Fibration Trees: A Unified Approach to Multi-Robot Motion Planning

International Journal of Robotics Research
XX(X):1–23

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Andreas Orthey[1] and Florian T. Pokorny[2] and Lydia E. Kavraki[3]

Abstract

State space projections and decompositions have emerged as powerful tools to tackle the curse of dimensionality in high-dimensional, multi-robot motion planning problems. However, existing methods lack a unified framework which seamlessly handles combinations of projections (prioritization or task-space) and decompositions (parallel or decoupled subspaces). To fill this gap, we introduce fibration trees, which are trees consisting of state spaces as nodes and fibrations as edges, whereby a fibration models a projection from a higher-dimensional space to a lower-dimensional (or simplified) space. By modeling projections as fibrations, we unify sequential prioritization, parallel decomposition, and task-space projections under a single, coherent formalism. Building on this, we develop the rapidly-exploring random fibration trees (Fibration-RRT) planner, a sampling-based motion planner that generalizes strategies from quotient-space RRT (for sequential prioritizations) and discrete RRT (for parallel decompositions), while allowing the inclusion of task-space projections. Fibration-RRT operates on user-defined fibration trees and is proven to be probabilistically complete. To test the generality and efficiency of Fibration-RRT, we provide an open-source implementation and conduct experiments on 32 scenarios using multi robot teams with up to 96 degrees of freedom. Our results indicate that Fibration-RRT efficiently solves high-dimensional problems by exploiting user-defined fibration trees, thereby establishing fibration trees as a powerful, unified framework for multi-robot motion planning.

Keywords

Multi-Robot Motion Planning, Heterogenous Planning, Hierarchical Motion Planning

1 Introduction

State space projections and decompositions are useful tools to break the complexity of high-dimensional motion planning and optimization problems. In the context of sampling-based motion planning, projections and decompositions have been used for multi-robot scenarios, where state spaces can have tens or hundreds of dimensions, like in the coordination of robot swarms (Dorigo et al. 2021; Kaiser and Hamann 2022), spacecrafts (Nino et al. 2025), robot construction teams (Hartmann et al. 2021), underwater vehicles (Zhou et al. 2022), or quadrotor swarms (Hönig et al. 2018). In previous works, efficient solvers have been devised to solve heterogeneous multi-robot planning problems using projections (prioritized planning) (Orthey et al. 2024a; Vidal et al. 2019; Ferbach and Barraquand 1997) or decompositions (parallel or decoupled planning) (Solovey et al. 2016a; Wagner and Choset 2015a; Shome et al. 2020). To us, those are examples of hierarchical representations which have been imposed onto the state space of a problem to more efficiently solve high-dimensional, multi-robot motion planning problems.

However, those hierarchical representations for multi-robot motion planning have so far been treated separately and lack a unified framework. This is problematic. First, when a user wants to change the hierarchical representation, they would also need the change the planner associated with it. This is inconvenient and makes it difficult to benchmark multi-robot planners, because we do not understand if the change in hierarchical representation or the change in the planner contributes to a lower runtime. Second, it is

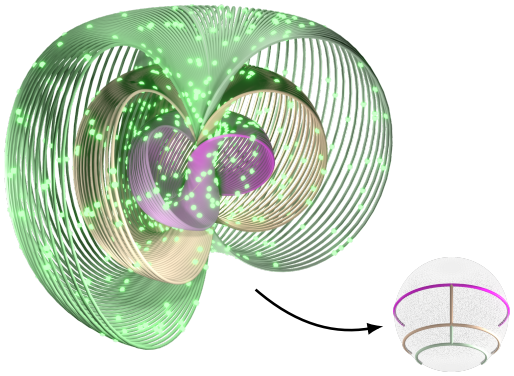


Figure 1. We simplify multi-robot motion planning problems using fibration trees, which are hierarchical representations of state spaces where nodes model state space simplifications. An example is the Hopf fibration $\mathbb{S}^3 \to \mathbb{S}^2$, which can be understood as a simplification of $\mathbb{S}^3$ by $\mathbb{S}^2$. A graph on $\mathbb{S}^2$ is shown (bottom right), with the graph restriction on $\mathbb{S}^3$ visualized as the fibers of the Hopf fibration. The color gradient of the edges on $\mathbb{S}^2$ reflects the corresponding fibers on $\mathbb{S}^3$.

[1]Technical University of Berlin, Germany
[2]KTH Royal Institute of Technology, Stockholm, Sweden
[3]Rice University and the Ken Kennedy Institute, Houston, TX, US

Corresponding author:
Andreas Orthey
Email: {andreas}@orthey.net

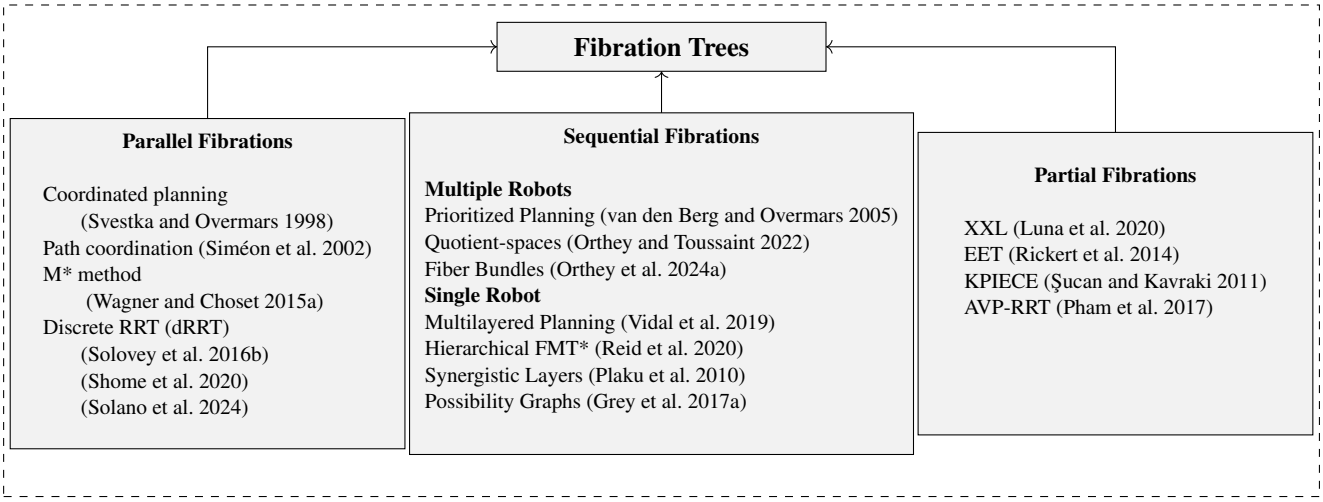


Figure 2. Fibration trees as unification tool to represent parallel, sequential, and partial fibrations. For each fibration, we list a non-exhaustive list of methods which make use of this particular instantiation of a fibration tree together with a tailor-made solver to exploit the particular structure. Please see Sec. 2 for a more comprehensive list of methods and Sec. 4.2 for related definitions.

currently not possible to switch hierarchical representations seamlessly without changing the planner. For example, when users understand a planning problem better and want to change the representation to improve runtime, they would also need to change the planner itself, because planners in the literature are often tailor-made for a particular representation (e.g. rapidly-exploring random quotient-space trees (QRRT) (Orthey et al. 2024a) for sequential prioritizations or discrete-RRT (dRRT) (Shome et al. 2020) for decoupled representations). Finally, there is no unified way to create hierarchical representations for multi-robot motion planning problems in the Open Motion Planning Library (OMPL) (Şucan et al. 2012). This makes it difficult to select, compare, and evaluate different representations to multi-robot motion planning and to improve upon them in the robotics community.

To close those gaps, we propose the framework of fibration trees. Fibration trees are trees consisting of state spaces as nodes and fibrations (Hatcher 2002) as edges, whereby a fibration models a projection from a higher-dimensional space to a lower-dimensional (or simplified) space. An example is the Hopf fibration (Hopf 1931) (see Fig. 1), which is a fibration from $\mathbb{S}^3$ to $\mathbb{S}^2$, which we use as a simplification of the space $\mathrm{SO}(3)$ by the space $\mathbb{S}^2$ (see details in Appendix A). In this example, we leverage the fibration by planning on $\mathbb{S}^2$ to create a tree on the sphere (visualized in lower right corner). By using this tree, we then build up the space of points on $\mathbb{S}^3$ which project onto the tree (called a restriction). By sampling exclusively this restriction, we can bias sampling towards the existing tree structure, which is similar to the way an admissible heuristic prioritizes paths which have lower cost on the simplified space (Orthey and Toussaint 2022). The resulting planned motions lie on $\mathrm{SO}(3)$ and represent those motions which project onto the existing tree on $\mathbb{S}^2$. Simplifications like this are helpful to derive admissible heuristics which have been shown to improve runtime significantly (Orthey and Toussaint 2022).

By developing fibration trees, we directly contribute to multi-robot motion planning by integrating hierarchical representations into a single, unified framework. This works as follows. Existing representations are analyzed and cast as specific fibration types. Prioritized planning is represented as sequential fibrations; Decomposition-based planning is represented as parallel fibrations; Task-space projections are represented as partial fibrations, which are projections which might have empty lifts. This is shown in Fig. 2, where the three categories of fibrations are shown together with algorithms which use those representations to simplify planning problems. In each case, users of the algorithm have to (implicitly) make a decision about which representation to use for a given problem. We make this decision explicit by requiring users to explicitly construct fibration trees, which requires users for example to decide between a prioritization or a decomposition.

To efficiently exploit a given fibration tree, we propose a new algorithm, the Rapidly-Exploring Random Fibration Trees method (Fibration-RRT). Fibration-RRT can solve motion planning problems with an added structure like projection-based planning, decomposition-based planning, or planning with task-space projections. On top of that, it can also deal with any combination of projections, for example when users have domain knowledge and want a custom hierarchical representation. To implement Fibration-RRT, we propose an extension to the OMPL library, such that multi-robot motion planning problems can be defined both using prioritization-based approaches, decomposition-based planning, using task-space projections, or using a combination of them. This opens the way towards more robust, comparable multi-robot planning algorithms.

To summarize, this paper makes the following contributions:

1. We define the framework of fibration trees unifying decomposition-based frameworks, projection-based frameworks, and task-space projections.

2. We develop a novel planner, Fibration-RRT, which combines elements from dRRT (Solovey et al. 2016a), and QRRT (Orthey et al. 2024a) to exploit fibration trees and we show that Fibration-RRT is probabilistically complete.

Table 1. Comparison of planners using projections to solve motion planning problems. For each planner, we note if a certain property exists (green) or does not exists (grey). Sec. 4.1 for the definition of projections.

Method	Admissible	Liftable	Non-Euclidean	Non-trivial	Partial	Decompositions
Parallel Fibrations (Decompositions)						
M* (Wagner and Choset 2015a)	■	■	■			■
Discrete RRT (dRRT*) (Shome et al. 2020)	■	■				■
Fast-dRRT* (Solano et al. 2024)	■	■				■
Partial Fibrations						
XXL (Luna et al. 2020)	■	■	■		■	
KPIECE (Şucan and Kavraki 2011)	■	■	■		■	
Path-Velocity Decomposition						
AVP-RRT (Pham et al. 2017)	■	■			■	
Sequential Fibrations						
Hierarchical FMT* (Reid et al. 2020)	■	■				
Possibility Graphs (Grey et al. 2017a)	■	■				
Quotient-space (Orthey et al. 2018)	■	■	■			
Fiber Bundles (Orthey et al. 2024a)	■	■	■	■		
Fibration Trees (**this paper**)	■	■	■	■	■	■

3. We evaluate Fibration-RRT on 32 scenarios divided into 5 benchmarks, involving comparisons to classical planners, prioritization-based planners, decomposition-based planners, and task-space planners on different multi-robot planning scenarios.

4. We implement Fibration-RRT in the open motion planning library (OMPL) (Moll et al. 2015; Şucan et al. 2012) with a front-end based upon the Dynamic Animation and Robotics Toolkit (DART) (Lee et al. 2018a) to create fibration trees.

To our knowledge, this is the first time that one single planner (Fibration-RRT) can solve all scenarios by taking the different projection types into account. We thereby demonstrate that fibration trees provide a unified framework for multi-robot planning algorithms.

2 Related Work

Fibration trees form a framework for sampling-based motion planning, which is based upon and extends prioritization-based and decomposition-based planning. Moreover, it integrates task-space and workspace projections to create a unified framework. To put fibration trees into the wider scientific context, we give a brief overview about sampling-based multi-robot motion planning and review the underlying methods of prioritization-based and decomposition-based planning. Finally, we discuss how fibration trees are related to learning-based approaches in multi-robot motion planning.

2.1 Sampling-based Multi-Robot Motion Planning

Sampling-based motion planning (Orthey et al. 2024b) is an approach to motion planning (Kavraki and LaValle 2016; Choset et al. 2005; LaValle 2006), whereby motions are found through random sampling of robot configurations. Early works in the field formulated this problem in terms of planning through configuration spaces (Lozano-Pérez and Wesley 1979; Lozano-Perez 1983), whereby sampling-based methods like graph-based approaches (Kavraki et al. 1996) and tree-based approaches (Kuffner and LaValle 2000; Hsu et al. 1999) have become the de facto standard way to conduct planning (LaValle 2006; Orthey et al. 2024b).

Multi-robot motion planning is a generalization of single-robot motion planning, whereby planning is conducted for multiple disconnected kinematic chains. This can be formulated as planning in a composite configuration space (Schwartz and Sharir 1983). However, since the complexity of motion planning scales exponentially with the number of dimensions (Reif 1979; Hopcroft et al. 1984; Canny 1988), research on multi-robot motion planning has focused on a smarter fragmentation of the composite configuration space, such that planners can operate more efficiently. For some specific cases, like homogeneous robot teams, this can be accomplished by reducing the problem to the pebbles-on-a-graph problem (Kornhauser et al. 1984), which can for example be solved using methods like push and swap (Luna and Bekris 2011), linear programming (Yu and LaValle 2016), subgraph partitioning (Ryan 2010), or conflict-based search (CBS) (Sharon et al. 2015; Hönig et al. 2018; Moldagalieva et al. 2024). Our work explicitly tackles the heterogeneous case, where robot teams can have different compositions.

For heterogeneous robot teams, most planning approaches can be divided into either prioritized or decomposition-based planning. Prioritization-based approaches (Schwartz and Sharir 1983; Erdmann and Lozano-Perez 1987) plan first for a single robot, then use the resulting path as a constraint for the next robot, until we accounted for all robots (Orthey et al. 2024a). This creates a sequential planning problem where the ordering of the robots has to be prespecified (van den Berg and Overmars 2005) or optimized (Bennewitz et al. 2001). Decomposition-based (or decoupled) planning (Svestka and Overmars 1995; van den Berg et al. 2010; Solovey et al. 2016a) differs by planning in parallel for each robot, and then leveraging the results to compute a solution in the composite configuration space. We review both approaches and discuss their individual trade-offs.

2.2 Prioritization-based and Projection-based planning

Instead of planning in the composite state space, it is often advantageous to plan motions sequentially, where the robots are ordered and planning is prioritized (Erdmann and Lozano-Perez 1987; Tournassoud 1986; van den Berg and Overmars 2005). Prioritized motion planning is particularly effective if robots are near-independent and do not block each other (Orthey et al. 2024a).

It is often crucial to choose the right ordering of robots to avoid any deadlocks (Heselden and Das 2023). While most methods utilize a fixed ordering (van den Berg and Overmars 2005), one can also search over the space of all robot orderings to find an order which leads to a collision-free path (Ma et al. 2019). Methods such as hill-climbing can

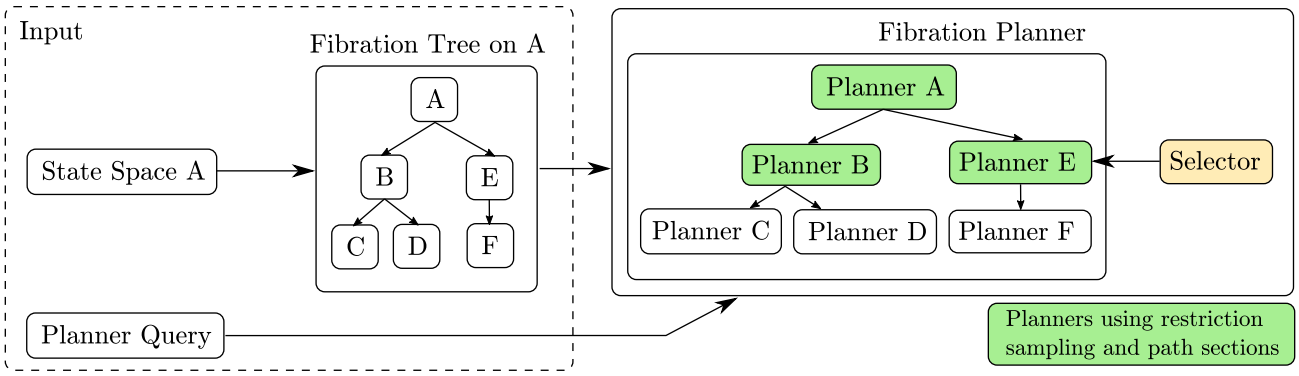


Figure 3. Overview about planning with fibration trees. Given a state space A, a fibration tree formalizes abstractions imposed onto A. This fibration tree contains, as an example, additional state spaces B, C, D, E, F which are simplifications of the original state space A. A number of projections are attached as edges between state spaces, which describe how state spaces are related. Given such a fibration tree and a planner query, a fibration planner is created, which attaches an individual planner to each state space, and uses a selector to choose which planner to run. The non-leaf nodes in the fibration tree have attached planners which can plan efficiently using restriction sampling (see Sec. 5.3) and path sections (see Sec. 5.4).

be utilized (Bennewitz et al. 2001), which swap two robots in the order on each iteration step.

Once a prioritization (an ordering of robots) has been chosen, there are multiple (heuristic) ways to efficiently compute solutions and resolve conflicts. A frequently used approach is based on the analysis of the robot workspace. By using the workspace of the robot, one can for instance compute an environment skeleton (McBeth et al. 2023). This skeleton can then be used to quickly reroute robots if conflicts arise (Zhang et al. 2025). Another method is based upon homotopy classes (Bhattacharya and Ghrist 2018). Here, robots compute path prospects (paths in different homotopy classes) to avoid and resolve conflicts on the fly (Wu et al. 2020). Another strategy is to use distributed planning, where robots plan in parallel (Velagapudi et al. 2010) and conflicts are resolved according to priority. Some research has also focused on using approximations of the area swept by a lower-priority robot to improve planning time for subsequent higher-priority robots (Keppler and Wagner 2020).

Often it is even possible to generate sufficient conditions on the solvability of prioritized motions (Čáp et al. 2015). This involves low-priority robots avoiding the start configurations of higher-priority robots, while higher-priority robots avoiding the goal configuration of the lower-priority robots. If such paths are found, higher-priority robots could just wait until lower priority robots have reached their goal. This would be a sufficient condition for a solution. Our work is complementary by providing a unified framework in which those heuristics can be implemented and compared.

2.2.1 Projection-Based Planning Prioritization-based approaches for multiple robots are also closely related to projection-based planning for single robots. While prioritization-based approaches simplify problems by removing robots, projection-based approaches simplify problems by removing degrees of freedom (Orthey et al. 2018; Orthey and Toussaint 2022). This is known in the literature under different names like multi-layered planning (Vidal et al. 2019), quotient-space planning (Orthey et al. 2018; Orthey and Toussaint 2022), or subspace projections (Reid et al. 2019, 2020).

A widely used method to solve projection-based problems more efficiently is the path section method (also called root- or lead-path method) (Wermelinger et al. 2016; Tonneau et al. 2018; Vidal et al. 2019; Orthey and Toussaint 2021a). This method works by first computing a solution on a simplified space (a base path). Using this base path, the search space is restricted to allow only solutions ”above” the base path, i.e. solutions which lie in the path restriction (see our terminology in Sec. 4.3). This can speed up planning problems significantly (Orthey and Toussaint 2022). Another popular method leverages probabilistic roadmaps (PRM) (Kavraki et al. 1996) by using it as a method to solve individual spaces (Orthey et al. 2018). This is similar to subspace projections, a method which has been used to extend the fast marching trees (FMT) (Janson et al. 2015) planner to create the hierarchical fast marching trees (HFMT*) planner (Reid et al. 2019, 2020). HFMT* can optimally solve sequential fibrations in euclidean spaces. Our method is similar in that we can integrate projections as input while implementing a unified path section method (see Sec. 5.4).

Fibration trees are also closely related to the fiber bundle approach (Orthey et al. 2024a). This approach used quotient-spaces to create sequential projections. Such projections can be efficiently exploited using dedicated planners like QMP (Orthey et al. 2018), QRRT (Orthey and Toussaint 2022), QMP*, and QRRT* (Orthey et al. 2024a). Our approach generalizes this approach to provide a unified perspective including not only sequential, but also parallel and partial fibrations (Şucan and Kavraki 2011).

2.3 Decomposition-based planning

An efficient method to solve multi-robot motion planning problems are decomposition-based planners (Siméon et al. 2002). Instead of planning in the combined state space of all robots, decomposition-based planners solve first the state spaces of each individual robot (while ignoring the other robots), and then lift those solutions into the combined state space (while checking collisions in-between robots). This

lifting action can either use path coordination, where only the solution paths are considered (Siméon et al. 2002), or use graph coordination, where all graphs on the individual spaces are considered (Svestka and Overmars 1998).

Since graph coordination leads to probabilistically complete planners (Svestka and Overmars 1998), it has been the main approach for decomposition-based planning. Several planners have been proposed for graph coordination, such as M* (Wagner and Choset 2015a), which extends the A* planner, by using the individual solutions as admissible heuristics to guide the search A similar idea is used in the discrete RRT (dRRT) (Solovey et al. 2016a) and dRRT* planner (Dobson et al. 2017; Shome et al. 2020), which improves upon M* by using a more efficient neighbor evaluation function (oracle) for Euclidean spaces.

The dRRT planner has been particularly successful and has been extended by multiple works. For example, one can combine it with a conflict-based search where a sampling-based motion planner is spawned in a subspace whenever a conflict between two robots arises (Solis et al. 2024). For lower dimensional spaces, it can also be helpful to use lattice-based roadmaps (Parque and Miyashita 2023) or simultaneous sampling (Keisuke and Xavier 2023) on the individual spaces.

Recently, planners have been developed which pursue a hybrid approach. Scheduling to Avoid Collisions (StAC) (Guo et al. 2026) is such a planner which uses a hybrid strategy between coupled (prioritization-based) and decoupled (decomposition-based) approaches. They plan motions for each individual robot, then do path coordination with random pauses to find a valid solution for all robots. If this fails, the individual planners are updated with this information and planning continues until a valid solution is found.

Our work is complementary to approaches like dRRT or StAC by providing a unified framework in which decomposition-based or hybrid planning can be implemented as a special case. Since our framework can handle any fibration, it allows us to compare decomposition-based planners (Solovey et al. 2016a; Shome et al. 2020) with prioritization-based approaches—to understand which approach is more appropriate for a given task.

2.4 Learning-Based Approaches to Multi-Robot Planning

While this work focuses on a unified framework for sampling-based multi-robot motion planning, recent research has focused on learning-based multi-robot motion generation. Promising methods are diffusion models (Shaoul et al. 2025; Liang et al. 2025; Jiang et al. 2023), which are networks which learn trajectories by gradually adding noise to the input data. Other notable learning-based directions include hierarchical reinforcement learning approaches (Bettini et al. 2023, 2024; Sebastián et al. 2025; Wu and Suh 2024), imitation learning (Sun and Liao 2025), and dedicated multi-robot reinforcement-learning environments (Hu et al. 2023). Closely related to our work are also quotient-based Markov decision processes (Quotient-MDP) (Welde* et al. 2024), which are ways to train a policy on a lower-dimensional quotient-MDP which is then lifted using a tracking controller to the original space. This is particularly effective for homogeneous robot teams, since robots can share neural network weights between themselves.

While learning-based approaches have shown significant improvements, they require large datasets to train a policy. Our approach is complementary in that we provide tools to provide abstractions and methods to efficiently solve those problems. Our methods can then be leveraged to generate data points to learn and extend robot policies (Ha et al. 2020; Lai et al. 2025).

3 Multi-Robot Motion Planning

Let $X = Y_1 \times \cdots \times Y_M$ be the composite state space of M robots with individual state spaces Y_m. Each state space X is defined by the set of configurations a robot can attain plus the following structures:

- Distance metric $d : X \times X \to \mathbb{R}_{\geq 0}$ which measure the distance between two states.
- Steering function f which extends path segments from a state $x \in X$ towards another state $y \in X$.
- Boolean constraint function $\phi : X \to \{0, 1\}$ which evaluates to 0 if a state x is constraint free and 1 otherwise.
- Sampling sequence $s = \{x_1, x_2, \ldots\}$ which produces a uniform, infinite set of (dense) samples on X.

The constraint function ϕ decomposes the state space into two regions, whereby the states for which ϕ evaluates to true are called *feasible states* and $X_f = \{x \in X \mid \phi(x) = 0\}$ denotes the free state space.

Let (x_I, X_G) be a planner query consisting of an initial state $x_I \in X_f$ and a goal region $X_G \subseteq X_f$. A motion planning problem is then defined as the tuple (X, x_I, X_G) which asks us to find a path (a continuous mapping $p : [0, 1] \to X_f$) from the start state to the goal region.

4 Fibration Trees

In this section, we introduce the concept of fibration trees. A fibration tree is a user-defined structure, which can be imposed on the state space and which serves as a hierarchical representation of it. Fibration trees can be exploited by a dedicated planner like Fibration-RRT (see Sec. 5) to efficiently search the state space and lower runtime significantly. Examples of fibration trees include decompositions of the state space, prioritizations of robots in multi-robot planning, or task-space projections.

Let us define fibration trees more precisely. Let X be a state space. A fibration tree on X is a directed tree $T_X = (V, E)$ consisting of nodes V and edges E. Nodes represent state spaces together with an associated node planner, internal datastructure (tree, graph), and initial state and goal region. The root node contains the original state space X. Edges represent fibrations, which are tuples (A, B, π_{AB}) (written as $\pi_{AB} : A \to B$ or just $A \to B$), consisting of a state space A (the total space of a fibration), a state space B (the base space of a fibration), and a projection π_{AB} which maps elements of A to elements of B. Those concepts are

Name	Description
Node	A structure including a state space, a local planner, a datastructure (tree), plus a local planning problem including an initial state and a goal region.
Edge	Representing a Fibration between nodes.
Node Space	The corresponding space associated to a node.
Node Planner	The corresponding planner associated to a node.
Root Node	The root of a fibration tree
Fibration $\pi : A \to B$	A mapping between a node A (total space) and a node B (base space)
Total Space	The domain of a fibration
Base Space	The co-domain of a fibration

Table 2. Notations used in this paper.

summarized in Table. 2. The fibration mapping have to fulfill two axioms, which we detail next.

4.1 Projection axioms

Let $\pi_{AB} : A \to B$ be a fibration. To be useful for motion planning, a fibration has to be liftable and admissible.

4.1.1 Liftable A fibration is liftable if its projection has an inverse. Given the projection π_{AB} mapping elements of $a \in A$ to $b \in B$, we say that π_{AB} is liftable, if π_{AB} has an inverse π_{AB}^{-1} mapping an element $b \in B$ to an element $a \in A$.

Raison d'être: Motion planning with non-liftable projections would prevent us from transfering information upwards in a fibration tree.

4.1.2 Admissible A projection is called admissible, if it preserves validity. A projection $A \to B$ is admissible, if, for any $a \in A$, $\phi(a) \Rightarrow \phi(\pi_{AB}(a))$ (Orthey and Toussaint 2022). This means that every feasible element $a \in A$ will be projected onto a feasible element $b \in B$.

Raison d'être: Motion planning with non-admissible projections would prevent us from keeping theoretical guarantees like probabilistic completeness.

4.2 Fibration Types

While many fibrations are liftable and admissible, there are three types which are particularly interesting for motion planning applications. Those are sequential, parallel, and partial fibrations. Those fibrations types are visualized in Fig. 4, and further detailed below.

4.2.1 Sequential Fibrations Sequential fibrations are fiber bundles (Steenrod 1951; Lee 2003), which are tuples (A, B, π_{AB}, F) consisting of a fibration with an additional fiber F. A fiber is the preimage $\pi_{AB}^{-1}(b)$, i.e. the subspace of all configurations on A which project onto the same point b on the base space B. The defining property of a fiber bundle is that A is locally a product space $B \times F$. This means that the preimage of π_{AB} is homotopically equivalent to the fiber, i.e. for any $b \in B$, the preimage $\pi_{AB}^{-1}(b)$ is a copy of F. For fiber bundles, the total space A can be thought of as a union of identical copies of F glued together by the properties of π. For details, see (Orthey et al. 2024a).

4.2.2 Parallel Fibrations Parallel fibrations are product spaces (Solovey et al. 2016b), such that A factors as $A \to A_1 \times \cdots \times A_M$, whereby $A = A_1 \times \cdots \times A_M$ is the product space, together with M projections $\pi_{A_1}, \ldots, \pi_{A_M}$ onto the individual factors as $\pi_{A_m} : A \to A_m$. Parallel fibrations model for example multi-robot decomposition-based planning problems, where A is the compound space, and $A_1, \cdots, A_M$ are the individual robot subspaces.

4.2.3 Partial Fibrations A partial fibration (A, B, π_{AB}) consists of a projection which can only lift a subset of elements of B, i.e. some points $b \in B$ lift to an empty set. As an example, consider the fibration $\mathbb{R}^6 \to T$ of a 6-dof manipulator arm with a projection onto a Tool-center point (Tcp) region $T \subseteq \mathbb{R}^3$ of the robot. For each point b on the base space (Tcp region), the states projecting onto b are given by the set $F(b) = \{x \in X \mid \pi^{-1}(b) = x\}$ (the fiber of b). For the manipulator, this involves solving an inverse kinematics (IK) problem. However, the fiber depends on the IK solver, and can be either an empty space (no IK solution), contain a single point, contain multiple points, or even contain a subspace of IK solutions. This means that there is only a subspace $B' \subset B$ on which the fibration returns a solution.

4.3 Restrictions and Sections on Fibration Trees

Fibrations provide the useful concepts of restrictions and sections, which we summarize in the next two sections.

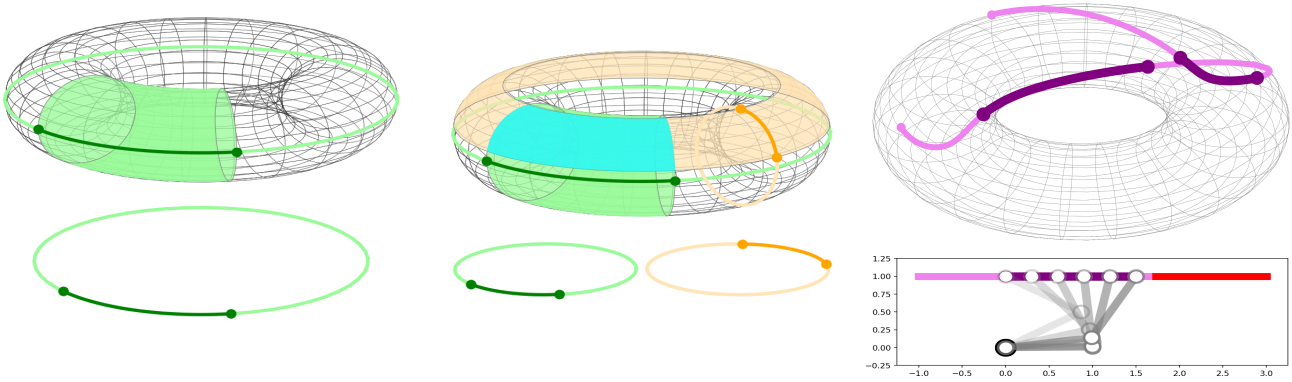
Those two concepts are used in two primitives of the Fibration-RRT planner, namely the unified restriction sampling (Sec. 5.3) and the unified path section search (Sec. 5.4).

4.3.1 Restrictions Given a fibration $A \to B$, and a set $U \subseteq B$, a restriction is defined as the set $\pi_{AB}^{-1}(U)$ on A. In this work, we make use of three main types of restrictions.

- **Fiber**: Given a point $b \in B$, we call $\pi^{-1}(b)$ the *fiber* of b in A. The fiber contains all points which, when projected onto B, are exactly equivalent to the point b.
- **Path restriction**: Given a path $p : [0, 1] \to B$, we call the inverse of the image* of p the path restriction on A. It contains all points which, when projected onto B, are located on the path image of p.
- **Graph restriction**: Let $G = (V, E)$ be a graph on B with vertices being points, and edges being path segments. The union of all fibers and path restrictions associated to the vertices and edges is called the *graph restriction*. It contains all points which, when projected onto B, are located on an edge or vertex of G.

4.3.2 Sections Let $A \to B$ be a fibration, and $R = \pi_{AB}^{-1}(U)$ be a restriction on A with associated set U on B. We define a section as a mapping $s : U \to R$ from $U \subseteq B$ to the restriction R. We make use of the following two section concepts.

*Note that we make an explicit distinction between a *path*, a mapping from $[0, 1]$ to a space B, and a *path image*, the image of the path, i.e. the set $p([0, 1])$.



(a) Sequential Fibration **(b)** Parallel Fibration **(c)** Partial Fibration

Figure 4. Examples of the different fibration types we consider in this paper as visualized on the torus T^2. **Left:** Sequential fibration with a projection $T^2 \to S^1$. The circle (green) has a path segment (darkgreen), which is lifted to a path restriction (green, on torus). **Middle:** A parallel fibration of T^2 to two circle spaces, one for the major axis (green), one for the minor axis (orange) of the torus. Two path segments are shown and their corresponding path restrictions on the torus. The intersection of both path restrictions (cyan) is shown on the torus. **Right:** A partial fibration $T^2 \to R^1$ of the torus to a line segment in the workspace of a 2-dof robot. Note that there are segments not liftable (red) in the workspace. The path on the torus (light purple) is a two-on-one mapping, since every point on the depicted workspace path (dark purple) has two IK solutions (corresponding to the dark purple segments on the torus).

Algorithm 1 FibrationRRT

Input: Planning query (X, x_I, X_G), Fibration tree T_X
Output: Solution path on X

$\mathbf{X} = \{\text{GETLEAFNODES}(T_X)\}$
while $\neg\text{PTC}(\mathbf{X}, T_X)$ **do**
  $A \leftarrow \text{SELECTNODE}(\mathbf{X})$ ▷ Sec. 5.1
  $A \leftarrow \text{PLANNODE}(A)$ ▷ Sec. 5.2
  $\mathbf{X} \leftarrow \text{UPDATEACTIVENODES}(T_X, A, \mathbf{X})$
end while
return G_X

- **Lift**: Given a fiber R over a base point b, we define a lift as the mapping $s : b \to R$, which selects a point in the fiber associated to b.
- **Path section**: Given a path restriction R over a base path p, we define a path section as $s : [0, 1] \to X$ such that $\pi(s(x)) = x$ for any b in $p[I]$.

This concludes our description of fibration trees, their notations, and their internal structures. Next, we describe a dedicated planner which can efficiently exploit fibration trees to improve runtime for high-dimensional multi-robot motion planning problems.

5 Rapidly-Exploring Random Fibration Trees

In this section, we describe the Rapidly-Exploring Random Fibration Trees (Fibration-RRT) algorithm. Fibration-RRT generalizes RRT (Kuffner and LaValle 2000) by taking as input an additional, user-defined fibration tree as defined in Sec. 4. Given a motion planning problem and a fibration tree, Fibration-RRT can efficiently exploit the tree to quickly solve even high-dimensional, multi-robot planning problems. We further show that Fibration-RRT is probabilistically complete given an arbitrary fibration tree (see Appendix B) and we detail how goals are set, which metric we use, and how the algorithm is implemented.

Algorithm 2 UpdateActiveNodes

Input: Fibration Tree T_X, A, $\mathbf{X}$
Output: Updated active nodes $\mathbf{X}$

if $\neg\text{HASPARENT}(A)$ **then**
  return
end if
if $\neg\text{SOLUTIONEXISTS}(A)$ **then**
  return
end if
$X_p \leftarrow \text{PARENT}(A, T_X)$
if $X_p \notin \mathbf{X}$ **then**
  $\mathbf{X}.\text{INSERT}(X_p)$
end if
return $\mathbf{X}$

Algorithm 3 SelectNode

Input: $\mathbf{X}$
Output: Node from $\mathbf{X}$

$\Phi \leftarrow \emptyset$ ▷ Probability Density Function (PDF)
for $X \in \mathbf{X}$ **do**
  if ISSOLVED(X) **then**
    if HASNONSOLVEDSIBLINGS(X) **then**
      continue
    end if
  end if
  $n \leftarrow \text{GETDIMENSION}(X)$
  $V \leftarrow \text{VALIDNODES}(X)$
  $p \leftarrow \frac{1}{\|V\|^{1/n} + 1}$
  $\Phi.\text{INSERT}(\{X, p\})$
end for
return $\text{SAMPLEFROMPDF}(\Phi)$

To start, we describe the high-level algorithm in Alg. 1. As input we use a planning query consisting of a state space X, start configuration x_I and a goal X_G plus a fibration tree structure T_X. Our first task is to extract the leaf nodes of T_X (Line 1) and add them to $\mathbf{X}$, the set of active nodes. We then iterate while the planner terminate condition (PTC) is false (Line 2). Inside one iteration, we run a three-step process. In step one (Line 3), we select a node from the set of active nodes. This is detailed in Sec. 5.1. In step two (Line 4), we run the planner associated to the selected node (detailed in Sec. 5.2). Finally, in step three (Line 5), we update the active nodes.

The update of the active nodes is further detailed in Alg. 2. This method adds the parents of an input node A to the active nodes when three conditions are true: Node A needs to have a parent (Line 1–3), a solution has to exist on node A (Line 4–6), and the parent of node A is not yet member of the set of active nodes (Line 8). If those conditions are met, we add the parent to the active nodes (Line 9).

5.1 Select Node

Given all nodes in the queue, a selection method is used to decide which node (and the planner therein) to execute next. This selection is based on an exponential importance criterion (Orthey et al. 2024a), which is defined as

$$i(X_k) = \frac{1}{\|V_k\|^{1/n_k} + 1} \tag{1}$$

whereby X_k is the k-th active node, V_k are the number of valid vertices in the tree of the planner on X_k, and n_k is the dimensionality of X_k. This criterion is motivated by the sampling density—the number of samples per unit distance—on an n-dimensional space which scales with $V^{1/n}$, whereby V are the number of samples (Hastie et al. 2009). Intuitively, Eq. (1) gives a low number for a high sampling density (i.e. we sampled a node already sufficiently well) and it gives a high number for a low sampling density (i.e. we have not yet sampled a node enough).

The algorithm to select a node is depicted in Alg. 3. It works by building a probability distribution (Line 1) over nodes which are then sampled. We start by iterating over all active nodes (Line 2) and check if the node should be considered for selection (Line 3-7). This is a special case for parallel fibrations, i.e. whenever we have a solution for one node in a parallel fibration, we first need to solve all sibling nodes—otherwise we cannot move on to the parent node.

After we verified that the node is selectable, we compute the exponential importance criterion from Eq. (1) (Line 8-10) and add it to the probability distribution function (Line 11). Eventually, we sample from the probability distribution to select a node—biased towards the lower sampling density (Line 13).

It is important to note that the planner selection will never stop exploring active nodes, and it ensures that every node is selected infinitely many times in the limit. We say that the selection method has the *Uniform Infinite Recurrence* property. This is a requirement to prove probabilistic completeness, as we will do in Appendix B.

To obtain a classical prioritized multi-robot planner (sequential fibration) (Van den Berg and Overmars 2005), we can modify the selection by using a *last-node* selection where only the last unsolved node is selected. While this has been implemented in our code, we have not used this in our experiments, since it would violate the Uniform Infinite Recurrence property and thereby remove the probabilistic completeness guarantee of the planner.

Algorithm 4 PlanNode

Input: Node A including initial configuration x_I, goal region X_G, fibration $\pi : A \to B$, and data structure G
Output: Updated node A

1: **if** IsEmpty(G) **then**
2: $\quad G \leftarrow$ PathSectionSearch(π, x_I, X_G) ▷ Sec. 5.4
3: **end if**
4: **while** $\neg$PTC(A) **do**
5: $\quad a_{\text{rand}} \leftarrow$ RestrictionSampling(π) ▷ Sec. 5.3
6: $\quad a_{\text{nearest}} \leftarrow$ Nearest(a_{rand}, G)
7: $\quad a_{\text{new}} \leftarrow$ Steer$(a_{\text{nearest}}, a_{\text{rand}}, A)$
8: $\quad$ **if** IsValid$(a_{\text{nearest}}, a_{\text{new}})$ **then**
9: $\quad\quad G \leftarrow G \cup \{a_{\text{nearest}}, a_{\text{new}}\}$
10: $\quad\quad$ **if** $a_{\text{new}} \in X_G$ **then**
11: $\quad\quad\quad A$.solved $\leftarrow$ True
12: $\quad\quad$ **end if**
13: $\quad$ **end if**
14: **end while**
15: **return** A

5.2 Planning on a Node

The core of the Fibration-RRT planner is the `PlanNode` method. While any motion planning algorithm could be used in this function, we have chosen the RRT (Kuffner and LaValle 2000) method due to its versatility.

The adjusted algorithm is shown in Alg. 4. This method is used to find a solution on a given total space represented by a node A. From the node, we use the (projected) initial configuration x_I and goal region X_G, fibration $\pi : A \to B$, and datastructure G. In our case this includes the tree of the RRT and additional information about the number of runs.

Planning on a node differs from RRT in two important aspects. First, if there is no data yet in G (Line 1), we run a path section method (Line 2), where we compute a path directly in the path restriction of the solution on B (if possible). This is further detailed in Sec. 5.4. Second, the sampling function is replaced by restriction sampling (Line 5), which takes the datastructure on the base space B into account. This is detailed in Sec. 5.3. The remainder of the algorithm follows the standard RRT method (Kuffner and LaValle 2000) by executing a loop while the planner termination condition (PTC) is not fulfilled (Line 4). Inside the loop, we sample and compute the nearest configuration in the datastructure G (Line 6), steer towards it (Line 7), and add the resulting edge to the datastructure (Line 8–9). If the new configuration has reached the goal region (Line 10), we mark the node as solved (Line 11). Finally, we return the updated node (Line 15).

Note that the `PlanNode` is an extension of the standard RRT (Kuffner and LaValle 2000) planner. This can be seen when we look at the special case where π is an empty projection, i.e. there is no base space. In this case,

Algorithm 5 RestrictionSampling

Input: Fibration $\pi : A \to B$
Parameters: Path sampling bias (Case 2), path restriction surrounding bias (Case 2), sampling perturbation bias (Case 2)
Output: A configuration $a \in A$

▷ **Case 1**: A is a leaf node or π is not a fibration.
1: **if** $\neg$EXISTS(B) **then**
2: | **return** SAMPLE(A)
3: **end if**
▷ **Case 2**: π is a sequential- or partial fibration.
4: **if** HASFIBER(π) **then**
5: | $x_{\text{base}} \leftarrow$ SAMPLEDATASTRUCTURE(B)
6: | $x \leftarrow$ SAMPLEFIBER(x_{base}, π)
7: | **return** x
8: **end if**
▷ **Case 3**: π is a parallel fibration.
9: $x \leftarrow$ IDENTITYELEMENT(A)
10: **for** $B_m \in B_{1:M}$ **do**
11: | $x_{\text{base}} \leftarrow$ SAMPLEDATASTRUCTURE(B_m)
12: | $x \leftarrow$ INSERTIONMAP($x_{\text{base}}, \pi, B_m$)
13: **end for**
14: **return** x

Fibration-RRT performs exactly equivalent to RRT, whereby path section performs no operation and restriction sampling reverts back to uniform sampling. This is similar to previous works, where RRT was extended to the quotient-space RRT (Orthey and Toussaint 2022). However, this only worked for the special case of quotient spaces as sequential fibrations, while our method represents a general approach which can also handle parallel and partial fibrations.

5.3 Unified Restriction Sampling

Given a fibration $\pi : A \to B$, one of the most important planning primitives is restriction sampling. Restriction sampling (Alg. 5) returns a configuration which lies in the restriction of the current datastructure on the base space B.

Depending on the type of projection, we have to distinguish three cases. In the first case (Line 1–3), π is an empty projection (i.e. the space A belongs to a leaf node of the fibration tree). In this case, we uniformly sample the space A and return.

In the second case (Line 4–8), if the projection has a fiber, there is an explicit way to sample the fiber. We therefore first sample a configuration x_{base} on the base space B (Line 5) and then compute a fiber element to obtain a configuration x on the total space A. Note that partial fibrations can have no solutions, in which case a default invalid element is returned. Note that this case has three associated parameters. First, the path sampling bias. This is a percentage of samples drawn from the solution path on B instead of sampling from the tree. Second, the path restriction surrounding bias, which provides a small margin around the solution path on B in which the path sampling takes place. Finally, the sampling perturbation bias, which additionally perturbates a sampled configuration to sample effectively in a neighborhood of the graph restriction (Orthey and Toussaint 2021b). See Sec. 6.1 for implementation values.

Algorithm 6 PathSectionSearch

Input: Fibration $\pi : A \to B$, initial configuration x_I
Parameters: Maximum branching (Case 2), maximum depth (Case 2), Restriction accuracy (Case 3), propagation attempts (Case 3), maximum number of permutations (Case 4)
Output: A (solution) path on the path restriction or a failure

▷ **Case 1**: A is a leaf node or has no solution.
1: **if** $\neg$EXISTS(B) **or** $\neg$HASSOLUTION(B) **then**
2: | **return**
3: **end if**
4: $p_{\text{base}} \leftarrow$ GETSOLUTIONPATH(B)
▷ **Case 2**: π is a sequential fibration
5: **if** ISSEQUENTIAL(π) **then**
6: | $F \leftarrow$ GETFIBER(π)
7: | $p \leftarrow$ INTERPOLATESECTION($p_{\text{base}}, \pi, x_I$)
8: | $p \leftarrow$ RECURSIVESIDESTEP(p_{base}, π)
9: | **return** p
10: **end if**
▷ **Case 3**: π is a partial fibration
11: **if** ISPARTIAL(π) **then**
12: | $n \leftarrow 1$
13: | $p \leftarrow \{x_I\}$
14: | **while** NOTEMPTY($p_{\text{base}}(n)$) **do**
15: | | $a' \leftarrow$ SAMPLEFIBER($p_{\text{base}}(n), \pi$)
16: | | $a \leftarrow p(n-1)$
17: | | $a' \leftarrow$ STEER(a, a')
18: | | **if** $\neg$ISVALID(a, a') **then**
19: | | | **break**
20: | | **end if**
21: | | $p \leftarrow p \cup \{a, a'\}$
22: | **end while**
23: | **return** p
24: **end if**
▷ **Case 4**: π is a parallel fibration
25: $\Sigma \leftarrow$ GETBOUNDEDUNIQUEPERMUTATIONS(π)
26: **for** $\sigma \in \Sigma$ **do**
27: | $p \leftarrow$ INTERPOLATEPERMUTATEDPATH($p_{\text{base}}, \sigma, x_I$)
28: | **if** ISVALID(p) **then**
29: | | **return** p
30: | **end if**
31: **end for**
32: **return**

In the last case (Line 9–14), π is a parallel projection. In this case we iterate over all children spaces, sample a base space element (Line 11) and apply the insertion map to map the element into the total space (Line 12). Once all states are mapped, the total space element (Line 14) is returned.

5.4 Unified Path Section Search

Given a fibration $\pi : A \to B$, and a solution path on the base space B, a valuable primitive for efficient planning is the path section search (Orthey and Toussaint 2021a). A path section search is a dedicated local planner which searches exclusively in the path restriction for a valid path. In detail, given a solution path $p_{\text{base}} : [0, 1] \to B$, the path restriction on A is defined as the set $\{x \in A \mid \pi(x) \in p_{\text{base}}([0, 1])\}$, i.e. all elements in A, which, when projected onto B, lie

on the path image $p_{\text{base}}([0,1])$ in B. The start and goal configurations for the path section search are elements of the fibers of $p_{\text{base}}(0)$ and $p_{\text{base}}(1)$ on A, respectively. Those are either well-defined from a top-down goal, or sampleable using a bottom-up goal.

The inner workings of the path section search are shown in Alg. 6. There are four cases we have to address. First, when A is a leaf node or no solution exists on B, there does not exist a path section. In this case, we can terminate immediately (Line 1–3). Once this case is exhausted, we can get the solution path from B (Line 4) and continue.

In the second case, π is a sequential fibration. In this case, we can employ a path section search like section patterns (Orthey and Toussaint 2021a). However, due to the possible high-dimensionality of the space A, especially when using multiple robots, we used the recursive side step method which we introduced in an earlier paper (Orthey et al. 2024a). This method interpolates an L1-metric path on the path restriction (Line 6–7), and then follows it until a constraint violation occurs. Once this happens, we compute a new L1-metric path starting from the violated configuration and continue until no side step is valid or we reach the goal (Line 8). The resulting path is then returned (Line 9). This function uses two parameters, the maximum branching (which determines how many L1 paths are created once a violated configuration is found) and the maximum depth (which determines how many times a branching is allowed to happen until we stop).

In the third case, π is a partial fibration. In this case, a path interpolation method is not directly available. Instead, we use the steering function to move from one configuration to the next along the restriction of the solution path. We start by initializing a path from the initial configuration (Line 13). We then steer until we reach the last base path configuration (Line 14). For each base path configuration, we sample a random fiber element (Line 15) and steer towards it (Line 17). If this segment is not valid or we do not reach the lifted element (Line 18), we break (Line 19) and return the path so far (Line 23). Otherwise, we add the last segment to the path and continue (Line 21). This method uses two parameters, the restriction accuracy (which determines how close we need to steer to a fiber element) and the number of propagation attemps (which determines how often we repeat the steering when unsuccessful).

Finally, in the last case, π is a parallel fibration. We found that in this case, the most efficient way to find a path section is similar to a side step function (Orthey et al. 2024a), but instead of interpolating an L1-metric path, we interpolate each factor separately. For example, when there are three factors, we would go through every permutation and find a path section for each without moving the degrees of freedom of the other two factors. This method can be generalized by first finding a set of permutations (Line 25). We cap this number because the number of permutations grows exponentially with the number of factors. We then iterate over all permutations (Line 26) and interpolate the section based on the permutation (Line 27). Once this path section is found, we check its validity and return the path in case of success (Line 28–29). If no valid section is found, we return without a path (Line 32). This method uses one parameter, the maximum number of permutations.

Note that this method is similar to the local connector method Solovey et al. (2016a); van den Berg and Overmars (2005), but uses multiple ordering permutations to increase the likelihood of finding a feasible path.

5.5 Goal Setting

Fibration trees support two modes of goal setting, a top-down goal and a bottom-up goal. Those are two ways of specifying problems: In a top-down goal, you specify a single configuration as a goal which is valid for all robots. In a bottom-up goal, you only specify the goals for leaf nodes, which is useful for task-space goals (e.g. the end-effector position) or for simplified robot geometries (e.g. the base of a robot). We explain both types below.

Top-Down Goal For the top-down goal, a single goal configuration is set on the root node. In this case, the goals for all nodes are defined by recursively projecting this goal configuration downwards. This provides a unique, well-defined goal configuration for each of the nodes in the fibration tree. Note that a top-down goal, if available, is the recommended choice for fibration tree planning. However, a top-down goal might not always be available or the goal has different regions, where picking a single configuration might lead to making the problem infeasible.

Bottom-Up Goal A bottom-up goal is defined by specifying a configuration for each leaf node in the fibration tree. This is not well-defined, since multiple root configurations can project on the same leaf-node configurations. This ambiguity can be resolved in one of two ways. Either, before planning, all the leaf-nodes are lifted to the root node state and a validity check is performed. If the state is valid it is taken as a top-down goal. Otherwise, the procedure continues until a maximum number of iterations is reached, or a valid root state is found. This method is called *valid leaf-state lifting*.

Or, as a second approach, leaf-node states can be used as sampleable goals for their parent nodes. Every goal, including the root goal, is defined implicitly by sampling leaf-node states and then lifting them upwards. Note that this differs from valid leaf-state lifting by postponing the lifting for each node towards a time when a goal state is already available for the corresponding child node. This method is called a *liftable goal region*.

Synthesis of Bottom-Up Approaches Both bottom-up approaches have disadvantages. One disadvantage is early termination. If a problem is infeasible, a liftable goal region will need to go through all nodes and perform unneccessary computations, even if valid leaf-state lifting could quickly terminate. Another disadvantage is completeness. If valid leaf-state lifting produces a configuration, this configuration might be unreachable, and therefore make the problem infeasible—even if a solution exists.

In practice, we therefore use a combination of both approaches to resolve their respective disadvantages. First, we use valid leaf-state lifting to guarantee that a valid goal configuration exists. Second, we start planning with liftable goal regions. This makes the planner keep the probabilistic completeness properties, while ensuring that a goal configuration theoretically exists on the root node.

5.6 L-Infinity Metric

In this work, Fibration-RRT and all planners in our benchmark use exclusively the L-infinity metric (Atias et al. 2018). In this section, we explain our reasoning for this choice. While earlier work (Atias et al. 2018) has not found a significant difference in the choice of metric, we have opted to use the L-infinity metric for Fibration-RRT based upon the following reasons.

- **Interpretation** Thresholds for goal regions in high-dimensional spaces using Lp-metrics are difficult to interpret. However, when we use the L-infinity metric, a threshold can be interpreted as the maximum joint displacement over all DoFs of all robots. This is straightforward to define and intuitive for users of the planner.
- **Avoidance of Clustering** Lp-metrics consider the sum of many independent variables, which, by the law of large numbers, often have similar expected values. Most of the outcome therefore clusters around certain values (Aggarwal et al. 2001). In contrast, the L-infinity metric provides the largest DoF deviation, which is independent of the number of DoFs.
- **More Meaningful Nearest Neighbor**. When using Lp-metrics, we could get a low value for the distance, even if one robot is having a large distance, since we average over all dimensions. However, when using the L-infinity metric, we ensure that robots are really nearby.

Due to those reasons, we have opted to use the L-infinity metric for Fibration-RRT. To ensure that the metric is not influencing benchmarking, we use the same metric for all planners. We believe future work should investigate the influence of the metric when the dimensionality is increased significantly (Atias et al. 2018; Aggarwal et al. 2001).

5.7 Software Implementations

All code is open source and available as an extension of the OMPL library[†]. We plan to merge this as a multi-robot extension into the OMPL main github repository. For evaluations and implementation of partial (task-space) projections, we wrote a framework based upon this OMPL extension. This framework is based upon the Dynamic Animation and Robotics Toolkit (DART) (Lee et al. 2018b) and is also available on github[‡].

6 Benchmarks

To evaluate Fibration-RRT on different fibration trees, we utilize the DART library (Lee et al. 2018a) in combination with OMPL for benchmarking (Moll et al. 2015). Our evaluations involve five benchmarks:

- **Overhead Benchmark** Comparison of Fibration-RRT with the classical RRT algorithm to find any computational overhead (Sec. 6.2).
- **Structural Benchmark** Comparison of decomposition-based and prioritization-based fibration trees using Fibration-RRT and comparing them to single-tree, single-query planners (Sec. 6.3).
- **Prioritization Benchmark** Comparison of Fibration-RRT using a prioritization-based fibration tree against a prioritization-based planner, namely the rapidly exploring random quotient-space tree (QRRT) planner (Orthey et al. 2024a) (Sec. 6.4).
- **Decomposition Benchmark** Comparison of Fibration-RRT using a decomposition-based fibration tree against a decomposition-based planner, namely multi-robot discrete RRT (dRRT) planner (Solovey et al. 2016a) (Sec. 6.5).
- **Task-Space Benchmark** Comparison of Fibration-RRT using task-space constraints with Task-RRT (Berenson et al. 2011) (Sec. 6.6).

We report in the following sections on the results of those benchmarks. Please see Sec. 7 on the interpretation of those results.

6.1 Experimental Setup

We evaluate our benchmarks on a 4-core, 8GB RAM laptop running Ubuntu 24.04. The following parameters are specified for Fibration-RRT, together with the default values set for the experiments.

- **Goal bias** A value in $[0, 1]$ indicating the percentage of samples drawn from the goal. The default value is 0.05 (same as RRT).
- **Range** A value in $\mathrm{R}_{\geq 0}$ indicating the maximum extension distance of a single iteration. The default value is computed from the volume of the configuration space (same as RRT).
- **Selector function type** A type indicating that either uniform selection, exponential selection (Sec. 5.1), or the last active level is used. The default is exponential selection.

Additionally, we need to specify parameters for the two primitives of restriction sampling and path section search. For unified restriction sampling, we use a path restriction sampling bias of 0.5, a path restriction surrounding bias of 0.1, and a sampling perturbation value of 0.05. For unified path section search, we use a maximum branching of 2, a maximum depth of 5, a restriction accuracy of 0.1, a number of propagation attempts of 5, and a maximum number of permutations of 10.

We run each planner in each benchmark for 10 runs and collect the time when a first feasible solution has been found. The results are visualized using a success graph, which shows the success percentage over runtime. Please see Tab. 3 for the properties of each scenario.

[†] https://github.com/aorthey/ompl/tree/FibrationTrees
[‡] https://github.com/aorthey/FibrationTrees

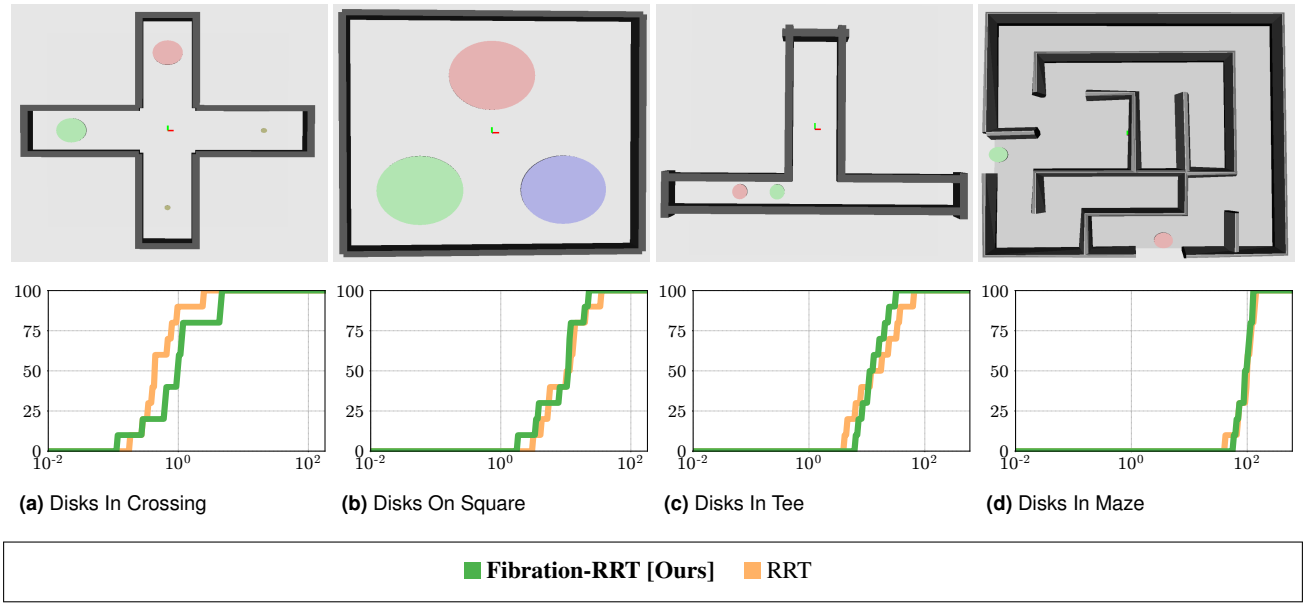


Figure 5. Overhead Benchmarks. Success graphs of the benchmarks for the Multi-Disks scenarios where we plan in component space (Fibration tree has a single root node which represents the component state space). Top row shows the scenarios, bottom row shows the success graphs, whereby the x-axis shows time in log-space, while the y-axis shows the success rate from 0 to 100 percent. Colors indicate the planner as shown in the legend above.

6.2 Overhead Benchmark

In the first four scenarios, we like to establish a baseline to show that Fibration-RRT performs similar to RRT when no fibration trees are used. This should show that the overhead of using of fibration trees is minimal. We call those experiments the overhead experiments.

The scenarios are shown in Fig. 5 (top row). Each scenario is a multi-disk planning problem, inspired by problems in the multi-robot literature (Solovey et al. 2016a; Orthey and Toussaint 2021c). In scenario Disks in Crossing (Fig. 5 (a)), two disks need to traverse a crossing to the opposite side while avoiding collision with themselves. In scenario Disks on Square (Fig. 5 (b)), three larger disks are located inside a square and have to move to the opposite side of their start location. The green disks has to move to the upper right corner, the blue disks to the upper left corner, and the right disk to the bottom center position. In scenario Disks in Tee (Fig. 5 (c)), two disks have to move from the left to the right side of a Tee-shaped structure. However, both disks have to exchange their positions to reach their respective goal configurations. Finally, in scenario Disks in Maze (Fig. 5 (d)), two disks are located on opposite ends of a maze and have to exchange positions by navigating the maze while avoiding collisions with each other.

6.2.1 Results The results for Fibration-RRT and RRT are shown in Fig. 5 (bottom row). The scenarios are ordered in ascending difficulty, meaning it takes, on average, longer to solve (d) than it takes to solve (a). It can be seen that there is a small, visible difference for low runtimes in (a), but both algorithms perform relatively similar in (b), (c), and (d).

The results show that there seems to be a small, measurable overhead for low runtimes, but that it vanishes for higher runtimes and does not constitute a bottleneck.

6.3 Structural Benchmark

In the structural benchmark, we compare Fibration-RRT with different fibration trees as input to a suite of single-tree planners. This benchmark uses eight multi-robot scenarios as depicted in Fig. 6. We compare Fibration-RRT with a decomposition-based fibration tree (Fibration-RRT-Decomposition) and Fibration-RRT with a prioritization-based fibration tree (Fibration-RRT-Prioritization). Additionally, we compare those two planners to the single-query, single-tree planners rapidly-exploring random tree (RRT) (Lavalle 1998), expansive space-trees (EST) (Hsu et al. 1999), fast marching trees (FMT) (Janson et al. 2015), and lower bound tree-RRT (LBT-RRT) (Salzman and Halperin 2016).

6.3.1 Scenarios The eight scenarios consists of the following problems. In Multi Disks (Fig. 6 (a)), we have a scenario of eight disk robots (2-dof) with four on the left and four on the right of a plane. Both sets of disks have to change their position with the opposite disk. A single cylindrical obstacle in the middle of the plane has to be avoided. This scenario has a total of 16-dof.

In Airship Coordination (Fig. 6 (b)), eight flying airship robots are given which can move freely in space while rotating around its Z-axis (4-dof). Those airships have to fly from one side of the plane to the other while avoiding collisions with themselves and with multiple objects in the environment. This scenario has a total of 32-dof.

In Mobile Navigation (Fig. 6 (c)), a total of six mobile manipulator robots are given. Each robot consists of a base (3-dof) and a manipulator arm (7-dof) for a total of 10-dof. Two robots are tasked to move from the right side to the left, while the other robots have to move from the left side to the right. The goal positions for the robots are indicated by the transparent yellow robot (showing only base position for visibility). This scenario has a total of 60-dof.

Table 3. Overview of the properties of each scenario used for benchmarking. The properties include if a scenario is homogeneous (all robots are the same), heterogeneous (robots differ), task-constrained (IK solver required), space-time (planning includes a time dimensions and dynamic obstacles), and if the fibration trees used includes parallel, sequential, or partial fibrations. For each scenario, we note if a certain property exists (green), or does not exists (grey).

Scenario	Homogeneous	Heterogeneous	Task-Constraint	Space-Time	Parallel	Sequential	Partial
Multi Disks (16-dof)	■				■	■	
Zeppelin Control (32-dof)	■				■	■	
Mobile Navigation (60-dof)	■				■	■	
Reeds Sheep (12-dof)	■				■	■	
Cube Robots (24-dof)		■			■	■	
Drones In Pipe (96-dof)	■				■	■	
Forest (40-dof)	■				■	■	
Warehouse (25-dof)		■			■	■	
Fixed Manipulator (7-dof)	■		■				■
Fixed on Wall (14-dof)	■		■		■		■
Mobile Manipulators (30-dof)	■		■		■		■
Path Velocity Decomposition (11-dof)	■		■	■		■	■

In Reeds Shepp (Fig. 6 (d)), four cars are given, which are modelled as Reeds Shepp cars (Reeds and Shepp 1990) (3-dof). The tasks consists of a parking lot, where two cars have to leave while two cars have to enter and reach the previous occupied parking spots. This scenario has a total of 12-dof.

In Cube Robots (Fig. 6 (e)), a total of nine cubes are given, which can translate and rotate on the plane (3-dof each). The cubes have different sizes, and need to move each from one side of the plane to the opposite side while avoiding collisions. This scenario has a total of 24-dof.

In Drones in Pipe (Fig. 6 (f)), we have a scenario of sixteen drone robots which can freely translate and rotate in space (6-dof). The task requires the drones to fly through a narrow pipe with metal lattices. This scenario has a total of 96-dof.

In Forest (Fig. 6 (g)), we use four mobile manipulator robots (as in Mobile Navigation) which have to trade places with each other in a diagonal fashion. The difficulty here consists of a narrow passage environment where multiple tall cuboids are blocking the robots. This scenario has a total of 40-dof.

Finally, in Warehouse (Fig. 6 (h)), we use two forklifts (3-dof) and three lifting trucks (3-dof) which have to reach specific positions inside a warehouse. This scenario has a total of 15-dof.

6.3.2 Results Fig. 7 shows the results for this benchmark. It can be seen that Fibration-RRT achieves a success rate of over 80 percent in 12 out of 12 cases and reaches a 100% success rate in 7 cases—given the time limits of each scenario. The planners RRT, EST, FMT, and LBTRRT achieve a success rate of over 80 percent on 3 out of 8 cases and reach a 100% success rate in 1 case, namely FMT for the Cube Robots (Fig. 7e).

In the cases where Fibration-RRT reaches 100% success rate, it outperforms the remaining planners in 5 cases by at least one order of magnitude in reaching the 100% success rate. This is the case for Multi Disks (Fig. 7a), Airship Coordination (Fig. 7c), Reeds-Shepp Simple (Fig. 7d), Drones in Pipe (Fig. 7f), and Warehouse (Fig. 7h).

6.4 Prioritization Benchmark

In this benchmark, we compare Fibration-RRT to the quotient-space RRT (QRRT) (Orthey and Toussaint 2022; Orthey et al. 2024a), by comparing a prioritization-based multi-robot motion planner to Fibration-RRT using a prioritized fibration tree. We compare the two planners on the same set of multi-robot scenarios as for the structural benchmark (see Fig. 6). Ideally, the benchmarks should show that Fibration-RRT performs equivalent to QRRT, while having more flexibility in the input. To compare both algorithms, we ensured that QRRT uses the same path section method, same selection method (exponential importance criterion), and same restriction sampler with equivalent parameters. Note that QRRT is just one prioritization-based algorithm which we use to showcase that Fibration-RRT can handle such scenarios in a similar fashion.

6.4.1 Results The results are shown in Fig. 8. Both algorithms, Fibration-RRT and QRRT, perform relatively similar on the first six scenarios (a–f), where there is no clear advantage for either one of them. However, in the last two scenarios (g, h), Fibration-RRT outperforms QRRT slightly by reaching a success rate of 100% inside the time budget, while QRRT fails to do this.

6.5 Decomposition Benchmark

In the decomposition benchmark, we compare Fibration-RRT using a decomposition-based fibration tree (a single parallel fibration onto the individual robot spaces) with the decomposition-based multi-robot discrete RRT (dRRT) planner (Solovey et al. 2016b). The scenarios are the same as for the structural and prioritization benchmark (see Fig. 6).

To guarantee a fair comparison, we created four different versions of dRRT because of one important distinction between Fibration-RRT and dRRT. While Fibration-RRT uses a selection function (Sec. 5.1) to decide with which planner to continue, dRRT has a fixed selection sequence. This sequence first computes roadmaps on the individual spaces until a fixed number of vertices has been generated or a time budget has been reached (Solovey et al. 2016b). After those roadmaps have been generated, the planner invokes the dRRT coordination method which explores the (implicit) tensor graph of the individual roadmaps. It is therefore important to choose a good time budget for the roadmap generation phase of dRRT. To have a fair comparison, we opted to include four different values, namely time budgets of 1s, 5s, 10s, and 50s. We call the corresponding planners dRRT-1, dRRT-5, dRRT-10, and dRRT-50, respectively.

6.5.1 Results The results are shown in Fig. 9. Fibration-RRT is able to solve six scenarios with a success rate of over 90% while only having a success rate of 20% for the remaining two scenarios, namely Drones in Pipe (Fig. 9f) and Mobile Robots Forest (Fig. 9g). dRRT has a success rate of over 90% for three scenarios (a, d, e), reaches 50% for one scenario (c), but fails for the remaining four scenarios.

6.6 Task-Space Benchmark

In this benchmark, we evaluate Fibration-RRT on scenarios containing task-space constraints on the end-effector of the robots. To compare this planner to classical planners, we implement a task-space version of RRT (Task-RRT). Task-RRT extends RRT in two aspects. First, a task-space sampler is implemented, which generates samples which are constrained on the respective task-space. For example, for the Vertical Maze scenario (see below), samples are generated via an inverse kinematics (IK) sampler who constraints the end-effector to lie close to the surface, consistent with the task constraint used by Fibration-RRT. Second, we use a forward propagation function, which moves the robot towards a target sample in such a way that its end-effector is guaranteed to stay on the task-space. This is accomplished using a Jacobian-based method (Craig 2005) to move the robot. Thus, the planner is similar to the task-space region RRT (Berenson et al. 2011), but uses an explicit representation of the task manifold for sampling (Kingston et al. 2019).

6.6.1 Scenarios The task-space constraints in those four scenarios are limitations on the movement of the Tool-center-point (Tcp). In scenario Vertical Maze, a fixed manipulator robot is tasked to move its Tcp through a maze while keeping the Tcp close to the surface of the wall as shown in Fig. 10 (a). This is modeled using a single partial fibration as shown in Fig. 11. The projection is defined as $\pi : X \to T$ with X being the joint space, and T being the position of the Tcp of the manipulator. The projection mapping itself is given by the forward kinematics from joint space to Tcp space. The lift of the projection is given by solving an inverse kinematics problem with a random seed. This is an admissible projection, since a valid joint space configuration has by definition a valid Tcp position.

In scenario Fixed on Wall (Fig. 10 (b)) , we combine parallel and partial fibrations by having two fixed manipulator robots in close proximity, which both need to cross to reach goal points along the wall. Both robots are constrained to keep the Tcp at all times close to the wall surface. This scenario is modeled using a fibration tree consisting of a parallel fibration and two partial fibrations (see Fig. 11).

The third scenario is Mobile Manipulators (Fig. 10 (c)), which consists of three mobile manipulators which have different Tcp constraints, whereby two manipulators need to follow a straight line along the coordinate axes at a height of 0.5m, while the third robot needs to keep its Tcp at a height of 0.5m, but not necessarily along a straight line. We model this using a parallel, and three partial fibration as shown in Fig. 11.

In the last scenario, Path Velocity Decomposition (Fig. 10 (d)), we use a time-based state space with a mobile manipulator which has to cross over a floor segment, where it faces both static and dynamic obstacles. We also constrain the manipulator to keep its Tcp at a height of 0.7 ± 0.05m above the floor. This scenario is modeled using a fibration tree with a single sequential fibration from the time-based state space to a time-less state space, plus a partial fibration onto the task-constraint (see Fig. 11).

To find motions which are consistent with time, we require two modifications to the steering method (Kant and Zucker 1986; Grothe et al. 2022). First, we need to ensure monotonicity, i.e. a connection between two elements (s_1, t_1) and (s_2, t_2) is only possible if $t_2 > t_1$. Second, we cannot move arbitrarily fast, i.e. we have to ensure that a maximum velocity is respected. This is achieved by constraining the slope $\left|\frac{ds}{dt}\right| < V_{max}$. For this scenario, the mobile robot has $V_{max} = 1.0$.

6.6.2 Results The results are shown in Fig. 10. They show that Fibration-RRT achieves a success rate of over 80 percent in 4 out of 4 cases and reaches a 100% success rate in 2 cases. TaskRRT achieves a success rate of over 80 percent on 2 out of 4 cases and reach a 100% success rate in only one case for the Mobile Manipulators (Fig. 10c).

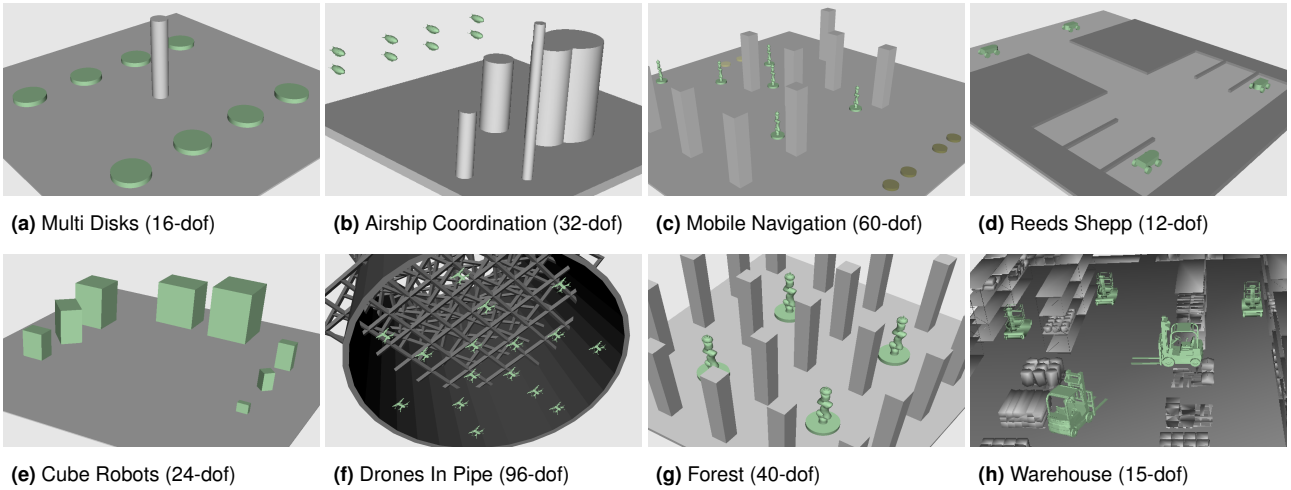
In the cases where Fibration-RRT reaches 100% success rate, it outperforms Task-RRT in 2 cases by at least one order of magnitude in reaching the 100% success rate. This is the case for Vertical Maze (Fig. 10a) and the Mobile Manipulators (Fig. 10c).

7 Discussion and Limitations

Let us discuss the results from the benchmarks in Sec. 6. We compared Fibration-RRT on different scenarios to showcase the flexibility and unified nature of fibration trees.

The results indicate that both prioritization-based and decomposition-based approaches are valuable structures for multi-robot motion planning. While approaches which plan purely in the composite state space have problems finding solutions, we were able to show that Fibration-RRT can exploit both structures and reach results which improve runtime significantly (Sec. 6.3). This is in line with previous results on multi-robot motion planning, where decomposition-based approaches (Wagner and Choset 2015a; Shome et al. 2020; Dobson et al. 2017; Guo et al. 2026) and prioritization-based approaches (Erdmann and Lozano-Perez 1987; van den Berg and Overmars 2005; Orthey et al. 2024a) have shown similar results and significant improvement on runtime. This again underlines that Fibration-RRT can efficiently exploit a given fibration tree structure.

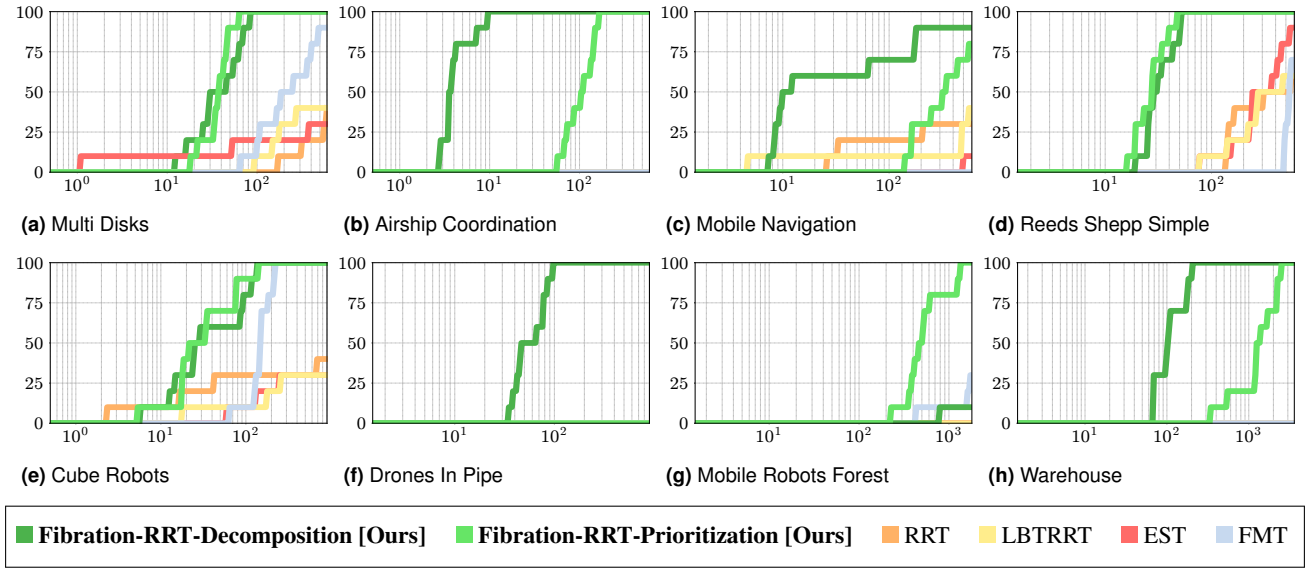
A rather surprising result is the advantage of the decomposition-based approach compared to a prioritization-based approach. While both approaches perform similar in the structure benchmark (Sec. 6.3) on scenarios (a,d,e), there is a significant difference between them on the remaining scenarios with the decomposition-based approach outperforming the prioritization-based approach on (b,c,f,h), while the opposite is true for scenario (g). We believe this could be caused by some scenarios having fewer inter-robot collisions which have to be resolved (which would be an advantage for decomposition-based approaches because they would not need to resolve them). However, scenario (g) consists of a difficult narrow-passage environment where most single-robot movements would results in inter-robot collisions. In those types of scenarios, resolving robot paths one by one seems to be advantageous to find a solution quicker.



(a) Multi Disks (16-dof) **(b)** Airship Coordination (32-dof) **(c)** Mobile Navigation (60-dof) **(d)** Reeds Shepp (12-dof)

(e) Cube Robots (24-dof) **(f)** Drones In Pipe (96-dof) **(g)** Forest (40-dof) **(h)** Warehouse (15-dof)

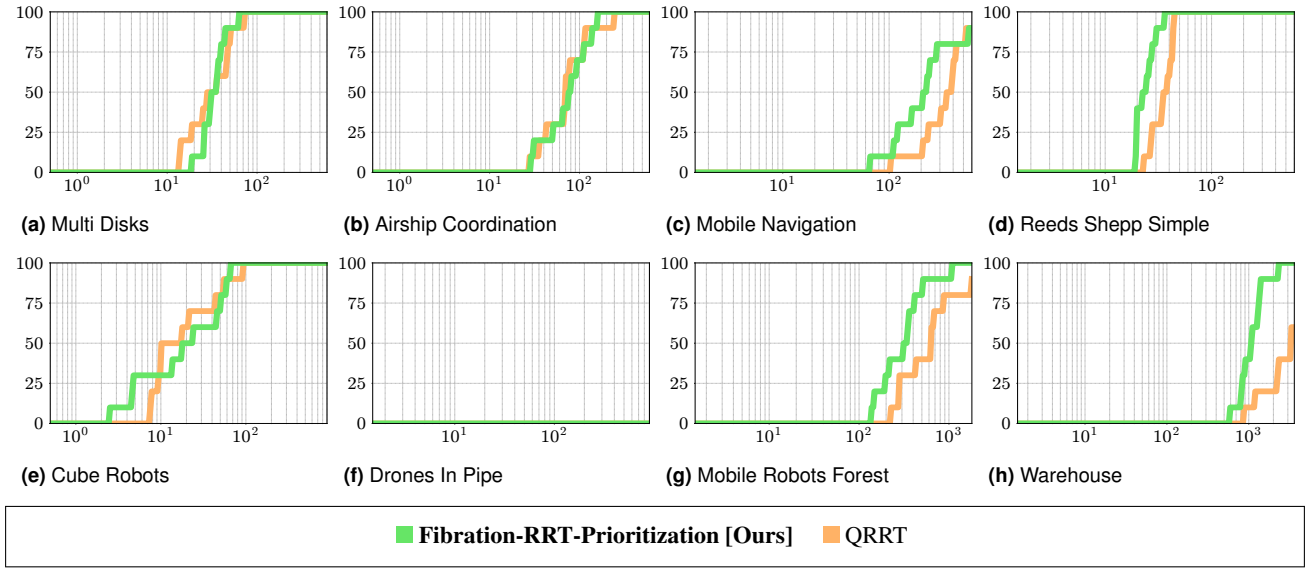
Figure 6. Experiment used for benchmarking. Movable robots are depicted in light green, obstacles in gray, and movable obstacles in light red. First and second row are multi-robot navigation scenarios, while the third row uses task constraints on the endeffector and dynamic obstacles in space-time.



(a) Multi Disks **(b)** Airship Coordination **(c)** Mobile Navigation **(d)** Reeds Shepp Simple

(e) Cube Robots **(f)** Drones In Pipe **(g)** Mobile Robots Forest **(h)** Warehouse

Figure 7. Success graphs of the benchmarks for scenarios from Fig. 6. The x-axis shows time in log-space, while the y-axis shows the success rate from 0 to 100 percent. Colors indicate the planner as shown above.



(a) Multi Disks **(b)** Airship Coordination **(c)** Mobile Navigation **(d)** Reeds Shepp Simple

(e) Cube Robots **(f)** Drones In Pipe **(g)** Mobile Robots Forest **(h)** Warehouse

Figure 8. Success graphs for the prioritization benchmarks.

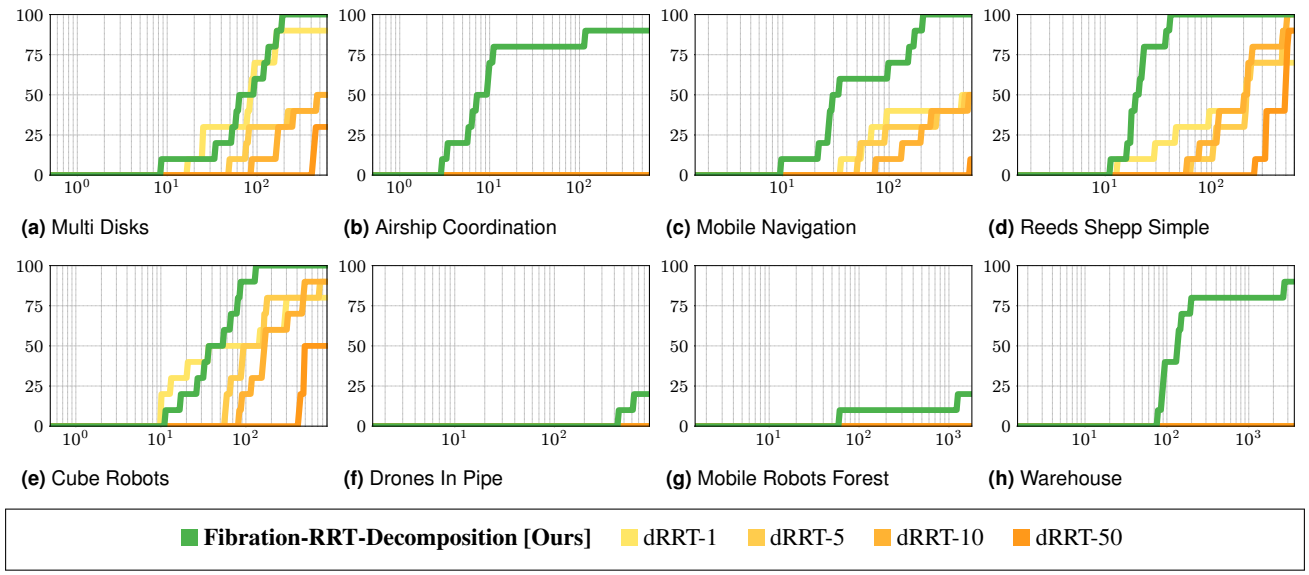


Figure 9. Success graphs for the decomposition benchmarks.

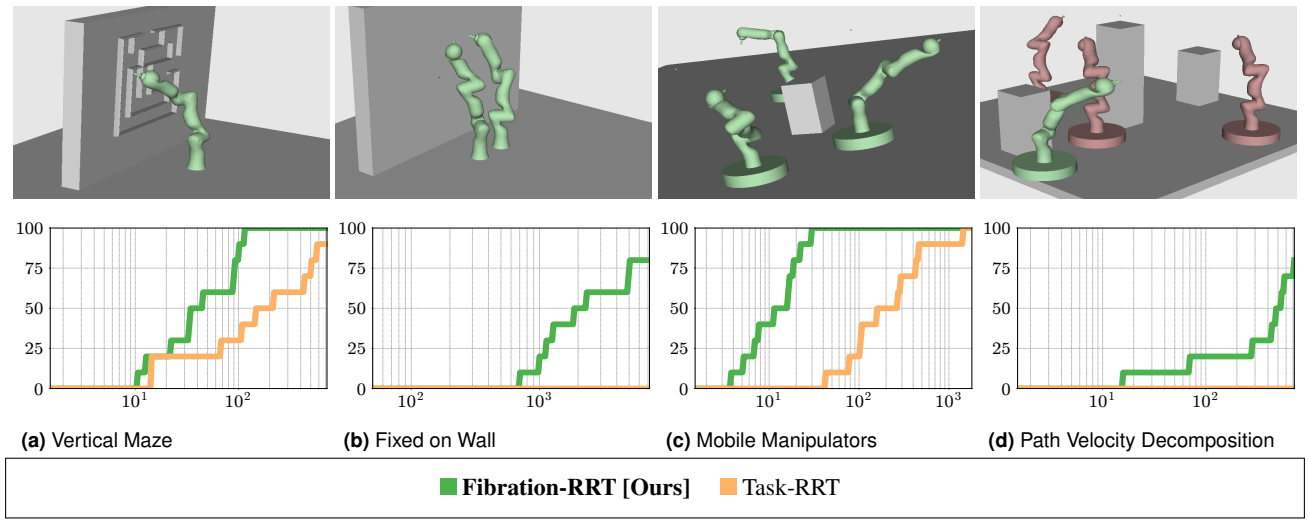


Figure 10. Success graphs of the benchmarks for scenarios from Fig. 6. The x-axis shows time in log-space, while the y-axis shows the success rate from 0 to 100 percent. Colors indicate the planner as shown above.

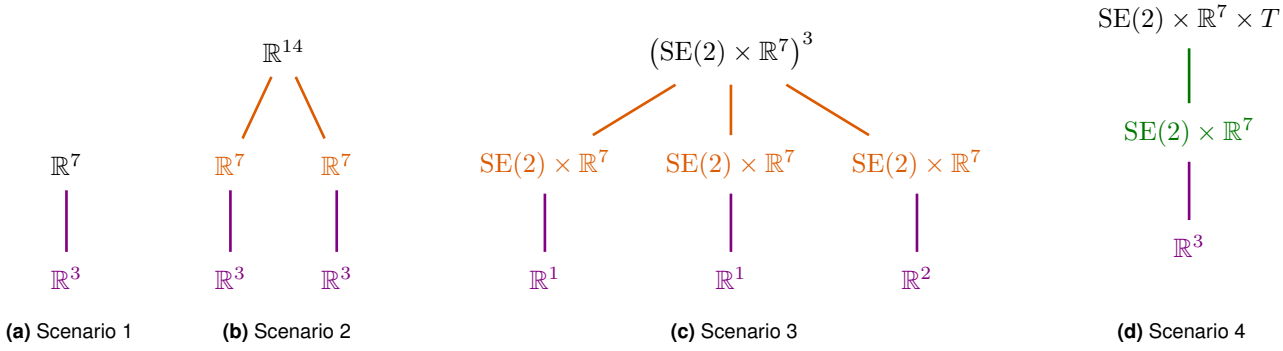


Figure 11. Fibration trees used for task-space evaluations. Colors indicate type of projection: Parallel ■, Sequential ■, and Partial ■.

7.1 Limitations

While we showed Fibration-RRT to perform well on all scenarios, there are still several improvements possible.

Handling of variable-volumetric fibers In restriction sampling, we handle all fibers as if they would have the same volume. Often, however, the volume of fibers might differ significantly and could lead to a highly non-uniform distribution of samples. A classical example would be a partial fibration where an inverse IK solver lifts workspace points to the configuration space. The fibers involved could

represent subsets of the full configuration space, lower-dimensional manifolds, or even zero sets. To handle such fibrations, we could add an additional distance check, which estimates the volume of a fiber by comparing samples lying on a single fiber. Depending on this check, we could adjust the global sampling distribution for graph- and path-restriction sampling accordingly.

Parallel execution of parallel fibrations Nodes in a parallel fibrations are usually independent, and could be computed in parallel to speed the method up. This could be handled by analyzing the fibration tree and creating a new thread for each decomposition split. This would create a set of independent subtrees, which each can then be run in parallel.

Integration of specialized solutions While we propose a unified framework, there are some cases which can be solved using specialized methods. This includes:

- The pebbles-on-a-graph reduction for parallel fibrations with identical base spaces (homogeneous decompositions) (Kornhauser et al. 1984; Yu and LaValle 2016; Ryan 2010; Luna and Bekris 2011).
- Implicit graph search for parallel fibrations with non-identical base spaces (heterogeneous decompositions) (Wagner and Choset 2015b; Shome et al. 2020).
- Advanced path section methods for sequential fibrations (Grey et al. 2017b; Reid et al. 2019; Orthey and Toussaint 2021a; Ichter and Pavone 2019).
- Reciprocal velocity obstacles for simple robot shapes in path-velocity decompositions (van den Berg et al. 2011).
- Specialized planners for path-velocity decompositions, like the admissible velocity propagation (AVP) method (Pham et al. 2017) to forward propagate admissible velocity profiles, or the space-time RRT (ST-RRT*) (Grothe et al. 2022) to handle state spaces with unbounded time.

Eventually, we aim to integrate all those specialized solutions as modules into our framework to increase the efficiency of Fibration-RRT, and to make it a one-stop solution for any multi-robot motion planning problem.

Infeasible Problems Fibration-RRT can currently only deal with feasible problems, but it cannot finish in finite time on an infeasible problem. Recently, we used sparse roadmaps to find probabilistic guarantees that a problem is infeasible (Orthey and Toussaint 2021b). A combination of Fibration-RRT with sparse planners (Siméon et al. 2000; Dobson and Bekris 2014) would be an effective extension to tackle infeasible problems. Another valuable approach would be to investigate exact infeasibility proofs in this context (Li and Dantam 2023), which could be beneficial for lower-dimensional projection mappings.

Asymptotically Optimal Planner While Fibration-RRT does not guarantee asymptotic optimality (AO) (Gammell and Strub 2021), there is in theory nothing preventing us from extending it into an AO planner. However, while this is a fruitful future venue to explore, it is beyond the scope of the current paper, since the focus is on the general structure supporting multi-robot planners and how we can efficiently exploit them.

Additional Heuristics Fibration-RRT could benefit from additional heuristics like informed sets (Gammell et al. 2020). However, the application is not straightforward, because pruning of states might prune edges belonging to the path restriction of the global optimal solution in the total space (root node of fibration tree). A better communication between nodes could alleviate this problem by propagating solution costs downwards.

Informative Selection The current implementation of Fibration-RRT selects factors for expansion based on the exponential selection criterion as described in Sec. 5.1. However, as previous works have shown (Orthey and Toussaint 2022, 2021b), different selection methods could improve the planning time. In future work, we envision a selection unit, which collects data about all runs of all nodes in the fibration tree. In a subsequent step, this selection unit uses the collected data to choose a node which has the most utility of leading to a successful outcome. Such a system could also be learned from previous planning data.

Interpolation and Differentiability The framework of fibration trees also provides us with the opportunity to further exploit path restrictions for more efficient planning. We envision two forms of this. First, interpolation methods could exploit valid paths on a base space as additional information. This information could be used to generate more valid paths on the bundle space in comparison to geodesic interpolation (Orthey et al. 2018). Second, graph restrictions could be used to add differentiable information to a planner similar to the artificial potential field approach (Khatib 1986). This is left for future work.

8 Conclusion

We proposed fibration trees, a unified approach to simplify and decompose high-dimensional, multi-robot motion planning problems. Our formulation allows us to directly compare prioritization-based and decomposition-based approaches and create a single algorithm (Fibration-RRT) which is independent of the underlying structure. In our benchmarks, we demonstrated that Fibration-RRT has minimal overhead compared to RRT, that it can handle fibration trees of different kinds, and that we can use Fibration-RRT to directly compare prioritization-based and decomposition-based approaches. We also showed that Fibration-RRT performs similar to prioritization-based approaches on prioritization structures, and can outperform decomposition-based approaches when using a decomposition structure. Finally, we showed that task-space constraints can be integrated into fibration trees as another fibration type. This comprehensive evaluation on 32 scenarios shows that Fibration-RRT can tackle a wide variety of scenarios and fibration trees, thereby underlying the unified nature of fibration trees for multi-robot motion planning problems.

We believe that fibration trees will not only simplify future benchmarking of multi-robot planners, but could also lay the foundation for a unified multi-robot planning method which is independent of the underlying structure. This could establish fibration trees as a general-purpose framework for multi-robot planning problems.

References

Aggarwal CC, Hinneburg A and Keim DA (2001) On the surprising behavior of distance metrics in high dimensional space. In: *International Conference on Database Theory*. Springer, pp. 420–434.

Atias A, Solovey K, Salzman O and Halperin D (2018) Effective metrics for multi-robot motion-planning. *International Journal of Robotics Research* 37(13-14): 1741–1759.

Bennewitz M, Burgard W and Thrun S (2001) Optimizing schedules for prioritized path planning of multi-robot systems. In: *IEEE International Conference on Robotics and Automation*, volume 1. IEEE, pp. 271–276.

Berenson D, Srinivasa S and Kuffner J (2011) Task space regions: A framework for pose-constrained manipulation planning. *International Journal of Robotics Research* 30(12): 1435–1460.

Bettini M, Prorok A and Moens V (2024) Benchmarl: Benchmarking multi-agent reinforcement learning. *Journal of Machine Learning Research* 25(217): 1–10.

Bettini M, Shankar A and Prorok A (2023) Heterogeneous multi-robot reinforcement learning. In: *IEEE International Symposium on Multi-Robot and Multi-Agent Systems*. pp. 1485–1494.

Bhattacharya S and Ghrist R (2018) Path homotopy invariants and their application to optimal trajectory planning. *Annals of Mathematics and Artificial Intelligence* 84(3-4): 139–160.

Canny J (1988) *The complexity of robot motion planning*. MIT press.

Čáp M, Novák P, Kleiner A and Seleckỳ M (2015) Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Transactions on Automation Science and Engineering* 12(3): 835–849.

Choset H, Lynch KM, Hutchinson S, Kantor GA, Burgard W, Kavraki LE and Thrun S (2005) *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Intelligent Robotics and Autonomous Agents. Cambridge, MA: MIT Press. ISBN 0-262-03327-5.

Cormen TH, Leiserson CE, Rivest RL and Stein C (2022) *Introduction to algorithms*. MIT press.

Craig JJ (2005) *Introduction to Robotics: Mechanics and Control*. 3rd edition. Upper Saddle River, NJ: Pearson Prentice Hall.

Dobson A and Bekris KE (2014) Sparse roadmap spanners for asymptotically near-optimal motion planning. *International Journal of Robotics Research* 33(1): 18–47.

Dobson A, Solovey K, Shome R, Halperin D and Bekris KE (2017) Scalable asymptotically-optimal multi-robot motion planning. In: *IEEE International Symposium on Multi-Robot and Multi-Agent Systems*. IEEE, pp. 120–127.

Dorigo M, Theraulaz G and Trianni V (2021) Swarm robotics: Past, present, and future [point of view]. *IEEE Robotics and Automation Magazine* 28(2): 12–19.

Erdmann M and Lozano-Perez T (1987) On multiple moving objects. *Algorithmica* 2(1-4): 477.

Ferbach P and Barraquand J (1997) A method of progressive constraints for manipulation planning. *IEEE Transactions on Robotics* 13(4): 473–485.

Gammell JD, Barfoot TD and Srinivasa SS (2020) Batch informed trees (BIT*): Informed asymptotically optimal anytime search. *International Journal of Robotics Research* 39(5): 543–567.

Gammell JD and Strub MP (2021) A survey of asymptotically optimal sampling-based motion planning methods. *Annual Review of Control, Robotics, and Autonomous Systems* 4(1): 295–318.

Grey MX, Ames AD and Liu CK (2017a) Footstep and motion planning in semi-unstructured environments using randomized possibility graphs. In: *IEEE International Conference on Robotics and Automation*. pp. 4747–4753.

Grey MX, Ames AD and Liu CK (2017b) Footstep and motion planning in semi-unstructured environments using randomized possibility graphs. In: *IEEE International Conference on Robotics and Automation*. pp. 4747–4753.

Grothe F, Hartmann VN, Orthey A and Toussaint M (2022) St-rrt*: Asymptotically-optimal bidirectional motion planning through space-time. In: *IEEE International Conference on Robotics and Automation*. pp. 3314–3320.

Guo W, Kingston Z, Hang K and Kavraki LE (2026) Efficient multi-robot motion planning for manifold-constrained manipulators by randomized scheduling and informed path generation. *IEEE Robotics and Automation Letters* To Appear.

Ha H, Xu J and Song S (2020) Learning a decentralized multi-arm motion planner. In: *Conference on Robot Learning*, *Proceedings of Machine Learning Research*, volume 155. PMLR, pp. 103–114. URL `https://proceedings.mlr.press/v155/ha21a.html`.

Hartmann VN, Orthey A, Driess D, Oguz OS and Toussaint M (2021) Long-horizon multi-robot rearrangement planning for construction assembly. *IEEE Transactions on Robotics* 39(1): 239–252.

Hastie T, Tibshirani R and Friedman J (2009) *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

Hatcher A (2002) *Algebraic Topology*. Cambridge: Cambridge University Press. ISBN 9780521795401. URL `https://pi.math.cornell.edu/~hatcher/AT/ATpage.html`.

Heselden JR and Das GP (2023) Heuristics and rescheduling in prioritised multi-robot path planning: a literature review. *Machines* 11(11): 1033.

Hönig W, Preiss JA, Kumar TKS, Sukhatme GS and Ayanian N (2018) Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics* 34(4): 856–869.

Hopcroft JE, Schwartz JT and Sharir M (1984) On the complexity of motion planning for multiple independent objects; pspace-hardness of the" warehouseman's problem". *International Journal of Robotics Research* 3(4): 76–88.

Hopf H (1931) Über die Abbildungen der dreidimensionalen Sphäre auf die Kugelfläche. *Mathematische Annalen* 104(1): 637–665.

Hsu D, Latombe JC and Motwani R (1999) Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications* 9(4-5): 495–512.

Hu G, Li H, Liu S, Ma M, Zhu Y and Zhao D (2023) Neuronsmae: A novel multi-agent reinforcement learning environment for cooperative and competitive multi-robot tasks. In: *International Joint Conference on Artificial Intelligence*. IEEE, pp. 1–8. DOI:10.1109/IJCNN54540.2023.10191527.

Ichter B and Pavone M (2019) Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters* 4(3): 2407–2414.

Janson L, Schmerling E, Clark A and Pavone M (2015) Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *International Journal of Robotics Research* 34(7): 883–921.

Jiang C, Cornman A, Park C, Sapp B, Zhou Y and Anguelov D (2023) Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In: *Conference on Computer Vision and Pattern Recognition*. pp. 9644–9653.

Judson TW (2023) *Abstract Algebra: Theory and Applications*. 2023 edition. Ann Arbor, MI: Orthogonal Publishing L3C. ISBN 9781944325190.

Kaiser TK and Hamann H (2022) Innate motivation for robot swarms by minimizing surprise: From simple simulations to real-world experiments. *IEEE Transactions on Robotics* 38(6): 3582–3601.

Kant K and Zucker SW (1986) Toward efficient trajectory planning: The path-velocity decomposition. *International Journal of Robotics Research* 5(3): 72–89.

Kavraki LE and LaValle SM (2016) *Motion Planning*. Springer. DOI:10.1007/978-3-319-32552-1_7.

Kavraki LE, Svestka P, Latombe JC and Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics* 12(4): 566–580.

Keisuke O and Xavier D (2023) Quick multi-robot motion planning by combining sampling and search. *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence* : 252–261DOI:10.24963/ijcai.2023/29. URL https://cir.nii.ac.jp/crid/1360302864784508032.

Keppler F and Wagner S (2020) Prioritized multi-robot velocity planning for trajectory coordination of arbitrarily complex vehicle structures. In: *2020 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, pp. 1075–1080.

Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. In: *Autonomous Robot Vehicles*. Springer, pp. 396–404.

Kingston Z, Moll M and Kavraki LE (2019) Exploring implicit spaces for constrained sampling-based planning. *International Journal of Robotics Research* 38(10–11): 1151–1178. DOI: 10.1177/0278364919868530.

Kornhauser D, Miller GL and Spirakis P (1984) Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In: *Symposium on the Foundations of Computer Science*. IEEE, pp. 241–250.

Kuffner JJ and LaValle SM (2000) RRT-connect: An efficient approach to single-query path planning. In: *IEEE International Conference on Robotics and Automation*, volume 2. pp. 995–1001.

Lai M, Go K, Li Z, Kröger T, Schaal S, Allen K and Scholz J (2025) Roboballet: Planning for multirobot reaching with graph neural networks and reinforcement learning. *Science Robotics* 10(106): eads1204.

Lavalle SM (1998) Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University.

LaValle SM (2006) *Planning Algorithms*. Cambridge University Press.

Lee J, Grey MX, Ha S, Kunz T, Jain S, Ye Y, Srinivasa SS, Stilman M and Liu CK (2018a) DART: Dynamic animation and robotics toolkit. *The Journal of Open Source Software* 3(22): 500. DOI:10.21105/joss.00500. URL https://doi.org/10.21105/joss.00500.

Lee J, Grey MX, Ha S, Kunz T, Jain S, Ye Y, Srinivasa SS, Stilman M and Liu CK (2018b) Dart: Dynamic animation and robotics toolkit. *Journal of Open Source Software* 3(22): 500.

Lee JM (2003) *Introduction to Smooth Manifolds*. New York, NY: Springer New York.

Li S and Dantam NT (2023) A sampling and learning framework to prove motion planning infeasibility. *International Journal of Robotics Research* 42(10): 938–956. DOI:10.1177/02783649231154674. URL https://doi.org/10.1177/02783649231154674.

Liang J, Christopher JK, Koenig S and Fioretto F (2025) Simultaneous multi-robot motion planning with projected diffusion models. *arXiv preprint arXiv:2502.03607* .

Lozano-Perez T (1983) Spatial planning: A configuration space approach. *IEEE Transactions on Computers* 2(C-32): 108–120.

Lozano-Pérez T and Wesley MA (1979) An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM* 22(10): 560–570.

Luna R and Bekris KE (2011) Efficient and complete centralized multi-robot path planning. In: *IEEE International Conference on Intelligent Robots and Systems*. pp. 3268–3275.

Luna R, Moll M, Badger J and Kavraki LE (2020) A scalable motion planner for high-dimensional kinematic systems. *International Journal of Robotics Research* 39(4): 361–388.

Ma H, Harabor D, Stuckey PJ, Li J and Koenig S (2019) Searching with consistent prioritization for multi-agent path finding. In: *Proceedings of the AAAI conference on artificial intelligence*, volume 33. pp. 7643–7650.

McBeth C, Motes J, Uwacu D, Morales M and Amato NM (2023) Scalable multi-robot motion planning for congested environments with topological guidance. *IEEE Robotics and Automation Letters* 8(11): 6867–6874. DOI:10.1109/LRA.2023.3312980.

Moldagalieva A, Ortiz-Haro J, Toussaint M and Hönig W (2024) db-cbs: Discontinuity-bounded conflict-based search for multi-robot kinodynamic motion planning. In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 14569–14575.

Moll M, Şucan IA and Kavraki LE (2015) Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization. *IEEE Robotics and Automation Magazine* 22(3): 96–102.

Nino CF, Petersen CD and Dixon WE (2025) Collaborative spacecraft servicing under partial feedback using lyapunov-based deep neural networks. *The Journal of the Astronautical Sciences* 72(1): 40. DOI:10.1007/s40295-025-00473-w. URL https://doi.org/10.1007/s40295-025-00473-w.

Orthey A, Akbar S and Toussaint M (2024a) Multilevel motion planning: A fiber bundle formulation. *The International Journal of Robotics Research* 43(1): 3–33. DOI:10.1177/02783649231209337. URL https://doi.org/10.1177/02783649231209337.

Orthey A, Chamzas C and Kavraki LE (2024b) Sampling-based motion planning: A comparative review. *Annual Review of Control, Robotics, and Autonomous Systems* 7(1). DOI:10.1146/annurev-control-061623-094742. URL https://doi.org/10.1146/annurev-control-061623-094742.

Orthey A, Escande A and Yoshida E (2018) Quotient-space motion planning. In: *IEEE International Conference on Intelligent Robots and Systems*. pp. 8089–8096.

Orthey A and Toussaint M (2021a) Section patterns: Efficiently solving narrow passage problems in multilevel motion planning. *IEEE Transactions on Robotics* : 1891–1905.

Orthey A and Toussaint M (2021b) Sparse multilevel roadmaps for high-dimensional robot motion planning. In: *IEEE International Conference on Robotics and Automation*. p. 7851–7857.

Orthey A and Toussaint M (2021c) Visualizing local minima in multi-robot motion planning using multilevel morse theory. In: LaValle SM, Lin M, Ojala T, Shell D and Yu J (eds.) *Algorithmic Foundations of Robotics XIV*. Cham: Springer International Publishing. ISBN 978-3-030-66723-, pp. 502–517.

Orthey A and Toussaint M (2022) Rapidly-exploring quotient-space trees: Motion planning using sequential simplifications. In: Asfour T, Yoshida E, Park J, Christensen H and Khatib O (eds.) *Robotics Research*. Cham: Springer International Publishing, pp. 52–68.

Parque V and Miyashita T (2023) Multi-Robot Coordinated Motion Planning at Intersections using Lattice-Guided dRRT* . In: *2023 Seventh IEEE International Conference on Robotic Computing (IRC)*. Los Alamitos, CA, USA: IEEE Computer Society, pp. 348–351. DOI:10.1109/IRC59093.2023.00062. URL https://doi.ieeecomputersociety.org/10.1109/IRC59093.2023.00062.

Pham QC, Caron S, Lertkultanon P and Nakamura Y (2017) Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots. *International Journal of Robotics Research* 36(1): 44–67.

Plaku E, Kavraki LE and Vardi MY (2010) Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Transactions on Robotics* 26(3): 469–482.

Reeds J and Shepp L (1990) Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics* 145(2): 367–393.

Reid W, Fitch R, Göktoğan AH and Sukkarieh S (2019) Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover. *Journal of Field Robotics* .

Reid W, Fitch R, Göktoğgan AH and Sukkarieh S (2020) Motion planning for reconfigurable mobile robots using hierarchical fast marching trees. In: Goldberg K, Abbeel P, Bekris K and Miller L (eds.) *Algorithmic Foundations of Robotics XII*. Springer, pp. 656–671.

Reif JH (1979) Complexity of the mover's problem and generalizations. In: *Conference on Foundations of Computer Science*. pp. 421–427.

Rickert M, Sieverling A and Brock O (2014) Balancing exploration and exploitation in sampling-based motion planning. *IEEE Transactions on Robotics* 30(6): 1305–1317.

Ryan M (2010) Constraint-based multi-robot path planning. In: *IEEE International Conference on Robotics and Automation*. pp. 922–928.

Salzman O and Halperin D (2016) Asymptotically near-optimal RRT for fast, high-quality motion planning. *IEEE Transactions on Robotics* 32(3): 473–483.

Schwartz JT and Sharir M (1983) On the piano movers' problem: Iii. coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *International Journal of Robotics Research* 2(3): 46–75.

Sebastián E, Duong T, Atanasov N, Montijano E and Sagüés C (2025) Physics-informed multi-agent reinforcement learning for distributed multi-robot problems. *IEEE Transactions on Robotics* .

Shaoul Y, Mishani I, Vats S, Li J and Likhachev M (2025) Multi-robot motion planning with diffusion models. In: *International Conference on Learning Representations*.

Sharon G, Stern R, Felner A and Sturtevant NR (2015) Conflict-based search for optimal multi-agent pathfinding. *Artificial intelligence* 219: 40–66.

Shome R, Solovey K, Dobson A, Halperin D and Bekris KE (2020) drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning. *Autonomous Robots* 44(3): 443–467.

Siméon T, Laumond JP and Nissoux C (2000) Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics* 14(6): 477–493.

Siméon T, Leroy S and Laumond JP (2002) Path coordination for multiple mobile robots: A resolution-complete algorithm. *IEEE Transactions on Robotics and Automation* 18(1): 42–49.

Solano A, Sieverling A, Gieselmann R and Orthey A (2024) Fast-drrt*: Efficient multi-robot motion planning for automated industrial manufacturing. *ArXiv* abs/2309.10665. URL https://api.semanticscholar.org/CorpusID:262054235.

Solis I, Motes J, Qin M, Morales M and Amato NM (2024) Adaptive robot coordination: A subproblem-based approach for hybrid multi-robot motion planning. *IEEE Robotics and Automation Letters* 9(8): 7238–7245.

Solovey K, Janson L, Schmerling E, Frazzoli E and Pavone M (2020) Revisiting the asymptotic optimality of rrt. In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 2189–2195.

Solovey K, Salzman O and Halperin D (2016a) Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. *International Journal of Robotics Research* 35(5): 501–513.

Solovey K, Salzman O and Halperin D (2016b) Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. *International Journal of Robotics Research* 35(5): 501–513.

Steenrod NE (1951) *The topology of fibre bundles*. Princeton Univ. Press.

Şucan IA and Kavraki LE (2011) A sampling-based tree planner for systems with complex dynamics. *IEEE Transactions on Robotics* 28(1): 116–131.

Şucan IA, Moll M and Kavraki L (2012) The open motion planning library. *IEEE Robotics and Automation Magazine* 19(4): 72–82.

Sun D and Liao Q (2025) Real-time coordination of multiple robotic arms with reactive trajectory modulation. *IEEE Transactions on Robotics* 41: 200–219.

Svestka P and Overmars MH (1995) Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In: *IEEE International Conference on Robotics and Automation*.

pp. 1631–1636. DOI:10.1109/ROBOT.1995.525508.

Svestka P and Overmars MH (1998) Coordinated path planning for multiple robots. *IEEE Robotics and Autonomous Systems* 23(3): 125–152.

Tonneau S, Prete AD, Pettré J, Park C, Manocha D and Mansard N (2018) An Efficient Acyclic Contact Planner for Multiped Robots. *IEEE Transactions on Robotics* 34(3): 586–601.

Tournassoud P (1986) A strategy for obstacle avoidance and its application to multi-robot systems. In: *IEEE International Conference on Robotics and Automation*, volume 3. IEEE, pp. 1224–1229.

van den Berg J, Guy SJ, Lin M and Manocha D (2011) Reciprocal n-body collision avoidance. In: Pradalier C, Siegwart R and Hirzinger G (eds.) *Robotics Research*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 3–19.

van den Berg J and Overmars M (2005) Prioritized motion planning for multiple robots. In: *IEEE International Conference on Intelligent Robots and Systems*. pp. 430–435. DOI:10.1109/IROS.2005.1545306.

van den Berg J, Snoeyink J, Lin M and Manocha D (2010) Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In: *Robotics: Science and Systems*. The MIT Press. ISBN 9780262289801, pp. 137–144.

Van den Berg JP and Overmars MH (2005) Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. *International Journal of Robotics Research* 24(12): 1055–1071.

Velagapudi P, Sycara K and Scerri P (2010) Decentralized prioritized planning in large multirobot teams. In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE, pp. 4603–4609.

Vidal E, Moll M, Palomeras N, Hernández JD, Carreras M and Kavraki LE (2019) Online multilayered motion planning with dynamic constraints for autonomous underwater vehicles. In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 8936–8942.

Wagner G and Choset H (2015a) Subdimensional expansion for multirobot path planning. *Artificial Intelligence* 219: 1–24.

Wagner G and Choset H (2015b) Subdimensional expansion for multirobot path planning. *Artificial Intelligence* 219: 1–24.

Watterson M and Kumar V (2020) Control of quadrotors using the hopf fibration on so(3). In: Amato NM, Hager G, Thomas S and Torres-Torriti M (eds.) *Robotics Research*. Cham: Springer International Publishing. ISBN 978-3-030-28619-4, pp. 199–215.

Welde* J, Rao* N, Kunapuli* P, Jayaraman D and Kumar V (2024) Leveraging symmetry to accelerate learning of trajectory tracking controllers for free-flying robotic systems. *arXiv* .

Wermelinger M, Fankhauser P, Diethelm R, Krüsi P, Siegwart R and Hutter M (2016) Navigation planning for legged robots in challenging terrain. In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE, pp. 1184–1189.

Wu B and Suh CS (2024) State-of-the-art in robot learning for multi-robot collaboration: A comprehensive survey. *arXiv preprint arXiv:2408.11822* .

Wu W, Bhattacharya S and Prorok A (2020) Multi-robot path deconfliction through prioritization by path prospects. In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 9809–9815.

Yershova A, LaValle SM and Mitchell JC (2010) Generating uniform incremental grids on so (3) using the hopf fibration. In: *Algorithmic Foundation of Robotics VIII: Selected Contributions of the Eight International Workshop on the Algorithmic Foundations of Robotics*. Springer, pp. 385–399.

Yu J and LaValle SM (2016) Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics* 32(5): 1163–1177.

Zhang X, Xiong G, Wang Y, ALI H, Teng S, Li L and Chen L (2025) Hierarchical robot coordination: A topology-guided approach for multi-robot motion planning in congested environments. *Available at SSRN 5219981* .

Zhou ZY, Liu JC and Yu JZ (2022) A survey of underwater multi-robot systems. *IEEE/CAA Journal of Automatica Sinica* 9(1): 1–18. DOI:10.1109/JAS.2021.1004269.

A Motion Planning using the Hopf Fibration

The Hopf fibration (Hopf 1931) is a prototypical example of a fibration. It formalizes a projection from the sphere $\mathbb{S}^3$ (an object embeddable in four-dimensional space) to the sphere $\mathbb{S}^2$ (embeddable in three-dimensional space). The main idea is to decompose $\mathbb{S}^3$ into subspaces (also called cosets (Judson 2023)) of identical copies of $\mathbb{S}^1$. Each copy of $\mathbb{S}^1$ is then associated to a point on $\mathbb{S}^2$ (taking the quotient of $\mathbb{S}^3/\mathbb{S}^1$ and associating equivalence classes of the quotient with $\mathbb{S}^2$).

Its relevance to motion planning comes from the double cover of $\mathrm{SO}(3)$ by the sphere $\mathbb{S}^3$. $\mathrm{SO}(3)$ is equivalent (homeomorphic) to $\mathbb{S}^3$ with antipodal points identified, i.e. $\mathrm{SO}(3) \cong \mathbb{S}^3/\{x \sim -x\}$ (Yershova et al. 2010). This can be exploited for planning purposes like efficiently sampling $\mathrm{SO}(3)$ (Yershova et al. 2010) or by constructing controllers for UAVs which are less prone to singularities (Watterson and Kumar 2020).

In our example, we use the Hopf mapping, whereby points on $\mathbb{S}^3$ are associated with quaternions, which are elements q, represented by tuples (a, b, c, d) such that $q = a + b \cdot i + c \cdot j + d \cdot k$ and $\|q\| = 1$. Those quaternions are mapped onto the $\mathbb{S}^2$ sphere as

$$f(a, b, c, d) = \begin{bmatrix} \mathrm{atan2}(x_1, x_0) \\ \arccos(x_2/r) \end{bmatrix} \tag{2}$$

whereby $x_0 = a^2 + b^2 - c^2 - d^2$, $x_1 = 2(ad + bc)$, $x_3 = 2(bd - ac)$, and $r = \sqrt{x_0^2 + x_1^2 + x_2^2}$ is the radius of the sphere. For visualization purposes, we employ a second projection, the stereographic projection of the quaternion q onto the 3-dimensional space as

$$g(a, b, c, d) = \begin{bmatrix} b/(1-a) \\ c/(1-a) \\ d/(1-a) \end{bmatrix} \tag{3}$$

.

Both those maps allow us to visualize quaternions in 3-dimensional space, either by their associated point on the sphere $\mathbb{S}^2$ or by their stereographic projection.

Finally, we require the inverse mapping from the sphere $\mathbb{S}^2$ to $\mathbb{S}^3$ to lift points from the sphere up to $\mathbb{S}^3$. This requires both a point on $\mathbb{S}^2$ as spherical coordinates and an element of $\mathbb{S}^1$ (the fiber). Let $\theta \in [-\pi, +\pi]$ and $\phi \in [0, \pi]$ be the coordinates of $\mathbb{S}^2$ (azimuth and elevation) and let $\lambda \in [0, 2\pi]$

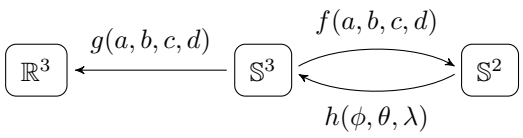


Figure 12. Hopf fibration with associated mappings. The Hopf map f from $\mathbb{S}^3$ to $\mathbb{S}^2$, the inverse hopf map h from $\mathbb{S}^2$ and fiber $\mathbb{S}^1$ to $\mathbb{S}^3$, and the stereographic projection g from $\mathbb{S}^3$ to 3-dimensional space.

be an element of $\mathbb{S}^1$. We can then lift the sphere element together with the fiber element by using

$$h(\phi,\theta,\lambda) = \begin{bmatrix} -t_1\sin(\lambda) \\ +t_1\cos(\lambda) \\ t_2\cos(\lambda) + t_3\sin(\lambda) \\ t_3\cos(\lambda) - t_2\sin(\lambda) \end{bmatrix} \tag{4}$$

whereby $t_1 = \frac{(1+x)}{n}$, $t_2 = \frac{y}{n}$, and $t_3 = \frac{z}{n}$ with $n = \sqrt{2(1+x)}$, $x = \sin(\phi)\cos(\theta)$, $y = \sin(\phi)\sin(\theta)$, and $z = \cos(\phi)$. Those mappings are summarized in Fig. 12.

B Proof of Completeness

We provide here a proof of probabilistic completeness (PC) for Fibration-RRT. The proof requires the following assumptions to hold:

1. **Single Node Probabilistic Completeness** The underlying planner used to plan paths on a single node is assumed to be PC. Any choice of tree-based (Kuffner and LaValle 2000) or roadmap-based planners (Kavraki et al. 1996) are possible. Fibration-RRT is fulfilling this property in `PlanNode`, since it is based upon RRT (Kuffner and LaValle 2000).

2. **Uniform Infinite Recurrence**. Given a fibration tree, we assume that every node in a fibration tree is chosen infinitely many times when the number of iterations of the planner goes to infinity. This is fulfilled by the `SelectNode` function for Fibration-RRT.

3. **Liftable** Every fibration has to be liftable (or partially liftable) to ensure that information can be transfered upwards (see Sec. 4.1).

4. **Admissibility** Every fibration in the fibration tree has to be admissible as detailed in Sec. 4.1. This is ensured for Fibration-RRT by only using admissible projections throughout.

Using those assumptions, we can prove the required property of Fibration-RRT.

Theorem 1. Probabilistic Completeness of Fibration-RRT. *Given Assumptions 1–4, together with a motion planning problem and an arbitrary fibration tree (as defined in Sec. 4), Fibration-RRT will find a feasible path if one exists, with probability one when the number of iterations goes to infinity.*

Proof. To prove PC for Fibration-RRT, we rely on the PC proofs given for RRT-like planners (Berenson et al. 2011; Solovey et al. 2020), which show PC by relying on a two-step method. In the first step, it is shown (or assumed) that a dense sampling sequence is given on the state space. In the second step, the series-of-balls argument is used. This involves assuming that a feasible path with clearance ϵ exists, and then covering this path with balls of size $\delta < \epsilon$. This guarantees that the δ neighborhood of the path has positive volume and is constrained-free. Using mathematical induction, it is then shown that the first ball is reached by the tree-based planner (base case), and it is then shown that ball $k+1$ is reached once ball k is reached (induction step). This relies on the fact that the dense sampling sequence will, eventually, sample a point in any positive volume ball in the state space. In fact, we can make a stronger statement: RRT-like algorithms are PC if the provided sampling sequence is dense in the *constrained-free* state space. This is enough, since the balls are assumed to be lying exclusively in the constrained-free state space.

Our argument for PC for Fibration-RRT builds upon this proof. We argue that Fibration-RRT is PC given any fibration tree, because (a) the planner on the root node is PC, and (b) since the restriction sampling is dense on the *constrained-free* state space. Let us show that both (a) and (b) are true. Property (a) is true by definition since RRT is used to plan on a single node and since by the uniform infinite recurrence (Assumption 2), this RRT is run for infinitely many iterations. It remains to show that (b) is true.

To show that restriction sampling is dense on the *constrained-free* state space, however, requires a bit of work. We formulate here a proof based upon structural induction on tree heights (Cormen et al. 2022). We divide this into two sections, whereby we first explain how structural induction extends induction, and we then use this to prove restriction sampling to be dense.

From Induction to Structural Induction

In induction, we try to prove a property P defined on the natural numbers $\mathbb{N}_{\geq 0}$ which come with their standard ordering $<$. To prove P for any $n \in \mathbb{N}_{\geq 0}$, we need to prove two cases: First we prove the *base case*, namely that $P(0)$ is true. Second, we prove the *induction step* that if $P(n)$ is true, then so is $P(n+1)$.

Structural induction generalizes this statement from the natural numbers to ordered structures like trees (Cormen et al. 2022). If we want to prove a property P on a tree T, we first define an ordering of trees by defining the ordering relation $T < T'$ if T is a subtree of T'. Given this ordering, we can prove properties of a tree T by proving the *base case*, namely that it is true for a single leaf-node tree. Once this is proven, we can prove the *induction step* namely that if the property is true for a finite set of trees $T_1, \ldots, T_N$, then it is true for T', which is the tree obtained by joining $T_1, \ldots, T_N$ under a new root node.

Denseness of Restriction Sampling

Let us prove, for any fibration tree, the property "Restriction sampling is dense on the constrained-free configuration space" (P1). To do that, we need to prove P1 on a single leaf node (base case), and we need to prove that P1 holds on a fibration tree A which is obtained as the root node of a set of fibration trees $B_1, \ldots, B_N$ on which P1 holds (induction step).

The base case is straightforward, since Fibration-RRT plans only on a single state space, thereby operating as the

underlying RRT planner (Kuffner and LaValle 2000), which is using uniform sampling, which is dense on the state space.

The main work lies in the induction step. Let us assume that P1 is true for a set of trees $B_1, \dots, B_N$, and that we want to show that P1 still holds on tree A, obtained by joining $B_1, \dots, B_N$ under a new root node. There are three cases which we have to look at.

1. **Sequential Fibration**. When $A \to B$ is sequential fibration, there exists only a single subtree by definition. As was shown in (Orthey and Toussaint 2022), restriction sampling on A is guaranteed to produce a sample in any positive volume ball on the *constrained-free* space of A. This is guaranteed since a projection of such a ball lies strictly inside the *constrained-free* space of B, since the projection onto B is admissible (see Sec. 4.1). Since the uniform infinite recurrence property holds (Assumption 2), we will, eventually, obtain a sample in any positive volume ball. See (Orthey and Toussaint 2022) for details.

2. **Partial Fibration**. Partial fibrations are similar to sequential fibrations. However, there is one important detail, namely that one has to ensure that the `SampleFiber` function is dense. For example, if the projection is onto the end-effector, we need to ensure that all possible inverse kinematics solution are sampled eventually. The remaining argument follows the argument above for sequential fibrations.

3. **Parallel Fibration**. Let $A \to A_1 \times \cdots \times A_M$ be a parallel fibration. By induction assumption, the sampling sequence on $A_1, \dots, A_M$ is dense on the constrained-free space. Let D be a positive-volume ball on the constrained free space A. Then, by admissibility (Assumption 3), the projection of D onto $A_1, \dots, A_M$ yields M balls $D_1, \dots, D_M$, which are strictly inside the respective constrained free spaces. Since the sampling sequences on those spaces are dense by induction assumption, we will, eventually, find samples in each ball $D_1 \dots, D_M$, which can be lifted into D.

We have thus proven the induction step and have shown that restriction sampling is dense on any arbitrary fibration tree (as defined in Sec. 4) which uses liftable and admissible fibrations. Thus, restriction sampling acts as a dense sampler in the constrained configuration space of the root node. Since Fibration-RRT uses RRT on the root node, this thereby shows that Fibration-RRT is PC. □