degree of $r = 2$, as demonstrated in the experiments of Section 5.2, by using the proposed sampling-based method described in Section 4.1. Moreover, when working with robots with complex kinematic chains, whose kinematic equations cannot be solved by algebraic elimination, we have proposed a continuation-based method that is capable of generating the SMMs for $r = 1$ in the same order of magnitude of time as the sampling-based method, and we tested that, for example, for the RPRRRRPR robot experiment of Section 5.2, it is able to generate SMMs in about 1 second.

The computational optimization strategies that we have discussed throughout this work, such as vectorization of the SMD computation, efficient data structures like k-d trees, or parallelization among available CPU cores, are essential to implement the framework on resource-constrained robotic control systems.

However, for the moment, $r = 3$ seems to be out of reach for every method aspiring to completely map three-dimensional (or higher-dimensional) self-motion manifolds. A possible suboptimal approach to tackle $r \geq 3$ problems would be to virtually remove redundancy levels by fixing joints, and then solving the IKP for the remaining $r = 2$ redundancy degree. The same strategy could be applied for sudden joint failures where a joint becomes locked, as the already-computed SMD could be intersected with a hyperplane defined by $q_j = c$, where q_j is the failed joint, and c is the value at which it has been locked. This would avoid the recomputation of the SMD, but subsequent stages of obtaining the graph and solving trajectories would still be needed to adapt the solution to the new constraints.

6.3. State-of-the-art comparison

Out of the state-of-the-art methods for global redundancy resolution reviewed in Section 2, few are directly comparable to the proposed method, since they usually study the redundancy for the whole workspace, or also plan the task trajectory. Ferrentino and Chiacchio [19], however, do solve the global redundancy resolution problem for a given task trajectory. In addition, the authors of [23] sketch, as a supplementary tool to their main contribution, an algorithm that builds the SMD and establishes paths across it. Since no publicly available repositories or benchmarks exist for a direct comparison, we have made every effort to replicate the methods described in these two studies faithfully. We replicated the scenario of the 3R planar manipulator of Section 5.1 with our implementation of the methods proposed in [19] and [23].

In [19], the authors resolve the redundancy by means of feasibility maps instead of SMMs. When applying the method of [19] to our example of Section 5.1, we chose q_1 as the redundant variable, and discretized its range of values $[-2\pi, 2\pi]$ into $N_u = 144$ points, which provides a resolution comparable to the one that we used in Section 5.1, whereas the time axis is discretized into $N_i = 100$ points, as in Section 5.1. This sets a scenario that is as fair as possible for comparison purposes. For the cost function, we adapted Eq. (2) to align with the formulation in [19], and set their function l as the norm of the difference between each pair of tested joint configurations $\mathbf{q}$ (i.e., $l = \|\mathbf{q}(t_1) - \mathbf{q}(t_2)\|$). The obtained solution is given in Fig. 11, where the FMs are plotted in the top row. Purple regions indicate areas where configurations lead to complex solutions to the IKP, i.e., when solving q_2 and q_3 for each value of q_1 and $\mathbf{x}(t)$ (there are two possible solutions for q_2 and q_3 for each pair (t, q_1), that is why there are two FMs in the first row of Fig. 11). Red regions represent collisions with the elliptical obstacles, Finally, uncoulored regions denote feasible configurations. The bottom-left subfigure of Fig. 11 shows the cost map of the FM that leads to the optimal solution, and is represented using the *viridis* colourmap, with dark purple indicating the lowest cost and yellow the highest. The optimal joint trajectories are plotted in the bottom-right part of Fig. 11, where the time history values of $q_1(t), q_2(t)$, and $q_3(t)$ obtained by our implementation of [19]'s algorithm are plotted in solid lines, while the solution provided by our method is plotted in dashed lines. The optimal trajectory of the redundant variable $q_1(t)$ is also plotted in the top-left FM and the bottom-left cost map.

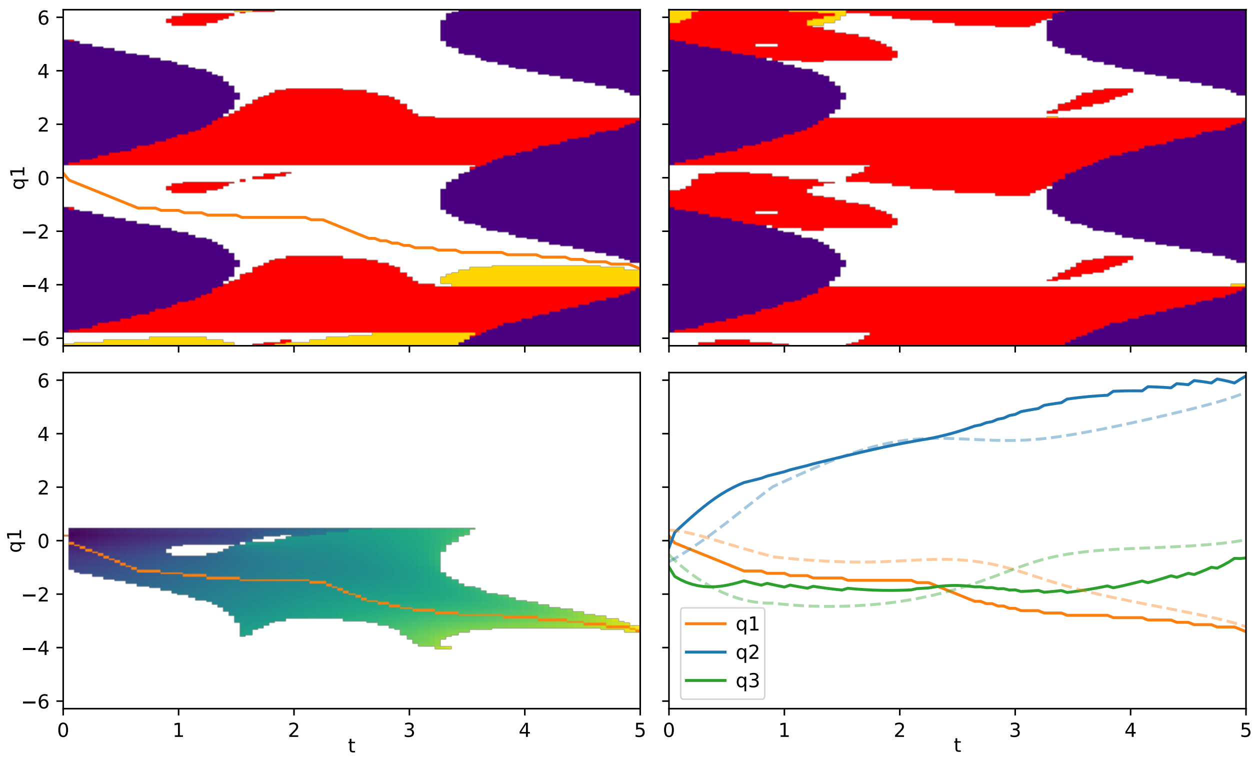

Figure 11: Results of the experiment with the 3R planar manipulator using the method of Ferrentino and Chiacchio [19]. The top row shows the two feasibility maps, while the bottom row shows the cost map (left) and the output joint trajectories (right).

Computing both FMs for the given task and constraints took an average of 6.474 seconds over 100 tests. Running our implementation of the algorithm described in Fig. 14 of [19] in the obtained FMs yielded an average runtime of 10.083 seconds. When applying Eq. (2) to the obtained solution, it resulted in a cost value of 1.44, which is comparable to the one returned by our method (1.195), if slightly larger. In fact, it can be visualized in the joint trajectories of the bottom-right subplot of Fig. 11 that both solutions share the same tendency, and approach the same optimal solution, as any small differences are likely attributable to resolution variations, implementation differences, or grid discretization. In terms of computational efficiency, both methods appear to share similar performances, as they are able to return the optimal solution in the same timeframe range, namely, around 16 seconds. For the FM method, these 16 seconds are the sum of the time required to generate the FMs (6.474 s) and to explore the FMs for the optimal solution (10.083 s). For our proposed framework, these 16 seconds are the sum of all times recorded in Table 4, counting SMD generation via continuation (2.1 s), generation of all raw joint trajectories (negligible), and their optimization (4.1 + 2.5 + 3.7 + 4 s), assuming that all trajectories are optimized sequentially on a single core. However, if we consider that our optimization stage is parallelized, the runtime of our method is lowered to around 6 seconds, corresponding to the sum of SMD generation (2.1 s) and optimization of the most demanding raw trajectory (4.1 s). Modifying the resolution N_u of our implementation of the method based on FMs in order to match these 6 seconds, it is able to provide a solution of cost 2.11 for similar timeframes.

In any case, while performance-wise both approaches seem similar, there exist some clear advantages of using the SMD for solving the redundancy resolution problem over FMs. Firstly, a notable distinction appears in how transitions between FMs are handled. [19] evaluates the possibility of performing transitions between FMs associated with different extended aspects, so that every possible feasible trajectory can be explored and potentially better solutions are not missed. Extended aspects are different solutions of the IKP when one solves the joint coordinates from the kinematic equations of the robot for known values of the task coordinates and redundant variables. In fact, FMs associated to different extended aspects are different branches of the projection of the SMD on a hyperplane $(t, \mathbf{q}_r)$, where t is time and $\mathbf{q}_r$ are r redundant coordinates (where r is the degree of redundancy). For example: imagine a SMD that is a unit sphere $t^2+q_1^2+q_2^2=1$, where t is time $(-1<t<1)$ and (q_1, q_2) are joint coordinates. If we pick q_1 as the redundant coordinate $(\mathbf{q}_r = q_1)$ and project this sphere on plane (t, q_1), we get two FMs associated to two different extended aspects: one projection due to one hemisphere $(q_2 > 0)$ and another projection due to the other hemisphere $(q_2 < 0)$. In the method proposed in [19], one would explore the plane (t, q_1) for feasible trajectories and, in order to avoid missing potentially better solutions, it would require to explore both projections of this sphere, which can be connected at the projection of the equator, which are singularities of the projection. As it can be observed in Fig. 19 of ref. [19], these transitions through singularities of the projection may produce small artificial spikes in the obtained trajectories, which should actually be smooth transitions because the robot is not going through kinematic singularities, only singularities of the projection of the SMD on the FMs. In contrast, our framework naturally handles transitions between extended aspects by considering the entire SMD, eliminating discontinuities and ensuring smoother joint trajectories. This is a direct result of our approach, which considers the global set of inverse solutions holistically, rather than partitioning them into projections based on extended aspects. Returning to the illustrative example of the sphere, our would method searches for optimal trajectories by moving directly on the sphere SMD, instead of working with its projections.

More importantly, even if we omit or somehow correct these spikes in the output trajectories, the method based on FMs is not robust to certain situations, as it may not find a solution even if one exists. For instance, take the same task and robot, but remove every obstacle, and simply constraint joint limits to $[-3, 3]$ for q_1 and $[-\pi, -0.5]$ for q_2 (q_3 remains unbounded). This produces the FMs depicted in Fig. 12(a-b), where the FM1 of Fig. 12(a) consists of three connected components (A, B, C), whereas the FM2 of Fig. 12(b) consists of a single connected component. *Transition maps* [42] are plotted over these FMs following a colourmap, where dark blue indicates lower values (see Fig. 12(d)). These values correspond to the angular distance between the two joint configurations $\mathbf{q}$ that correspond to each pair (t, q_1) (recall that each pair (t, q_1) has two solutions for (q_2, q_3), which produce two different $\mathbf{q}$). Values close to 0 (dark blue) indicate that transitions between extended aspects (and FMs) are possible through those points, which are singularities of the projection of the SMD onto plane (t, q_1). We will refer to these 0-distance configurations as *gateways* between FMs. Note that each FM has been represented with a different colourmap in order to differentiate them when plotting the entire SMD (Fig. 12(c)), but they share the same dark blue colouring for angular distance values close to 0 (see how the lowest values of both colourmaps are similar in Fig. 12(d)).

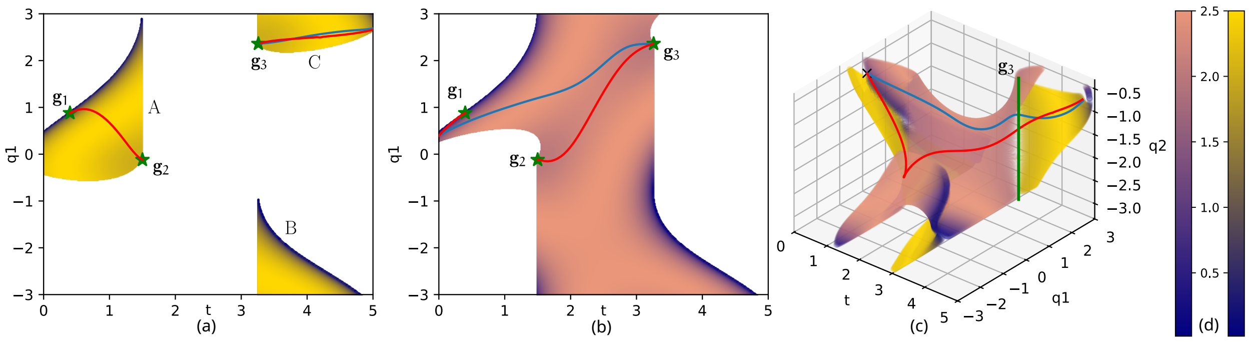

Figure 12: (a): FM1. (b): FM2. (c): SMD projected onto the subspace (t, q_1, q_2). (d): Colourmaps employed to plot the transition maps.

If we rely on this angular distance to determine possible transitions between FMs, it would appear that no solution can be found when planning the trajectories using feasibility maps (but a trajectory is indeed possible, as shown by the blue and red trajectories, which were obtained by our proposed method). This happens because, if we start at $t = 0$ in the FM2 shown in Fig. 12(b), the only regions with a distance close to 0 (those that allow for a transition from FM2 to FM1) correspond to gateways that lead to components A and B of FM1, which impedes completion of the task since those components do not reach $t = 5$. On the contrary, if we were able to transition into component C of FM1, the trajectory would be completed. Nevertheless, this desired transition seems not possible when relying on angular distance to identify the mentioned gateways, as there do not exist points with values close to 0 (dark blue) that join FM2 with component C of FM1.

However, as shown in Fig. 12(c), when visualizing the SMD projected onto the subspace (t, q_1, q_2) (q_3 is omitted), we clearly see that the said transition is indeed possible through a vertical line of gateway points $\mathbf{g}_3$ that is represented as a green segment, therefore it is a singularity of the projection. This problem of FMs not detecting this gateway $\mathbf{g}_3$ arises because this vertical segment of singularities, which allows for the transition between FMs (both plotted in different colourmaps), projects onto a single point in the (t, q_1) plane (represented as the rightmost green stars $\mathbf{g}_3$ in Fig. 12(a-b)), and this point will be missed with probability one when discretizing the (t, q_1) plane into a grid for computing the FM and transition maps. In other words, detecting this singularity while sweeping (t, q_1) is nearly impossible, as it would require that one of the points of the discretization grid coincides exactly with this singularity, which is extremely unlikely. That is the reason why attempting to find a feasible trajectory by means of FMs would fail in situations like this, because, as Fig. 12 shows, the colourmaps used to indicate angular distances do not suggest small values near this special singularity, unlike for other gateways marked in dark blue.

If we run our proposed framework for the same scenario, we obtain two trajectories that are plotted in blue and red in Fig. 12. The blue trajectory corresponds to the optimal solution, and the red trajectory corresponds to a suboptimal solution that suffers a complete wrapping in angle q_3. It can be observed how feasible joint trajectories that complete the task are actually possible, and perform the desired transition between extended aspects through the otherwise-missed gateway $\mathbf{g}_3$. The blue trajectory completes the task with a single transition between extended aspects through the said gateway $\mathbf{g}_3$, while the blue trajectory performs three transitions: the first one through the FM-detected gateway $\mathbf{g}_1$ that joins component A of FM1 with FM2, and the other two through FM-missed gateways $\mathbf{g}_2$ and $\mathbf{g}_3$. A supplementary video material (*ex_missed_gateways.mp4*) shows this globally optimal solution in motion (blue configuration), together with the secondary suboptimal solution detected by our method (in red). It also shows the SMD, and the animated sequence of SMMs, where the returned trajectories are plotted. Additionally, the c-bundle graph from which our method derives the feasible c-bundle chains is also represented. Note that both solutions correspond to the c-bundle chains formed by the succession of c-bundles coloured blue-purple and blue-red-lightblue in the said supplementary video, for the optimal and suboptimal solutions, respectively.

Another issue that arises is the sensitivity to the resolution (i.e., size of the discretization step) used to compute the FM. With [19]'s algorithm, it has been observed, both in the original paper and in our experiments, that the cost of the solution strongly depends on the resolution of the FM. In our case, while a fine-enough resolution is still required not to misinterpret disjoint SMMs as connected, the cost of the solution is not affected by the resolution of the SMD, as the optimization stage does not rely on the sampled points of the SMD, and will converge to the same solution independently of the raw trajectories produced by the sampled points of the SMD, as demonstrated in Section 6.1.

In addition to these advantages, it should be noted that the demonstration of the method of [19] was limited to examples with $r = 1$ degree of redundancy, whereas our examples in Sections 5.2 and 5.3 have demonstrated the effectiveness of our proposed framework for the more difficult case of $r = 2$ degrees of redundancy, which few research papers address.

Besides FMs, another method that is similar to our proposed framework is the one presented in [23], which proposes an algorithm to generate the SMD and establish paths across it, which we have replicated for comparison. Since the algorithm is probabilistically complete (i.e., given enough time it would densely generate the whole SMD), we had to stop it after some time running, which we set to a comparable time to what our method requires when parallelization at the optimization stage is not performed (around 16.5 seconds). The results are presented in Fig. 13(b). The sampled points of the SMD are plotted in blue, while the segments that represent connections between them are shown in green. We also plotted in orange the SMD obtained via continuation in the experiment of Section 5.1 for comparison. The optimal trajectory, whose median cost is 1.793 (versus our optimal cost of 1.195), is plotted in red. When parallelization is implemented at the optimization stage, our framework returns the optimized final trajectories in around 6 seconds. If [23]'s algorithm is stopped when reaching this runtime, the median cost of their solutions is raised to 1.967. An example solution for this configuration is shown in Fig. 13(a). While [23]'s algorithm for SMD generation was functional, its efficiency was significantly lower (particularly during the final stage, which involves connecting sampled points to form a complete path). For example, we run it again, but instead of stopping it when it has taken the same computation time as our method, we stop it when it achieves the same optimal cost as our method. In that case, it took an average of 40 seconds of runtime to return a solution of the same cost as our method, obtaining the result shown in Fig. 13(c). Also, we have encountered that the variance of the solution is quite high for different runs using the same parameters. For instance, it can be observed in Fig. 13(b), that the algorithm detects that an isolated region of SMD points (in the bottom-left part of the subfigure) is reachable, while in the other examples it does not. This may be due to the fact that this algorithm for SMD generation was not the primary focus of [23]'s work, but rather a supplementary tool supporting their main contribution: a continuous pseudoinversion of the forward kinematic map.

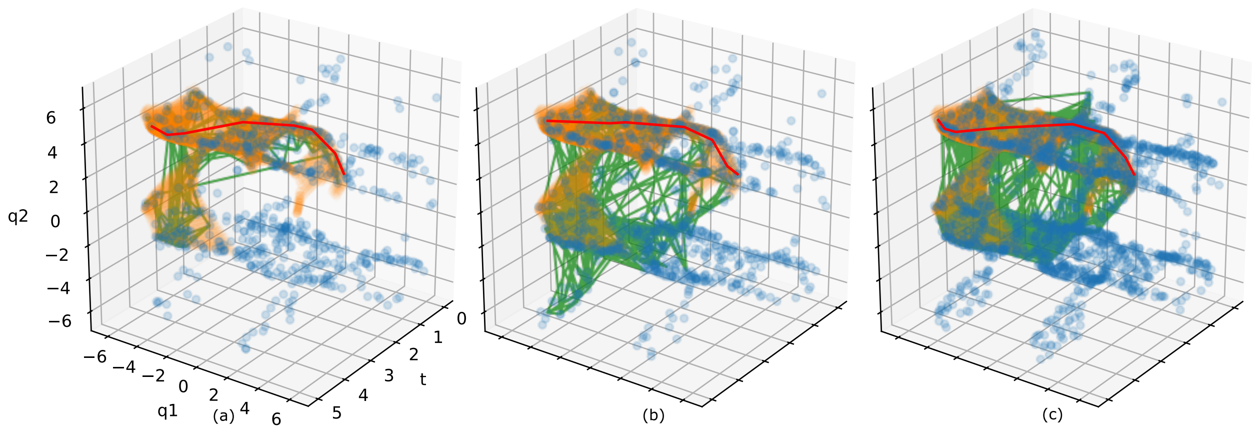

Figure 13: Results of the method of Hauser and Emmons [23]for SMD generation, applied to the scenario of Section 5.1. (a): results of applying [23] taking the same computation time as our method when it parallelizes the optimization of raw trajectories. (b): results of applying [23] taking the same computation time as our method without parallelization. (c): results of applying [23] until obtaining the same optimal cost as our method.

7. Conclusions and future work

In this paper, we have introduced a novel framework for global redundancy resolution. It is capable of determining the globally optimal utilization of additional degrees of freedom of a robot to execute a task. The framework minimizes a cost function while subject to kinematic constraints, such as joint limits and obstacles. It is composed of three stages: the generation of the SMD, the extraction of raw joint trajectories, and the optimization of said trajectories.

The proposed method is based on the concept of self-motion manifolds to compute the self-motion domain, which characterizes the infinite number of solutions to the inverse kinematics problem that satisfy the imposed constraints along a task trajectory. The self-motion domain is then spatially analysed to generate a graph whose nodes, the c-bundles, encapsulate self-motion manifolds that share the same topology. The graph is subsequently traversed to identify preliminary joint trajectories that successfully complete the task. Finally, those joint trajectories are optimized to minimize a cost function.

The proposed framework is flexible, allowing for the selection of different variants for each stage, depending on the application requirements in terms of computational efficiency, optimality, complexity of the kinematic equations, or number of degrees of redundancy. Regarding the first stage of the algorithm, the SMD is advised to be generated using the proposed sampling method when the degree of redundancy is $r = 1$ if there are no collision constraints, or when $r > 1$ in any case (with or without collisions). On the contrary, the presented continuation-based method is more efficient for tracing 1-dimensional SMM when there exist collision constraints that severely reduce the length of the SMM to trace via continuation because some parts of these SMM invade regions of the joint space that produce collisions. The proposed continuation variant to generate the SMD is also suggested when the robot has complex kinematic chains that make Eq. (1) difficult to solve by algebraic elimination as the sampling method requires. The selection for the second stage of the algorithm mainly depends on the desired computational efficiency. If real-time performance is required, the NN-based method is recommended for searching for raw trajectories, as it is the fastest. Otherwise, if the globally optimal solution is desired, the hybrid NN-PA version of the method is advised, since this will produce raw trajectories that are more optimal and the last optimization stage of the proposed framework will converge faster to the globally optimal solution. A similar trade-off is present in the third stage of the algorithm, where the online optimization is recommended for real-time applications where a decent (yet not necessarily globally optimal) feasible trajectory is sought, while the standard optimization is advised when searching for the globally optimal solution.

The proposed method has been evaluated through a series of simulations, which demonstrate the performance of the method in different scenarios, combining different variants of the framework, degrees of redundancy, degrees of freedom, and task constraints. However, for solving problems with a redundancy degree greater than 2, the processing time exceeds the order of magnitude that can be considered acceptable for real-time applications.

Computing SMMs employing the continuation variant we propose in Section 4.4.1 can be computationally expensive, especially for degrees of redundancy higher than 1. In fact, when our framework has to rely on continuation for SMM generation, it is mainly because attempting to solve the kinematic constraints that define the SMMs via algebraic elimination (as in Section 4.1) may become too complicated due to the complexity of the kinematic chains of the robot. If continuation becomes too computationally expensive, and pure algebraic elimination is intractable, we suggest hybrid approaches like the algebraic-numerical method based on constraint curves that we proposed in [43]. Currently, we are working on extensions of these hybrid methods to address limitations of purely algebraic methods and continuation methods.

Additional future work will focus on the development of efficient methods for the generation of self-motion manifolds for degrees of redundancy greater than 2. Also, an efficient method for the spatial analysis of the c-bundles that compose the SMD, which currently is done by the clustering and matching stages, would be beneficial for the computational efficiency of the graph generation stage, as these two stages take an important fraction of the overall time of the method. Real experiments with physical robots could also be conducted to validate the proposed framework in a real-world scenario.

Acknowledgements

The authors would like to thank Dr. Josep M. Porta for his insight on the implementation of the higher-dimensional continuation method for the generation of manifolds. This work was supported by the MCIN/AEI/ 10.13039/501100011033 and the ESF+ [PID2020-116418RB-I00, PRE2021-099226].

References

[1] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Force control, Springer, 2009.

[2] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, The international journal of robotics research 5 (1) (1986) 90–98.

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

[4] A. Liegeois, et al., Automatic supervisory control of the configuration and behavior of multibody mechanisms, IEEE transactions on systems, man, and cybernetics 7 (12) (1977) 868–871.

[5] S. B. Slotine, B. Siciliano, A general framework for managing multiple tasks in highly redundant robotic systems, in: proceeding of 5th International Conference on Advanced Robotics, Vol. 2, 1991, pp. 1211–1216.

[6] F. Flacco, A. De Luca, O. Khatib, Control of redundant robots under hard joint constraints: Saturation in the null space, IEEE Transactions on Robotics 31 (3) (2015) 637–654.

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

[8] M. Faroni, M. Beschi, N. Pedrocchi, A. Visioli, Predictive inverse kinematics for redundant manipulators with task scaling and kinematic constraints, IEEE Transactions on Robotics 35 (1) (2018) 278–285.

[9] A. Mavrommati, C. Osorio, R. G. Valenti, A. Rajhans, P. J. Mosterman, An application of model predictive control to reactive motion planning of robot manipulators, in: 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), IEEE, 2021, pp. 915–920.

[10] N. Ratliff, M. Zucker, J. A. Bagnell, S. Srinivasa, Chomp: Gradient optimization techniques for efficient motion planning, in: 2009 IEEE international conference on robotics and automation, IEEE, 2009, pp. 489–494.

[11] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, S. Schaal, Stomp: Stochastic trajectory optimization for motion planning, in: 2011 IEEE international conference on robotics and automation, IEEE, 2011, pp. 4569–4574.

[12] A. Albu-Schäffer, A. Sachtler, Redundancy resolution at position level, IEEE Transactions on Robotics (2023).

[13] A. Peidro, O. Reinoso, A. Gil, J. M. Marín, L. Paya, A method based on the vanishing of self-motion manifolds to determine the collision-free workspace of redundant robots, Mechanism and Machine Theory 128 (2018) 84–109.

[14] J. W. Burdick, 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, Springer, 1989, pp. 25–34.

[15] C. L. Lück, Self-motion representation and global path planning optimization for redundant manipulators through topology-based discretization, Journal of Intelligent and Robotic Systems 19 (1997) 23–38.

[16] Z. Zhou, J. Zhao, Z. Zhang, X. Li, Motion planning method of redundant dual-chain manipulator with multiple constraints, Journal of Intelligent & Robotic Systems 108 (4) (2023) 69.

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

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

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

[20] E. J. Haug, Redundant serial manipulator inverse position kinematics and dynamics, Journal of Mechanisms and Robotics 16 (8) (2024) 081008.

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

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

[23] K. Hauser, S. Emmons, Global redundancy resolution via continuous pseudoinversion of the forward kinematic map, IEEE Transactions on Automation Science and Engineering 15 (3) (2018) 932–944.

[24] S. LaValle, Rapidly-exploring random trees: A new tool for path planning, Research Report 9811 (1998).

[25] D. Berenson, S. S. Srinivasa, D. Ferguson, J. J. Kuffner, Manipulation planning on constraint manifolds, in: 2009 IEEE international conference on robotics and automation, IEEE, 2009, pp. 625–632.

[26] L. Jaillet, J. M. Porta, Path planning under kinematic constraints by rapidly exploring manifolds, IEEE Transactions on Robotics 29 (1) (2012) 105–117.

[27] L. Jaillet, J. M. Porta, Asymptotically-Optimal Path Planning on Manifolds, in: Robotics: Science and Systems VIII, The MIT Press, 2013, pp. 145–152.

[28] Y. Yang, Y. Wu, J. Pan, An interval branch-and-bound-based inverse kinematics algorithm towards global optimal redundancy resolution, arXiv preprint arXiv:2104.12183 (2021).

[29] M. E. Henderson, Multiple parameter continuation: Computing implicitly defined k-manifolds, International Journal of Bifurcation and Chaos 12 (03) (2002) 451–476.

[30] D. DeMers, K. Kreutz-Delgado, Canonically parameterized families of inverse kinematic functions for redundant manipulators, in: Proceedings of the 1994 IEEE International Conference on Robotics and Automation, IEEE, 1994, pp. 1881–1886.

[31] T. Wu, J. Zhao, B. Xie, A novel method for computing self-motion manifolds, Mechanism and Machine Theory 179 (2023) 105121.

[32] I. Banfield, H. Rodríguez, Generation of the self-motion manifolds of a functionally redundant robot using multi-objective optimization, in: Robotics for Sustainable Future: CLAWAR 2021 24, Springer, 2022, pp. 438–452.

[33] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: kdd, Vol. 96, 1996, pp. 226–231.

[34] R. Sedgewick, Algorithms in c, part 5: graph algorithms, 3rd Edition, Addison-Wesley Professional, 2001.

[35] S. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.

[36] A. Bambade, S. El-Kazdadi, A. Taylor, J. Carpentier, Prox-qp: Yet another quadratic programming solver for robotics and beyond, in: RSS 2022-Robotics: Science and Systems, 2022.

[37] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, S. Boyd, Osqp: An operator splitting solver for quadratic programs, Mathematical Programming Computation 12 (4) (2020) 637–672.

[38] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, P. Abbeel, Finding locally optimal, collision-free trajectories with sequential convex optimization., in: Robotics: science and systems, Vol. 9, Berlin, Germany, 2013, pp. 1–10.

[39] J. Pan, S. Chitta, D. Manocha, Fcl: A general purpose library for collision and proximity queries, in: 2012 IEEE International Conference on Robotics and Automation, IEEE, 2012, pp. 3859–3866.

[40] A. Peidró, A. Gil, J. M. Marín, Y. Berenguer, L. Payá, O. Reinoso, Monte-carlo workspace calculation of a serial-parallel biped robot, in: Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Volume 2, Springer, 2016, pp. 157–169.

[41] A. Peidró, M. Fabregat-Jaén, A. Gil, D. Valiente, Ó. Reinoso, Teaching redundancy resolution in redundant parallel manipulators with an interactive and graphical simulation, in: INTED2024 Proceedings, IATED, 2024, pp. 2677–2686.

[42] E. Ferrentino, P. Chiacchio, A topological approach to globally-optimal redundancy resolution with dynamic programming, in: ROMANSY 22–Robot Design, Dynamics and Control: Proceedings of the 22nd CISM IFToMM Symposium, June 25-28, 2018, Rennes, France, Springer, 2018, pp. 77–85.

[43] A. Peidró, L. Payá, S. Cebollada, V. Román, Ó. Reinoso, Solution of the forward kinematics of parallel robots based on constraint curves, in: International Conference on Informatics in Control, Automation and Robotics, Springer, 2020, pp. 386–409.