Check for updates

Article

The International Journal of Robotics Research
2024, Vol. 43(1) 3–33
© The Author(s) 2023
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/02783649231209337
journals.sagepub.com/home/ijr
Sage

Multilevel motion planning: A fiber bundle formulation

Andreas Orthey[1,2], Sohaib Akbar[3] and Marc Toussaint[1,2]

Abstract
High-dimensional motion planning problems can often be solved significantly faster by using multilevel abstractions. While there are various ways to formally capture multilevel abstractions, we formulate them in terms of fiber bundles. Fiber bundles essentially describe lower-dimensional projections of the state space using local product spaces, which allows us to concisely describe and derive novel algorithms in terms of bundle restrictions and bundle sections. Given such a structure and a corresponding admissible constraint function, we develop highly efficient and asymptotically optimal sampling-based motion planning methods for high-dimensional state spaces. Those methods exploit the structure of fiber bundles through the use of bundle primitives. Those primitives are used to create novel bundle planners, the rapidly-exploring quotient space trees (QRRT), and the quotient space roadmap planner (QMP*). Both planners are shown to be probabilistically complete and almost-surely asymptotically optimal. To evaluate our bundle planners, we compare them against classical sampling-based planners on benchmarks of four low-dimensional scenarios, and eight high-dimensional scenarios, ranging from 21 to 100 degrees of freedom, including multiple robots and nonholonomic constraints. Our findings show improvements up to two to six orders of magnitude and underline the efficiency of multilevel motion planners and the benefit of exploiting multilevel abstractions using the terminology of fiber bundles.*

Keywords
Optimal motion planning, multi-robot motion planning, nonholonomic planning, fiber bundles

Received 18 July 2020; Revised 21 July 2023; Accepted 1 October 2023
Senior Editor: Gregory Chirikjian
Associate Editor: Anirudha Majumdar

1. Introduction

As human beings, we often tackle complex problems by employing abstraction hierarchies (Simon, 1969; Ballard, 2015). Abstraction hierarchies, however, are not only helpful for us, but they can also be used by robots to solve high-dimensional motion planning problems—efficiently, and with strong guarantees (Orthey and Toussaint, 2019; Reid et al., 2019; Ichter and Pavone, 2019; Vidal et al., 2019; Gochev et al., 2012).

However, abstractions in robot motion planning are difficult to model. The state spaces involved are usually continuous, high-dimensional, and might contain intricate constraints (Konidaris, 2019). It is often unclear how to model abstractions over such state spaces, how such abstractions can be efficiently exploited, and how we can keep completeness, or optimality guarantees.

To tackle this problem, we introduce the framework of fiber bundles (Steenrod, 1951; Lee, 2003) to robot motion planning. Fiber bundles are a convenient way to model multilevel abstractions, because they provide the useful concepts of bundle sections and restrictions. Both bundle sections and restrictions allow us to develop novel

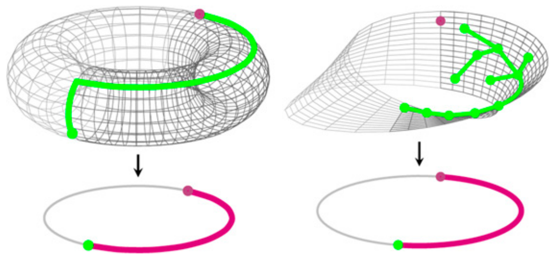
Figure 1. Efficient search over fiber bundles by exploiting path restrictions and sections. **Left**: A fiber bundle $T^2 \to S^1$ which abstracts the Torus to a base space (circle). A path on the base space (magenta) imposes a path restriction (dark gray) on the Torus, on which we can find a section (green). **Right**: A non-trivial fiber bundle from the Mobius strip to a circle, with path restriction (dark gray), and a planned tree (green).

[1]Max Planck Institute for Intelligent Systems, Stuttgart, Germany
[2]Technical University of Berlin, Stuttgart, Germany
[3]University of Stuttgart, Stuttgart, Germany

Corresponding author:
Andreas Orthey, Max Planck Institute for Intelligent SystemsHeisenbergstraße 3, Stuttgart 70569, Germany.
Email: {aorthey}@is.mpg.de

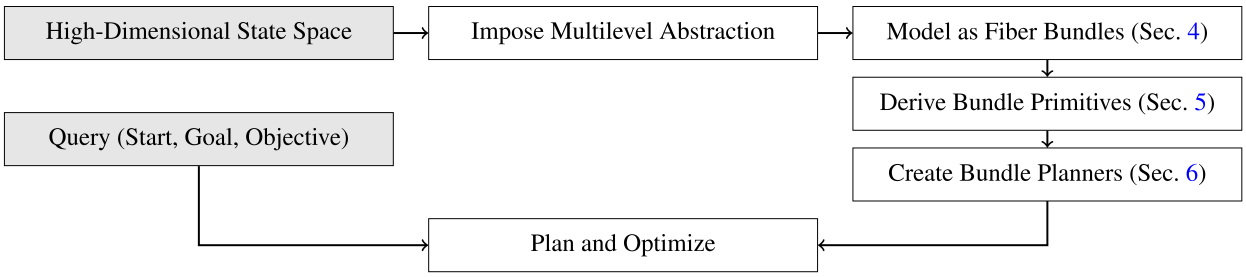
High-Dimensional State Space
Impose Multilevel Abstraction
Model as Fiber Bundles (Sec. 4)
Derive Bundle Primitives (Sec. 5)
Query (Start, Goal, Objective)
Create Bundle Planners (Sec. 6)
Plan and Optimize

Figure 2. Overview about the contributions of this paper (gray boxes are input). We tackle motion planning in high-dimensional state spaces by imposing multilevel abstractions. Those abstractions are modeled as fiber bundles (Section 4), from which bundle primitives are derived (Section 5), and the bundle planners QRRT, QRRT*, QMP, and QMP* are created (Section 6). Those bundle planners efficiently exploit bundle primitives. Given a new query in a high-dimensional state space, this allows us to plan and optimize motions, even in scenarios where non-bundle planners fail.

planning algorithms to exploit them for efficient sampling. Figure 1 illustrates these two notions, which we will introduce in more detail later. On the left, we show an abstraction (a projection) from the torus T^2 to the circle S^1. A path on the circle (magenta) imposes a path restriction (dark gray) on the torus T^2. On this path restriction, we can find path sections (green), which are paths which project onto the path on the circle. The path on the circle acts here as a guide to quickly find solution paths. On the right of Figure 1, we show the same scenario for the Mobius strip, which is a non-trivial fiber bundle (Möbius, 1858). A tree (green) is shown on the path restriction (dark gray), where planning is restricted by the path on the circle (magenta). By constructing sections and restrictions, we can create novel motion planners, which are able to quickly find relevant regions in state space to plan in. This allows those planner to efficiently solve high-dimensional motion planning problems. An overview of our approach is shown in Figure 2.

1.1. Our contributions

Our work builds on prior publications at the International Conference on Intelligent Robots and Systems (IROS) (Orthey et al., 2018) and the International Symposium on Robotics Research (ISRR) (Orthey and Toussaint, 2019). Our contributions over this prior work are:

1. We propose to formulate multilevel motion planning problems using the terminology of fiber bundles, and introduce the particularly useful notions of bundle sections and bundle restrictions.
2. Based on this formulation, we improve upon previous bundle planners QRRT and QMP and develop two new bundle planners QRRT* and QMP*.
3. We show QRRT* and QMP* to be probabilistically complete and asymptotically optimal by inheritance from RRT*, and PRM*.
4. We define primitive methods on fiber bundles and conduct a meta-analysis to find the best implementation of those methods.
5. We provide open source implementations of fiber bundles and bundle planners (together with a high-level introduction, a tutorial, and demos), which is freely available in the open motion planning library (OMPL).
6. We evaluate QMP, QMP*, QRRT, and QRRT* by comparing them against planners from OMPL on four low-dimensional scenarios ranging from 2 degrees of freedom (dof) to 7-dof, and on eight high-dimensional scenarios ranging from 21-dof to 100-dof.

Our evaluations show that our planners QMP, QMP*, QRRT, and QRRT* can efficiently exploit fiber bundles. While they are competitive in low-dimensional spaces, they are particularly useful in high-dimensional spaces, where other planners have difficulty finding solutions. We show that for high-dimensional state spaces, our bundle space planners can provide runtime improvements by up to two to six order of magnitude.

2. Related work

We provide a brief overview on motion planning with focus on optimal planning. We then discuss multilevel motion planning by discussing how our approach of fiber bundles is connected to existing research. In particular, we stress the point that fiber bundles often contribute additional vocabulary, which we can exploit to develop novel methods, simplify notations and better structure our code. We finish by reviewing complementary approaches to fiber bundles and we discuss what our approach adds to existing approaches in (optimal) multilevel motion planning.

2.1. Motion planning

To solve motion planning problems, we need to develop algorithms to find paths through the state space of a robot (Lozano-Pérez, 1983). Searching such a state space is NP-hard (Canny, 1988), but we can often efficiently find solutions using sampling-based algorithms (LaValle, 2006; Salzman, 2019), where we randomly sample states and connect them to a graph (Kavraki et al., 1996) or to a tree (Lavalle, 1998). Many variations are possible, for example, using bidirectional trees growing from start and goal state (Kuffner and LaValle, 2000; LaValle and Kuffner Jr, 2001), lazy evaluation of edges (Bohlin and Kavraki, 2000; Mandalika et al., 2019), sparse graphs (Siméon et al., 2000; Jaillet and Siméon, 2008), safety certificates (Bialkowski et al., 2016), or deterministic sampling sequences (Janson et al., 2018; Palmieri et al., 2019).

Often, we like to find an optimal path by minimizing an optimization objective (Karaman and Frazzoli, 2011). To find optimal paths, we could transfer ideas from classical planning like lazy edge evaluation (Hauser, 2015) or sparse graphs (Dobson and Bekris, 2014). However, cost function landscapes (Jaillet et al. 2010) often provide additional information we can exploit. Examples include informed sets to prune irrelevant states (Gammell et al., 2018, 2020) or fast marching trees to grow trees outward in cost to come space (Janson et al., 2015). Recently, sampling-based motion planning algorithms have also been extended to address zero-measure constraints (Kingston et al., 2019), implicit constraints (Jaillet and Porta, 2013), dynamic constraints (Li et al., 2016), or dynamic environments (Otte and Frazzoli, 2016).

All those algorithms can robustly solve many planning problems, provide formal guarantees like probabilistic completeness, or asymptotic optimality, and have been verified in a wide variety of applications (LaValle, 2006; Şucan et al., 2012). However, as we show in Section 8, we often cannot use them to solve high-dimensional planning problems in a reasonable amount of time (like less than 60 s). We believe additional information is required to solve those problems efficiently. A possible candidate for this additional information are multiple levels of abstraction.

2.2. *Multilevel motion planning*

In multilevel motion planning, we impose a multilevel abstraction on the state space and we develop algorithms which exploit this abstraction. While several models for multilevel motion planning have been put forward, we propose to use fiber bundles. To justify this decision, we show their relation to alternative modeling approaches and provide clues to the additional value they bring to the table.

2.2.1. Quotient spaces. Fiber bundles are related to quotient spaces (Orthey et al., 2018; Orthey and Toussaint, 2019; Brandao and Havoutis, 2020), latent spaces (Ichter and Pavone, 2019), or subspaces (Reid et al., 2020) in that we can represent those spaces as the base space of a fiber bundle. We can often create such a base space by taking the quotient of an equivalence class (Pappas et al., 2000). Using the ideas of base spaces, there are two interesting special cases. First, we can use base spaces to simplify a nonholonomic state space to a holonomic state space (Sekhavat et al., 1998; Vidal et al., 2019). Often, having a path on the base space is enough to find a global solution, in particular if some sort of smoothness constraint is imposed (Vidal et al., 2019; Hönig et al., 2018). Second, we can use sequences of base spaces to simplify multi-robot planning problems (Erdmann and Lozano-Perez, 1987; Siméon et al., 2002; Solovey and Halperin, 2014). We can often solve such problems efficiently by graph coordination. In graph coordination, we first plan a graph on each individual robot subspace, then we combine them using specialized algorithms like sub-dimensional expansion (Wagner and Choset, 2015) or directional oracles (Solovey et al., 2016; Shome et al., 2020). This is different from our approach, in that graphs are constructed independently, while we construct them sequentially, similar to a prioritization of robots (Erdmann and Lozano-Perez, 1987; Van Den Berg and Overmars, 2005; Ma et al., 2019).

While numerous works exist to exploit sequences of base spaces (Zhang et al., 2009; Vidal et al., 2019), we like to highlight two algorithms. First, the Manhattan-like rapidly-exploring random tree (ML-RRT) (Cortés et al., 2008; Nguyen et al., 2018), where path sections are computed similar to the L1 interpolation we advocate. However, the ML-RRT approach differs from ours, in that we use a different collision checking function for the base space and we give formal guarantees using restriction sampling. Second, the hierarchical bidirectional fast marching tree (HBFMT) algorithm (Reid et al., 2019, 2020), where restriction sampling is used on sequences of subspaces. Similar to our approach, Reid et al. (2020) prove HBFMT to be almost-surely asymptotically optimal by inheritance from BFMT*. Our approach is similar in that we develop asymptotically optimal algorithms based on RRT* and PRM*. However, contrary to both Reid et al. (2020) and Jaillet and Siméon (2008), we use quotient spaces instead of subspaces, we support manifolds instead of only Euclidean spaces, we support multiple robots with nonholonomic constraints, and we provide a variable path bias for restriction sampling (contrary to a fixed path bias Reid et al. (2020)). We also differ by providing a recursive path section method which we show to quickly find sections even in high-dimensional state spaces.

2.2.2. Constraint relaxations. Fiber bundles are related to constraint relaxations (Boyd and Vandenberghe, 2004; Roubíček, 2011), in that we can often model constraint relaxations as a particular type of fiber bundle, that is, a bundle with an admissible projection, which does not reduce the dimensionality. We can often create constraint relaxations by increasing the free space (Hsu et al., 2006), by retracting the obstacle geometry (Saha et al., 2005), or by shrinking robot links sequentially to zero (Baginski, 1996). While constraint relaxations often do not decrease the dimensionality, there are, however, extensions which do decrease the dimensionality, like progressive relaxations (Ferbach and Barraquand, 1997) or iterative constraint relaxations (Bayazit et al., 2005). In both methods, we either remove links or robots from the problem and we can use them to model the same multilevel abstractions as we can do with fiber bundles. However, by using fiber bundles, we can add additional insights like path sections and restriction sampling.

Closely related to relaxations are projections (Şucan and Kavraki 2009, 2011; Röwekämper et al., 2013; Luna et al., 2020). Projections are a component of fiber bundles, which we use to project the state space onto a lower-dimensional base space. Contrary to quotient spaces, projections are often not required to be admissible but can even be random (Şucan and Kavraki, 2009). A noteworthy approach is projection using adaptive dimensionality (Vahrenkamp et al., 2008; Gochev et al., 2012, 2013), where projections remove degrees of freedom (dof). We can remove dofs of a robot by having a fixed projection (Gochev et al., 2012; Yu et al., 2020) or by adjusting the projection depending on which links are closest to obstacles (Yoshida, 2005; Kim et al., 2015). While similar to fiber bundles, both Yoshida (2005) and Kim et al. (2015) emphasize the role of distances in workspace to choose a multilevel abstraction, which is an interesting complementary approach to ours. We differ, however, by supporting multiple robots, nonholonomic constraints, and by providing asymptotic optimality guarantees.

2.2.3. Admissible heuristics. Fiber bundles are related to admissible heuristics (Pearl, 1984; Persson and Sharf, 2014; Aine et al., 2016), in that we can use metrics on the lower-dimensional base space as admissible heuristics (Passino and Antsaklis, 1994) to guide search on the state space. This is closely related to the idea of computing lower bounds for planning problems (Salzman and Halperin, 2016). When using sequences of fiber bundles, we basically use tighter and tighter lower bounds on the real solution. Our approach differs, however, in that we do not consider inadmissible heuristics, which we could combine with admissible heuristics to often speed up planning (Aine et al., 2016; Tonneau et al., 2018).

While there are many ways to define admissible heuristics (Aine et al., 2016), we believe there are two main approaches for the case of continuous state spaces, namely, low-dimensional sampling and guide paths. In low-dimensional sampling (Şucan and Kavraki, 2009), we first sample on a lower-dimensional base space, then use those samples to restrict sampling on the state space. There are two main approaches. First, we can select sequences of subspaces of the state space (Xanthidis et al., 2018), then sample them by selecting the subspaces based on the density of samples. Second, we can use workspace sampling (Van Den Berg and Overmars, 2005; Zucker et al., 2008; Rickert et al., 2014; Luna et al., 2020), where state space samples are taken from the restriction of collision-free sets in workspace. We can do workspace sampling by focusing on narrow passages (Van Den Berg and Overmars, 2005), or by selecting promising points on the robot and guiding them through the workspace (Luna et al., 2020). Our approach is similar in that we use lower-dimensional sampling on the base space and we select base spaces based on a density criterion (Xanthidis et al., 2018). However, we differ by smoothly changing between path and graph restriction sampling, and by using a recursive path section method to efficiently find solution paths.

Closely related to low-dimensional sampling is the concept of guide paths (Tonneau et al., 2018; Ha et al., 2019). A guide path is a solution on the base space, which we use to restrict sampling on the state space (Palmieri et al., 2016). Guide paths are often used in contact planning (Bretl, 2006; Tonneau et al., 2018), where we can often give sufficiency conditions on when a feasible section exists (Grey et al., 2017). When no feasible section exists, some methods fail while other gradually shift towards graph restriction sampling (Grey et al., 2017). It is also possible to compute multiple guide paths which increase our chance to find a feasible section (Vonásek and Pěniĝka, 2019; Ha et al., 2019; Orthey et al., 2020). While we also sample along guide paths (path restriction sampling), we differ in two ways. First, we use adaptive restriction sampling to gradually change sampling from path to graph restriction, whereby we guarantee asymptotic optimality. Second, we use a recursive path section method to quickly find feasible path sections in high-dimensional state spaces.

2.3. Exploiting additional information

Fiber bundles are a way to exploit additional information. Other approaches, complementary to fiber bundles, exists. One approach is region-based decomposition. In a region-based decomposition, the problem is divided into regions in which planning becomes computationally efficient (Toussaint and Lopes, 2017; Orthey et al., 2020). Such an approach can be done in two ways. First, the workspace can be divided (Plaku et al., 2010; Vega-Brown and Roy, 2018), for example, using subdivision grids (Plaku, 2015), Delaunay decompositions (Plaku et al., 2010), or convex regions (Deits and Tedrake, 2014; Vega-Brown and Roy, 2018). Second, the solution path space can be divided (Farber, 2008), for example, by using the notion of homotopy classes (Munkres, 2000), where two paths are considered to be equivalent if we can deform them into each other. Homotopy classes are closely related to the notion of topological complexity (Farber, 2017), the minimal number of regions in state space which are collapsible into a point (null-homotopic). Several practical solutions exists to compute path homotopy classes, like the H-value (Bhattacharya et al., 2012; Bhattacharya and Ghrist, 2018), simplicial complices (Pokorny et al., 2016a), task projections (Pokorny et al., 2016b), or mutual crossings of robots (Mavrogiannis and Knepper, 2016). However, all those approaches often become computationally intractable for high-dimensional systems, multiple robots, or nonholonomic constraints. Fiber bundles are a complementary effort to organize regions on different levels of abstraction (Orthey and Toussaint, 2020).

Apart from region-based decompositions, we identify three other methods to exploit additional information. First, we can exploit distance information in workspace to compute sets of feasible states (Quinlan, 1994), which can be used to plan safe motions (Bialkowski et al., 2016), or to compute covers of free space (Lacevic et al., 2016; Lacevic and Osmankovic, 2020). Second, we can exploit differentiable constraints when available (Toussaint et al., 2018; Henkel and Toussaint, 2020). Third, we can exploit alternative state space representations, for example, by using topology-preserving mappings (Zarubin et al., 2012; Ivan et al., 2013). This is complementary to our approach, in that Zarubin et al. (2012) tries to find alternative representations of a state space, while we concentrate on finding simplifications of a given space.

2.4. Fiber bundles and prior approaches

Fiber bundle planners exploit a number of projections to accelerate planning performance. Prior approaches using projections are the KPIECE planner (Şucan and Kavraki, 2009), which uses a projection onto a simplified space, and the SBL planner (Sánchez and Latombe, 2003a), which plans using a simplified grid of the state space. However, most prior approaches are limited in the number of projections (Şucan and Kavraki, 2009; Cortés et al., 2008), the number of robots (Vidal et al., 2019), use only holonomic constraints (Zhang et al., 2009), use only Euclidean spaces (Reid et al., 2019, 2020), or work only in specific situations (Gochev et al., 2012; Kim et al., 2015). Instead, we can apply fiber bundles to any manifold space (we show it for compound spaces including the special Euclidean and orthogonal groups in 2 d and 3 d), any finite number of projections (up to 98 in our evaluations), any finite number of robots (up to eight in our evaluations) and any nonholonomic constraint (for Dubin's state spaces in our evaluations). With fiber bundles, we also provide a shared vocabulary, which can unify methods like path restriction sampling (Zhang et al., 2009; Tonneau et al., 2018; Vidal et al., 2019), or graph restriction sampling (Grey et al., 2017; Orthey et al., 2018; Reid et al., 2020). Since we also provide an open source implementation in OMPL, we can benchmark different multilevel strategies (Appendix C) and we can show the benefit of fiber bundles compared to classical motion planners (Section 8).

3. Background on optimal motion planning

Let $R_1, \ldots, R_M$ be M robots with associated (component) state spaces $Y_1, \ldots, Y_M$, respectively. We can combine the robots into one generalized robot R with associated (composite) state space $X = Y_1 \times \cdots \times Y_M$.

To each state space X, we add two complementary structures. First, we add a constraint function $\phi: X \to \{0, 1\}$ on X which takes an element x in X and returns zero if x is feasible and one otherwise. Examples of constraints are joint limits, self-collisions, environment-robot collisions and robot-robot collisions. Second, we add a steering function ψ, which takes two elements x_1 and x_2 in X as input and returns a path steering the robot from x_1 to x_2 (while potentially ignoring constraints). We denote a state space X together with the constraint function ϕ and the steering function ψ as a *planning space* (X, ϕ, ψ). The planning space implicitly defines the *free state space* as $X_\text{f} = \{x \in X | \phi(x) = 0\}$.

Given a planning space, we define a *motion planning problem* as a tuple (x_I, X_G, X). To solve a motion planning problem, we need to develop an algorithm to find a path from the initial state $x_I \in X_\text{f}$ to a desired goal region $X_G \subseteq X_\text{f}$. Often, we are not only interested in some path, but in a path which optimizes a cost functional $c: X^I \to \mathbb{R}_{\geq 0}$ whereby I is the unit interval and X^I is the set of continuous paths from I to X with finite length (Karaman and Frazzoli, 2011; Janson et al., 2018). We define the *optimal motion planning problem* as finding a path from x_I to X_G minimizing the cost functional c.

4. Multilevel motion planning

Let X be a state space and let $X_K \xrightarrow{\pi_{K-1}} \ldots \xrightarrow{\pi_1} X_1$ be a multilevel abstraction of X such that $X_K = X$, and π_k are projections from a state space X_k to a state space X_{k-1}. Each projection π_k: $X_{k-1} \to X_k$ is modeled as a fiber bundle (see Section 4.1). Given a start configuration $x_I \in X_K$, a goal region $X_G \subseteq X_K$, and an objective cost functional c, we define the *optimal multilevel motion planning problem* as the tuple $(x_I, X_G, X_1, \ldots, X_K)$ asking us to find a path from x_I to X_G while minimizing the cost c. Thus, by defining an optimal multilevel motion planning problem, we generalize optimal motion planning (Section 3) by *adding additional information*.

In the following sections, we discuss the framework of fiber bundles which provides us with the concepts of bundle restrictions and bundle sections. Those concepts will be used to define primitive methods (Section 5), which are fundamental to create bundle planners which exploit those primitive methods and plan efficiently over fiber bundles (Section 6).

4.1. Fiber bundle formulation

To model multiple levels of abstractions of state spaces, we use the framework of fiber bundles (Steenrod, 1951; Husemoller, 1966; Lee, 2003). A fiber bundle is a tuple (F, X, B, π), consisting of the total space X, the fiber space F, the base space B and the projection map

$$\pi: X \to B. \quad (1)$$

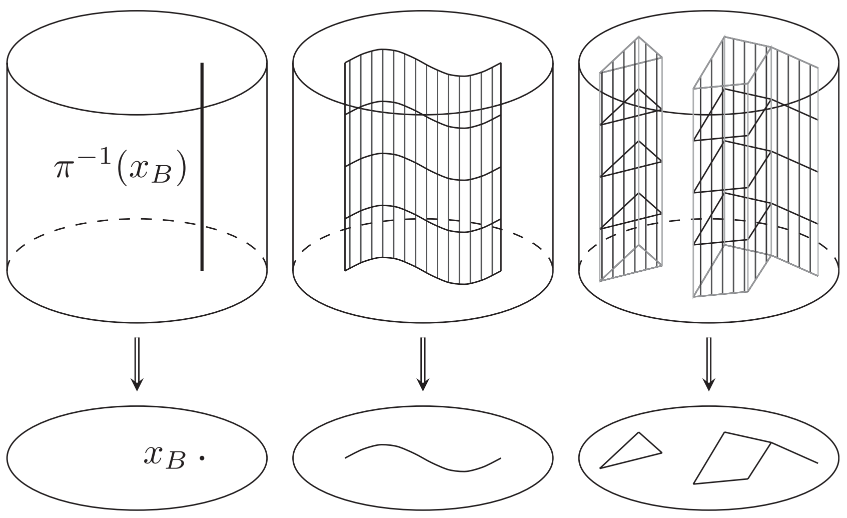

Figure 3. Bundle sections on fiber bundle $X \rightarrow B$ with base path $\{b_1, b_2, b_3, b_4, b_5\}$. We show three interpolated sections on the bundle space: L2 section (solid line), L1 fiber first section (dashed line), and L1 fiber last section (dotted line).

The mapping π needs to fulfill two properties:

1. **Union of Fibers:** The total space X is a (disjoint) union of copies of the fiber space F, parameterized by the base space B (Lee, 2003). This means that, if we take any element b in B, the preimage $\pi^{-1}(b)$ is isomorphic to the fiber space F.
2. **Local Product Space:** The total space X locally equals the product space $B \times F$. This means, if we take any element b in B, there exists a neighborhood U (an open set containing b) such that the preimage $\pi^{-1}(U)$ is homeomorphic to $U \times F$ (Lee, 2003).

In other words, a fiber bundle locally has the structure of a product space, and π provides a projection from the total space X to a "parameterization" of fibers in B. This local product structure and the projection π aligns with the terms of equivalence classes and quotient spaces as described in Appendix A. Our main motivation for leveraging the terminology of fiber bundles are the notions we introduce next.

4.2. Bundle restrictions

Given a fiber bundle (F, X, B, π), and a subset $U \subseteq B$, a bundle restriction $X|_U \subseteq X$ is the subset of the total space that projects to U. This set $X|_U = \pi^{-1}(U)$ is called the restriction of X to U (Tu, 2017).

We consider three kinds of restrictions. First, given a point x_B on the base space, we use its restriction $F|_{x_B} = \pi^{-1}(\{x_B\})$. Note that we call $F|_{x_B}$ a fiber as it is, by definition, isomorphic to F (Assertion 1.). We visualize this in Figure 3 (Left). Second, given a path $\mathbf{p}_B: I \rightarrow B$ on the base space, with $I = [0, 1]$ the unit interval, we have its restriction $X|_{p_B} = \pi^{-1}(\{\mathbf{p}_B(t) : t \in I\})$ (Figure 3, Middle). And third, given a graph $\mathbf{G}_B$ on the base space, we have the graph restriction $\pi^{-1}(\mathbf{G}_B) \subseteq X$, where we unproject the union of all its vertices and edges (Figure 3, Right).

We use these three restrictions for different computations. First, we use point restrictions (fibers) to lift base space elements up to the total space (Section 4.3). Second, we use path restrictions to quickly compute sections, which are paths on the total space constraint to the path restriction (Section 4.3 and Section 5.4). Third, we use graph restrictions to formulate restriction sampling, that is, sampling restricted to elements of the total space that project onto the base space graph (Section 5.1). It is important to note that restriction sampling is dense in the free total space, if the graph on B is dense. We use this denseness property to prove probabilistic completeness and asymptotic optimality (Section 7).

4.3. Bundle sections

Given a fiber bundle (F, X, B, π) and a subset $U \subseteq B$ of the base space, a section of U is a map $s: U \rightarrow X$ such that $\pi(s(u)) = u$ (Lee, 2003). In other words, while a restriction $X|_U$ unprojects U to *all* elements $x \in X$ that project to U, a section maps each $u \in U$ to just *one* specific element $x \in X$ that projects to u (It also follows, that $s(u) \in X|_U$ for any section s.). We define useful cases of sections in the following.

4.3.1. Lift. When U contains only a single element $\{b\}$, we call the section a *lift*. A lift $s(b)$ takes as input an element b in B and returns an element x on the total space. We often like to single out a specific element x by additionally choosing a fiber space element f, whereby we overload the lift as $s(b, f)$. In the case of X being a product space, we define the lift as $s(b, f) = (b, f)$. However, if $X \rightarrow B$ is not trivial, the base space element b defines an isomorphic transformation of the fiber space (including the fiber element f), which in turn uniquely defines the element x. In this work, all bundle spaces are trivial except the Mobius strip. For the Mobius strip, we define the transformation as a linear transformation, involving a translation of the fiber space (here: the unit [0, 1] interval) around the circle while simultaneously rotating the fiber space.

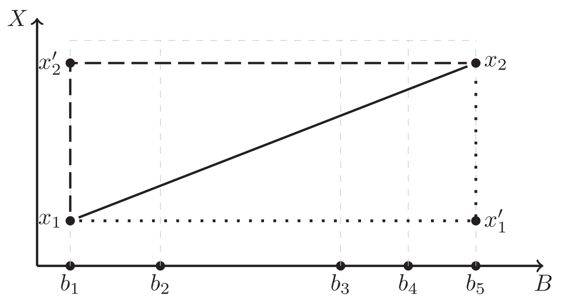

Figure 4. Bundle sections on fiber bundle X → B with base path fb1; b2; b3; b4; b5g. We show three interpolated sections on the bundle space: L2 section (solid line), L1 fiber first section (dashed line) and L1 fiber last section (dotted line).

4.3.2. Path section. When the subset U is an interval, we call the section a *path section*. A path section of a path $\mathbf{p}_B$: $I \to B$ is itself a path $\mathbf{p}$: $I \to X$ such that $\mathbf{p}_B = \pi \circ \mathbf{p}$.

Our algorithms will aim to find feasible path section, that is, feasible unprojections of paths in the base space to paths in the full space. We use three interpolation methods to this end. All three methods take as input a base path $\mathbf{p}_B$ and two total space elements x_1 and x_2 in X. Let π_F be the projection of the total space onto the the fiber space (i.e., orthogonal to the base space projection π). We then compute fiber space elements $f_1 = \pi_F(x_1)$ and $f_2 = \pi_F(x_2)$ that introduce coordinates along the fibers, and which we use to interpolate. Each method differs by how we interpolate between f_1 and f_2 along the path restriction $\pi^{-1}(\mathbf{p}_B)$ (see also Figure 4).

4.3.3. L2 section. To interpolate a section, we can use a straightforward L^2 section. To interpolate an L^2 section, we use the shortest path under the L^2-norm, which is simply the linear interpolation

$$l_{L^2}(t) = (1-t)f_1 + t(f_2 - f_1). \tag{2}$$

We then compute the section as

$$\boldsymbol{p}(t) = s(\boldsymbol{p}_B(t), l_{L^2}(t)) \tag{3}$$

by lifting each path base element to the bundle space. We use the L^2 section mainly to compute quotient space metrics (Section 5.2).

4.3.4. L1 section. An alternative to L^2 sections are L^1 sections. In an interpolation with an L^1 section, we compute the section as

$$\boldsymbol{p}(t) = s(\boldsymbol{p}_B(t), l_{L^1}(t)) \tag{4}$$

with the interpolation

$$l_{L^1} = \begin{cases} f_1, & \text{if } t < \frac{1}{2} \\ f_2, & \text{if } t \geq \frac{1}{2} \end{cases} \tag{5}$$

We use two flavors of L^1 sections. The first flavor is fiber first (FF) sections, where we use the adjusted base path as

$$\mathbf{p}_{\mathrm{FF}}(t) = \begin{cases} \mathbf{p}_B(0), & \text{if } t < \frac{1}{2} \\ \mathbf{p}_B\left(2\left(t - \frac{1}{2}\right)\right), & \text{if } t \geq \frac{1}{2} \end{cases} \tag{6}$$

The second flavor is L^1 fiber last (FL) sections, where we use the base path as

$$\mathbf{p}_{\mathrm{FL}}(t) = \begin{cases} \mathbf{p}_B(2t), & \text{if } t < \frac{1}{2} \\ \mathbf{p}_B(1), & \text{if } t \geq \frac{1}{2} \end{cases} \tag{7}$$

Both fiber first and fiber last L^1 sections are cornerstones of our find section method, which we will use alternately to find feasible sections (see Section 5.4 for details).

4.4. Bundle sequences

With a fiber bundle, we model a single state space abstraction. To model multiple levels of abstraction, we can chain fiber bundles into sequences. A fiber bundle sequence is a tuple $(X_{1:K}, F_{1:K-1}, \pi_{1:K-1})$ such that the k-th base space is equal to the $k-1$-th total space. We write such a sequence as

$$X_K \xrightarrow{\pi_{K-1}} X_{K-1} \xrightarrow{\pi_{K-2}} \ldots \xrightarrow{\pi_1} X_1 \tag{8}$$

whereby we call the space X_k the k-th bundle space.

4.5. Admissible fiber bundles

An important type of fiber bundles are the ones using admissible projections (Orthey et al., 2018). An admissible projection is a projection preserving the feasibility of a state. This is important to preserve probabilistic completeness and asymptotic optimality. We next define admissible projections and discuss the corresponding notion of admissible fiber bundles.

Let ϕ and ϕ_B be constraint functions on X and B, respectively. Given the constraint functions, we can define the free total space X_{f} and the free base space B_{f} (see Section 3). For an admissible projection, we require the projection mapping to fulfill the first two requirements (Assertions 1 and 2 above) plus the following third requirement:

3. **Admissible**: The projection mapping does not invalidate solutions. This means, if we map the free state space X_{f} via π onto the base space, then the image $\pi(X_{\mathrm{f}})$ is a subset of the free base space B_{f}. Or, equivalently,

$\phi_B(\pi(x)) \leq \phi(x)$ for any $x \in X$ (Orthey and Toussaint 2019).

If a projection mapping is admissible w.r.t. given ϕ and ϕ_B, we call the fiber bundle an admissible fiber bundle. Analogously, if a sequence of fiber bundles contains only admissible projections, we call it an admissible fiber bundle sequence. It is important to note that admissibility is a requirement, if we like to prove probabilistic completeness or asymptotic optimality.

Using admissible fiber bundle (sequences), we thus can tie together the notions of quotient spaces, constraint relaxations and admissible heuristics. First, we can interpret fiber bundles as a generalization of constraint relaxations (Orthey and Toussaint, 2019), where paths on the base space are lower bound estimates on solution paths on the total space. Second, we can use a solution on the base space as an admissible heuristic (Aine et al., 2016) and exploit it by using either restriction sampling, by using a quotient space (base space) metric (Passino and Antsaklis, 1994; Pearl, 1984), or by computing sections along a given base space path (Zhang et al., 2009).

4.6. Examples of fiber bundles

To make the discussion more concrete, we discuss two (multilevel) abstractions which are often used in motion planning.

4.6.1. Prioritized multi-robot motion planning. To plan motions for multiple robots, we can prioritize the robots (Erdmann and Lozano-Perez, 1987; Ma et al., 2019). Given M robots, we can order them, then plan for the first robot and use its motion as a constraint on the motion of the next robot. We can formalize this as a fiber bundle sequence

$$Y_{1:M} \xrightarrow{\pi_{M-1}} Y_{1:M-1} \xrightarrow{\pi_{M-2}} \cdots \xrightarrow{\pi_1} Y_1 \tag{9}$$

whereby Y_m is the state space of the m-th robot and $Y_{1:m}$ is the Cartesian product of the state spaces $Y_1, \ldots, Y_m$. In the fiber bundle sequence, we remove, in each step, the configuration space and the geometry of the least important robot. We can then either plan a path in Y_1 and use it as a constraint for the next robot (i.e., finding a feasible section in the path restriction). This is known as path coordination (Siméon et al., 2002). Or we use the graph on Y_1 to restrict sampling for the remaining robots, which is known as graph coordination (Svestka and Overmars, 1998). In practice, we can realize graph coordination either by using an oracle to guide expansion (Solovey et al., 2016) or by expanding dimensionality when conflicts arise (Wagner and Choset, 2015).

4.6.2. Tangent bundle and path-velocity decomposition. When planning for a dynamical system, we often can simplify planning using a tangent bundle decomposition. Given a state space X we impose a tangent bundle $X = TM = M \times \mathbb{R}^n$ with projection

$$M \times \mathbb{R}^n \rightarrow \pi M \tag{10}$$

whereby $n = \dim M$, $\mathbb{R}^n$ is the fiber space and M is the base space. We call M the configuration space and TM the tangent bundle. Planning on tangent bundles often follows a two-step approach. First, we compute a path p_M on the configuration space M (the base space) avoiding obstacles and self-collisions. Second, we compute a velocity along the path, that is, a time reparameterization. Such a time reparameterization is a path section of p_M and we can find such a section by solving a convex optimization problem (Bobrow et al., 1985), which we can solve efficiently (Pham and Pham, 2018). To guarantee completeness, however, we need to either plan on the full tangent bundle TM (LaValle and Kuffner Jr, 2001) or track valid speed profiles along paths on M (Pham et al., 2017).

5. Primitive methods on fiber bundles

Algorithm 1 RESTRICTIONSAMPLING(X_k)

```
1: if EXISTS($X_{k-1}$) then
2:     $x_{\text{base}} \leftarrow$ SAMPLEBASE($\mathbf{G}_{k-1}$)    ▷ Element of $X_{k-1}$
3:     $x_{\text{fiber}} \leftarrow$ SAMPLE($x_{\text{base}}, F_k$)    ▷ Element of $F_k$
4:     $x_{\text{rand}} \leftarrow$ LIFT($x_{\text{base}}, x_{\text{fiber}}$)    ▷ Element of $X_k$
5: else
6:     $x_{\text{rand}} \leftarrow$ SAMPLE($X_k$)
7: end if
8: return $x_{\text{rand}}$
```

To exploit fiber bundles, we derive a set of primitive methods. This includes methods to sample the base space, to compute a metric, to select a bundle space to grow next, and to rapidly find a feasible section. To implement each method, we develop several different strategies and discuss how they influence the algorithms. To select the best strategies for each algorithm, we perform a meta-analysis in Appendix C.

To use the primitive methods, we assume that every bundle space X_k has access to the following fiber bundle specific functions:

1. A fiber space $F_k = X_k/X_{k-1}$
2. A base space projection π_k: $X_k \rightarrow X_{k-1}$
3. A fiber space projection $\pi_k^F : X_k \rightarrow F_k$
4. Projected start state x_I^k and goal region X_G^k
5. A graph $\mathbf{G}_k = (V_k, E_k)$ containing $|V_k|$ vertices and $|E_k|$ edges

The primitives will be used in the development of the bundle planners (Section 6), where we exploit primitives for improved planning performance.

5.1. Restriction sampling

In restriction sampling, we sample states on the total space X_k by sampling exclusively in the graph restriction induced by the graph on the base space X_{k-1} (see Section 4.2), as we detail in Alg. 1. We first check if the base space X_{k-1} exists (Line 1.1). If it does not exists, we revert to a standard sampling method like uniform sampling (Line 1.6). If it does exists, we first sample a base space element (Line 1.2), then use it to sample a fiber space element (Line 1.3) and finally lift the base space element to the bundle space using the fiber space element (Line 1.4). The lift operation depends on if the bundle is trivial, in which case we just concatenate base element and fiber element. If the bundle is non-trivial (like the Mobius strip), we use the base element to index the correct fiber space, then use the fiber element to index the correct bundle space element (see Section 4.3).

To implement the SAMPLE function, we use uniform sampling of the space. However, other sampling techniques are certainly possible, like Gaussian sampling (Boor et al., 1999), obstacle-based sampling (Amato et al., 1998), bridge sampling (Hsu et al., 2003), maximum clearance (Wilmarth et al., 1999), quasi-random (Branicky et al., 2001), utility-based (Burns and Brock, 2005), or deterministic sampling (Janson et al., 2018; Palmieri et al., 2019). To guarantee probabilistic completeness and asymptotic optimality, we only need to verify that those sequences are dense.

The main method of restriction sampling is the SAMPLEBASE method. In the SAMPLEBASE method, we sample the graph $\mathbf{G}_{k-1}$ on the base space. While numerous methods exist to sample a graph (Leskovec and Faloutsos, 2006), we found five methods particularly important.

5.1.1. Random vertex sampling. First, we can choose a vertex at random, which we refer to as Random Vertex (RV) sampling (Leskovec and Faloutsos, 2006). In RV sampling, we choose a random integer between 1 and $|V|$ which uniquely defines a vertex on the graph $\mathbf{G}$. This sampling is particularly fast ($O(1)$ operations), but might be overly constrictive if we sample from a tree or a graph with long edges. However, for large graphs, this sampling procedure is often the only alternative to not slow down sampling.

5.1.2. Random edge sampling. Second, we can choose an edge at random, then pick a state on this edge, a method we refer to as Random Edge (RE) sampling (Leskovec and Faloutsos, 2006). This method requires two operations, first to pick an edge, then to pick a number between 0 and 1 to determine the state on the edge. This method seems to be superior if the graph is sparse and has long edges, in particular edges going through narrow passages.

5.1.3. Random degree vertex sampling. Third, we can choose a vertex at random, but biased towards vertices with a low degree (number of outgoing edges). We refer to this as Random Degree-Vertex (RDV) sampling. With RDV sampling, we bias samples to vertices which are either in tight corners or inside of narrow passages. Vertices in large open passages often have many neighbors and thereby a large degree. This method, however, requires to update a probability function which tracks the degrees of each vertex.

5.1.4. Path restriction sampling. Fourth, we can choose a sample on the lowest cost path on the graph, a method we refer to as *path restriction* (PR) sampling. We can utilize PR sampling in two ways. Either, we sample on the path restriction with a fixed probability β_{fixed}. This is similar to the fixed tunnel radius proposed by Reid et al. (2019). Or, we first sample exclusively on the path restriction, then gradually decay towards the fixed path bias. We call this method PR decay sampling.

PR decay sampling allows us to model a change in belief. It is often true that the shortest path on the base space contains a feasible section, which we should search for by exclusively sampling on the path restriction (Orthey et al., 2018). If we do not find a valid section, we should gradually dismiss our belief that a section exists and try to sample the graph restriction instead. To model this change in belief, we use an exponential decay function to smoothly transition from probability 1 down to the fixed probability β_{fixed} using a decay constant λ. See Appendix B for the definition of exponential decay.

Before using PR decay sampling, we simplify the path. Simplifying the path is similar to the local path refinement method (Zhang et al., 2009), where a path is optimized to increase its clearance. For this operation, we use a simple short-cutting path optimizer, which does not slow down planning in high-dimensional spaces.

We use a path optimizer with a cost term for path length.

5.1.5. Neighborhood sampling. Fifth, we can choose a sample not directly on the graph, but in an epsilon neighborhood. We refer to this as *neighborhood* (NBH) sampling. NBH sampling is helpful when there is a path through a narrow passage which comes close to its boundary. Those paths often do not have a feasible section. Instead, if we would perturbate the path slightly, we can often find a path admitting a feasible section. With NBH sampling, we first sample a configuration x exactly on the graph, and then sample a second configuration x' which we sample uniformly in an epsilon ball around x. In practice, we use an exponential decay (Appendix B) to smoothly vary the size of the neighborhood from zero up to epsilon. With NBH sampling, we can often solve problems where a solution through a narrow passage has few or no samples, while using nearby samples allows us a bit more wriggle room. Note that instead of uniform epsilon sampling, we could also use a Gaussian distribution with mean x and epsilon variance (Reid et al., 2019). However, in preliminary testing, we could not observe a difference between them.

5.2. Bundle space metric

An essential component of bundle algorithms are the nearest neighbor computations, which depend on choosing a good metric function. We discuss two possible metrics, the intrinsic bundle metric (ignoring the base space) and the quotient space metric (exploiting the base space).

5.2.1. Intrinsic bundle metric. To straightforwardly attach a metric to the bundle space, we use the geodesic distance between two points while ignoring the base space. We compute this intrinsic metric on X as

$$d(x_1, x_2) = d_X(x_1, x_2) \tag{11}$$

While this is a naive way to compute the metric, we note that using base space information is often costly, and the total space metric is often good enough (Orthey and Toussaint, 2019).

5.2.2. Quotient space metric. If a base space is available, we can consider it as a quotient space, on which we can define a quotient space metric (Guo et al., 2019). To define a quotient space metric between two states, we first project both states onto the base space, compute a shortest path p_B using the base graph and then interpolate an L^2 section along the path restriction $X|_{p_B}$ (see Section 4.3).

In particular, given two points x_1 and x_2 in X_k, we project them onto the base space X_{k-1} to yield $b_1 = \pi(x_1)$ and $b_2 = \pi(x_2)$. We then compute the nearest vertices v_1 and v_2 on the graph $\mathbf{G}_{k-1}$ and we compute a path on $\mathbf{G}_{k-1}$ between v_1 and v_2 using the A* algorithm with the intrinsic base space metric as an admissible heuristic. Finally, we use the fiber space projection of x_1 and x_2 to compute fiber space elements $f_1 = \pi_F(x_1)$ and $f_2 = \pi_F(x_2)$, which we use to integrate an L^2 section (Section 4.3). We then compute the bundle space metric as

$$d(x_1, x_2) = \begin{cases} d_X(x_1, x_2) & v_1 = v_2 \\ d_F(f_1, f_2) & \\ +d_B(y_1, v_1) & \\ +d_B(y_2, v_2) & \text{otherwise} \\ +d_{\mathbf{G}_k}(v_1, v_2) & \end{cases} \tag{12}$$

with d_F being the fiber space metric (L^2), d_B the base space metric and $d_{\mathbf{G}_k}$ the length of the shortest path on $\mathbf{G}_k$ between vertices under the base space metric.

While the quotient space metric is more mathematically sound, there are two practical problems. First, computing this metric is costly, because we need to perform a graph search operation. Second, the graph on the base space might not yet be dense, thereby potentially returning values leading to an inadmissible heuristic, which in turn would mislead the planner on the bundle space.

5.3. Bundle space importance

In each iteration of multilevel motion planning, we make a choice about expanding a graph by selecting a level. To select a level, we attach an importance function to each bundle space, which we use to rank the bundle spaces. We develop three different importance strategies.

5.3.1. Uniform. In uniform importance, we select all bundle spaces an equal amount of times. This is similar to round-robin change, similar to scheduling operations (Russell and Norvig, 2002). Here we use a slight variation, where we compute the importance based on the number of vertices, thereby ensuring a uniform expansion of each level. In particular, for bundle space X_k with graph $\mathbf{G}_k$ and $|V_k|$ vertices, we compute its importance as $1/|V_k| + 1$.

5.3.2. Exponential. To densely cover spaces with higher dimensions, we usually require more samples. In general, the sampling density is proportional to $N^{1/d}$ where N is the number of samples and d the dimensionality (Hastie et al., 2009). Therefore, we should select the space with the lowest density first, thereby guaranteeing equal sampling density across all spaces. We can compute an exponential importance as $1/|V_k|^{1/d} + 1$ which reflects an exponential increase of samples in higher dimensions. This idea is similar to the selection of bundle spaces using a geometric progression (Xanthidis et al., 2018). This is also related to multilevel monte carlo (Giles, 2015) and sparse grid methods (Bungartz and Griebel, 2004).

5.3.3. Epsilon greedy. Whenever we find a graph connecting initial and goal state on the base space, it seems reasonable to greedily exploit this graph to find a path on the bundle space. This strategy is not complete, since the graph might not yet contain a feasible section (see Section 4.2). We can, however, create a complete algorithm by extending the base space with an epsilon probability while extending the bundle space the rest of the time. We compute this as

$$f(k) = \begin{cases} \epsilon^{K-k} - \epsilon^{K-k+1} & k > 1 \\ \epsilon^{K-1} & \text{otherwise} \end{cases} \tag{13}$$

whereby k is the bundle space level and K is the total number of bundle spaces. We then compute the importance for the k-th bundle space as $1/|V_k|/f(k) + 1$, reflecting our desire to expand recent levels more aggressively.

5.4. Finding path sections

Finding path sections quickly and reliably is one of the cornerstones of all bundle planners. In this section, we use the interpolation methods of Section 4.3 to develop a recursive path section algorithm, which we depict in Alg. 2. For this to work, we need to have at least a base space (Line 2.1). We then compute the shortest path on the base space (Line 2.2) and recursively compute a section, either by starting from an L1 fiber first section (Line 2.3) or if unsuccessful, by starting from an L1 fiber last section (Line 2.5).

To recursively compute a section, we show the pseudocode in Alg. 3. We terminate the algorithm if we reach a certain depth d_{MAX} (Line 3.3) or if we reach the goal region (Line 3.8). Inside each recursion iteration, we interpolate an L1 section, either fiber first (if FF is true) or fiber last (if FF is false) (Line 3.6). We then propagate the system along the section while valid (Line 3.7) and return the last valid state.

If we do not reach the goal state with the last valid state, we do up to b_{MAX} sidesteps along the fiber space. Sidestepping means that we project the last valid state onto the base space (Line 3.11), then sample a random fiber space element (Line 3.13) and lift the states to the bundle space to obtain a state x_k (Line 3.14). We then check if we can move from the last valid state to the state x_k (Line 3.15). Since both states have the same base space projection, we call this a *sidestep* (i.e., a step orthogonal to the base space). If the motion is valid, we clip the remaining base path (Line 3.17) and recursively call the algorithm (Line 3.18). In the recursion call, we increase the depth, use the clipped base path segment and change the interpolation method from fiber first to fiber last. We change the interpolation at this point, because we observe an alternation between interpolation methods to substantially improve runtime.

5.4.1. Nonholonomic constraints. In the case of holonomic constraints, we can use the L^1 interpolation (Line 3.6) and the base space segment (Line 3.17) to follow the path restriction exactly. However, if we have nonholonomic constraints, we often cannot follow the path restriction exactly, in particular if the base space path is piece-wise linear. Note that a base space path is often piece-wise linear if we do not impose additional smoothness assumptions (Vidal et al., 2019; Hönig et al., 2018).

To still compute path sections over piece-wise linear base space paths in the nonholonomic case, we do a two-phase approach. First, we compute the interpolation values as in Section 4.3, but only at discrete points, which provides us with a set of points on the bundle space. Second, we interpolate between those points by using the nonholonomic steering function. While we might deviate from the base path restriction, we follow, however, the base path restriction as close as the steering function allows us. This approach is similar to the idea of interpolating waypoints with dynamically feasible path segments, which has been done for flying quadrotors (Richter et al., 2016) and for underwater vehicles (Yu et al., 2019). However, we differ by first interpolating values for the fiber spaces along the base space path. The remaining computation in Alg. 3 remains exactly as in the holonomic case.

6. Bundle space motion planners

Algorithm 2 FINDSECTION(X_k)

```
1: if EXISTS($X_{k-1}$) then
2:     $p \leftarrow$ SHORTESTPATH($x_I^{k-1}, X_G^{k-1}, \mathbf{G}_{k-1}$)
3:     FINDSECTION($p, x_I^k, X_G^k, 0,$ true)
4:     if $\neg$PTC($X_k$) then
5:         FINDSECTION($p, x_I^k, X_G^k, 0,$ false)
6:     end if
7: end if
```

To solve a multilevel motion planning problem, we develop a set of algorithms generalizing existing motion planners to fiber bundles. All those planners share the same high-level structure, which we call a BUNDLEPLANNER (Alg. 4). In the BUNDLEPLANNER method, we first initialize a priority queue sorted by the importance of each bundle space (Line 4.1). We then iterate over all bundle spaces, try to find a section on the k-th bundle space (Line 4.3) and then push the k-th bundle space into the priority queue (Line 4.4). We then execute the while loop while a planner terminate condition (PTC) is not fulfilled for the k-th bundle space (Line 4.5). Inside the loop, we select the most important bundle space, grow the graph or tree and push the space back into the queue (Line 4.6 to 4.8). We terminate if the PTC for the K-th bundle space has been fulfilled. This means we either terminate successfully, found the problem to be infeasible or reach a timelimit.

All bundle space algorithms are alike in sharing the same high-level structure; each bundle space algorithm differs in their GROW function (Line 4.7) and their primitive methods (Section 5).

6.1. Bundle planner variants

The BUNDLEPLANNER algorithm is used to develop novel algorithms by changing the GROW function. To implement the GROW function, we can utilize almost any single-level planning algorithm. In our case, we use the algorithms RRT, RRT*, PRM, and PRM* (please consult Table 1 for abbreviations of algorithms).

All grow functions in a multilevel versions of our algorithms differ from their single-level version in four points. First, we replace uniform sampling by restriction sampling, as we detail in Section 5.1. Algorithms might differ in how we implement graph sampling in restriction sampling. Second, when pushing a new bundle space into the priority queue, we check for a feasible section over the solution path on the last bundle space, as we detail in Section 5.4. This computation is equivalent for each bundle planner. Third, we rank bundle spaces based on a selection criterion, which we detail in Section 5.3. Algorithms might differ in the type of selection criterion we employ. Fourth, we adjust the metric on the bundle space, which affects both nearest neighbors computation and the steering method, as we detail in Section 5.2. While different metrics are possible

(Orthey et al., 2018), we use the intrinsic bundle metric for all algorithms (as determined by our meta-analysis in Appendix C).

Algorithm 3 FINDSECTION(p, x_a, X_G^k, d, FF)

```
1:  Let $b_{\text{MAX}}$ be the maximal branching factor
2:  Let $d_{\text{MAX}}$ be the maximal depth
3:  if $d \geq d_{\text{MAX}}$ then
4:      return false
5:  end if
6:  $s \leftarrow$ INTERPOLATEL1$(p, x_a, X_G^k, FF)$      ▷ see Sec. 4.3
7:  $x \leftarrow$ PROPAGATEWHILEVALID$(s)$      ▷ Return last valid
8:  if $x$ is in $X_G^k$ then
9:      return true
10: end if
11: $x_{\text{base}} \leftarrow$ PROJECTBASE$(x)$
12: for $j \leftarrow 1$ to $b_{\text{MAX}}$ do
13:     $x_{\text{fiber}} \leftarrow$ SAMPLE$(x_{\text{base}}, F_k)$      ▷ Sidestep on fiber
14:     $x_j \leftarrow$ LIFT$(x_{\text{base}}, x_{\text{fiber}})$
15:     if CHECKMOTION$(x, x_j)$ then
16:         $\mathbf{G}_k \leftarrow \mathbf{G}_k \cup \{x, x_j\}$
17:         $p_j \leftarrow$ GETSEGMENT$(p, x_{\text{base}})$
18:         return FINDSECTION$(p_j, x_j, X_G^k, d+1, \overline{FF})$
19:     end if
20: end for
```

Algorithm 4 BUNDLEPLANNER$(x_I, x_G, X_1, \cdots, X_K)$

```
1:  Let $\mathbf{X}$ be a PRIORITY QUEUE
2:  for $k = 1$ to $K$ do
3:      FINDSECTION$(X_k)$      ▷ Sec. 5.4
4:      $\mathbf{X}$.PUSH$(X_k)$
5:      while $\neg$PTC$(X_k)$ do
6:          $X_{\text{select}} = \mathbf{X}$.POP      ▷ Sec. 5.3
7:          GROW$(X_{\text{select}})$      ▷ Sec. 6.1
8:          $\mathbf{X}$.PUSH$(X_{\text{select}})$
9:      end while
10: end for
```

Algorithm 5 GROWQRRT(X_k)

```
1: $x_{\text{rand}} \leftarrow$ RESTRICTIONSAMPLING$(X_k)$
2: $x_{\text{near}} \leftarrow$ NEAREST$(x_{\text{rand}}, \mathbf{G}_k)$
3: $x_{\text{new}} \leftarrow$ STEER$(x_{\text{near}}, x_{\text{rand}}, \mathbf{G}_k)$
4: if $\neg$CHECKMOTION$(x_{\text{near}}, x_{\text{rand}})$ then
5:     return
6: end if
7: $\mathbf{G}_k = \mathbf{G}_k \cup \{x_{\text{near}}, x_{\text{new}}\}$
```

Algorithm 6 GROWQRRT*(X_k)

```
1: GROWQRRT$(X_k)$
2: $x_{1:K} \leftarrow$ K-NEARESTNEIGHBORS$(x_{\text{new}}, \mathbf{G}_k, \mathbf{G}_{k-1})$
3: for $x_{\text{nbh}} \in x_{1:K}$ do
4:     REWIRE$(x_{\text{nbh}}, x_{\text{new}})$
5: end for
6: for $x_{\text{nbh}} \in x_{1:K}$ do
7:     REWIRE$(x_{\text{new}}, x_{\text{nbh}})$
8: end for
```

Algorithm 7 REWIRE(x, y)

```
1: if CHECKMOTION(x,y) then
2:     $c \leftarrow$ COST$(x, y)$
3:     if COST$(x) + c <$ COST$(y)$ then
4:         UPDATETREE$(y, x, \mathbf{G}_k)$
5:     end if
6: end if
```

Algorithm 8 GROWQMP(X_k)

```
1: $x_{\text{rand}} \leftarrow$ RESTRICTIONSAMPLING$(X_k)$
2: if $\neg$ISVALID$(x_{\text{rand}})$ then return
3: end if
4: $\mathbf{G}_k = \mathbf{G}_k \cup \{x_{\text{rand}}\}$
5: $x_{1:K} \leftarrow$ K-NEARESTNEIGHBORS$(x_{\text{rand}}, \mathbf{G}_k, \mathbf{G}_{k-1})$
6: for $x_{\text{nbh}} \in x_{1:K}$ do
7:     $x_{\text{new}} \leftarrow$ STEER$(x_{\text{nbh}}, x_{\text{rand}}, \mathbf{G}_k)$
8:     $\mathbf{G}_k = \mathbf{G}_k \cup \{x_{\text{nbh}}, x_{\text{new}}\}$
9: end for
```

6.2. QRRT

In Alg. 5, we show the QRRT algorithm. We previously introduced QRRT in Orthey and Toussaint (2019). We differ here by using an exponential importance primitive (Section 5.3.2) and by adding the find section primitive (Section 5.4). The remaining structure, however, remains unchanged. In detail, we sample a random element from the bundle space (Line 5.1) using restriction sampling (Section 5.1). We then choose the nearest vertex from the tree (Line 5.2) and steer from the nearest to the random element (Line 5.3). We then check if the motion is collision-free and add the new state to the tree. Note that we stop steering if the distance goes above a threshold, similar to RRT (LaValle and Kuffner Jr, 2001).

*6.3. QRRT**

While QRRT performs well in our evaluations, we can improve upon QRRT by developing an asymptotic optimal version. We call this QRRT* and depict the algorithm in Alg. 6. By developing QRRT*, we generalize RRT* (Karaman and Frazzoli, 2011) to multiple levels of abstraction. To implement QRRT*, we use one iteration of QRRT (Line 6.1), then compute k nearest neighbors of the new state (Line 6.2). We choose the k as $k = k_{\text{RRT}} \log(N)$ whereby N is the number of vertices in the tree (Karaman and Frazzoli, 2011). The parameter k_{RRT} can be chosen based on the dimension of the problem (Karaman and Frazzoli, 2011; Kleinbort et al., 2019).

After computing k nearest neighbors, we perform two rewire operations (this dicussion follows closely Salzman and Halperin (2016)). First, we rewire the nearest neighbors to the new state (Line 5.4). Second, we rewire

Table 1. List of Motion planning Algorithms Used in Experimental Section. Properties of the Algorithms are: Supporting Fiber Bundles (FB), Being Probabilistically Complete (PC), and Being Asymptotically (Near-)Optimal (AnO).

Motion planner	Description	Origin paper	FB	PC	AnO
QRRT	Rapidly-exploring random quotient space trees	(Orthey and Toussaint 2019)	x	x	
QMP	Quotient space roadmap planner	(Orthey et al. 2018)	x	x	
QRRT*	Optimal version of QRRT	**This paper**	x	x	x
QMP*	Optimal version of QMP	**This paper**	x	x	x
PRM	Probabilistic roadmap planner	(Kavraki et al. 1996)		x	
PRM*	Optimal version of PRM	(Karaman and Frazzoli 2011)		x	
LazyPRM*	Optimal version of LazyPRM	(Karaman and Frazzoli 2011)		x	
SPARS	Sparse roadmap spanners	(Dobson and Bekris 2014)		x	x
SPARS2	SPARS without dense graph	(Dobson and Bekris 2014)		x	x
RRT	Rapidly-exploring random tree	(Lavalle 1998)		x	
RRTConnect	Bidirectional RRT	(Kuffner and LaValle 2000)		x	
RRT*	Optimal version of RRT	(Karaman and Frazzoli 2011)		x	
LazyRRT	Lazy edge evaluation RRT	(Kuffner and LaValle 2000)		x	
TRRT	Transition-based RRT	(Jaillet et al. 2010)		x	
BiTRRT	Bidirectional TRRT	(Jaillet et al. 2010)		x	
LBTRRT	Lower bound tree RRT	(Salzman and Halperin 2016)		x	x
RRTX	RRT with pseudo-optimal tree	(Otte and Frazzoli 2016)		x	x
RRT#	RRT sharp	(Arslan and Tsiotras 2013)		x	x
InformedRRT*	Informed search RRT*	(Gammell et al. 2014)		x	x
SORRT*	Sorted InformedRRT*	(Gammell et al. 2014)		x	x
SBL	Single-query bidirectional lazy PRM	(Sánchez and Latombe 2003b)		x	
SST	Stable sparse RRT	(Li et al. 2016)		x	x
STRIDE	Search tree with resolution independent density estimation	(Gipson et al. 2013)		x	
FMT	Fast marching tree	(Janson et al. 2015)		x	x
BFMT	Bidirectional FMT	(Janson et al. 2015)		x	x
BIT*	Batch informed trees	(Gammell et al. 2020)		x	x
ABIT*	Advanced BIT*	(Strub and Gammell 2020)		x	x
EST	Expansive spaces planner	(Hsu et al. 1999)		x	
BiEST	Bidirectional EST	(Hsu et al. 1999)		x	
ProjEST	Projection EST	(Hsu et al. 1999)		x	
KPIECE	Kinodynamic motion planning By interior-exterior cell exploration	(Şucan and Kavraki 2009)		x	
BKPIECE	Bidirectional KPIECE	(Şucan and Kavraki 2009)		x	
LBKPIECE	Lazy BKPIECE	(Şucan and Kavraki 2009)		x	
PDST	Path-directed subdivision tree	(Ladd and Kavraki 2004)		x	

the new state to the nearest neighbors (Line 5.7). We show the rewire operation in Alg. 7. Inside the rewire algorithm, we update the incoming edge of state y by checking if the cost to come from state x (cost from initial state to x) plus the cost to go from x to y is smaller than the cost to come for state y. In that case, we update the graph by removing all incoming edges into y and adding a directed edge from x to y. Contrary to similar implementations (Karaman and Frazzoli, 2011; Salzman and Halperin, 2016), we also update the tree $\mathbf{G}_k$ such that we can use the same restriction sampling method for each algorithm. While the grow method is similar to the RRT* method (Salzman and Halperin, 2016), we note that much of the complexity is encapsulated in the primitive methods (Section 5), which we use to sample, to compute distances, to find sections and to choose a bundle space to grow next.

6.4. QMP

In Alg. 8, we show the QMP algorithm, which we introduced in Orthey et al. (2018). In the QMP algorithm, we differ from QRRT by not growing a tree, but a graph (Kavraki et al., 1996). QMP generalizes PRM in the sense that QMP becomes equivalent to PRM when we choose a single-level abstraction. The algorithm QMP as presented here differs slightly from its original conception (Orthey et al., 2018) in three points. First, we use the epsilon greedy importance (Section 5.3.2) instead of uniform importance to select a bundle space to expand. Second, we use the intrinsic bundle metric (Section 5.2) instead of the quotient space metric, which we found to not scale well to high-dimensional state spaces (see Appendix C). Third, we use the FINDSECTION method to quickly check for sections (Section 5.4).

6.5. QMP*

QMP* is similar as QMP, but we use a different k in each iteration to choose the nearest neighbors. This k is chosen such that the resulting algorithm is almost-surely asymptotically optimal (Karaman and Frazzoli, 2011). In general we use $k = k_{\mathrm{PRM}} \log(N)$ with N being the number of vertices in the graph. See also Solovey and Kleinbort (2020) for recent developments on choosing the parameter k_{PRM}.

6.6. Open source implementation

To make the algorithms freely available, we provide implementations in C/C++, which we split into two frameworks. The first framework is a graphical user interface (GUI) where users can specify fiber bundles by providing URDF (Unified Robotic Description Format) files for each level and specify the bundle structure in an XML (Extensible Markup Language) file. We then provide functionalities to step through each level and to visualize the lowest cost path on each level. The code is freely available on github.[1]

The second framework is the actual implementation of fiber bundles, bundle algorithms, and primitives, which we implement as a submodule of the Open Motion Planning Library (OMPL) (Şucan and Kavraki, 2009). In particular, we encapsulate our code as an OMPL planner class, which we can use for benchmarking (Moll et al., 2015) or analysis. This code is part of OMPL version 1.6.0 and includes a high-level introduction, a tutorial, and additional demos.[2]

7. Analysis of bundle planners

Let $X_K \rightarrow^{\pi_{K-1}} \ldots \rightarrow^{\pi_1} X_1$ be a fiber bundle sequence. We like to prove that, on this fiber bundle sequence, the algorithms QRRT, QRRT*, QMP, and QMP* are probabilistically complete (PC) and that QRRT* and QMP* are asymptotically optimal (AO).

To prove those properties, we use two methods. First, we state three assumptions on the importance function and the datastructures, which we use to establish that restriction sampling is dense. Second, we argue that the bundle algorithms, when using restriction sampling, inherit the PC and AO properties from their single-level counterpart.

7.1. Assumptions

We require three assumptions to hold true.

1. The importance function of each bundle space (Section 5.3) monotonically converges to zero (we select every bundle space infinitely many times)
2. Restriction sampling is dense in X_1
3. If restriction sampling is dense, the graph on the k-th bundle space is space-filling in the connected initial component

whereby the *connected initial component* is the set of points in X_k which are path-connected[3] to $\pi_k(x_I)$, that is, to the projection of the initial state onto the k-th bundle space. A graph is said to be space-filling in a set U, if for any x in U there exists a path in the graph starting at x_I and converges to x (Kuffner and LaValle 2011) (in the limit when running time goes to infinity).

7.2. Proof that restriction sampling is dense

When stripping down to the essentials, we observe that the bundle planners differ on the last level from non-multilevel planners by replacing uniform sampling with restriction sampling. While uniform sampling is dense in the *complete* state space, restriction sampling differs, in that we can prove it to be dense in the connected initial component.

To prove denseness, we need some notations. First, a set U is dense in X if the intersection of U with any non-empty open subset V of X is non-empty (Munkres, 2000). We abbreviate this by saying that a set is dense if its *closure* $cl(U)$, the smallest closed set containing U, contains the space X. When using a sequence of samples $\alpha_1, \alpha_2, \ldots$, we can interpret the sequence as a set $A = \{\alpha_i | \, i \in \mathbb{N}\}$. We can then say that the sequence is dense in the space X if the closure $cl(A)$ contains X (or is equal to).

Let I_k be the connected initial component(on the bundle space k) and let A_k be a restriction sampling sequence. To prove A_k to be dense in I_k, we choose an arbitrary set U in I_k. We then prove that there will be a non-empty intersection of U with A_k, that is, given enough time, we will at least sample once from U. Our proof is inductive, that is, we prove it to be true for $k = 1$, then use this to inductively argue for arbitrary k.

In a preliminary version of the proof (Orthey and Toussaint, 2019), we showed restriction sampling to be dense in the free state space, which is true only if there is a single connected component. To make the proof more general, we replace the free state space here with the connected initial component.

Theorem 1. *A_k is dense in I_k for $k \geq 1$.*
Proof. By induction for $k = 1$, A_1 is dense in X_1 by assumption and therefore dense in I_1 since $I_1 \subseteq X_1$. For the induction step, we can assume A_{k-1} to be dense in I_{k-1}. Let U be a non-empty open subset of I_k. Since U is open, $\pi_{k-1}(U)$ is open (by property of fiber bundle). By induction assumption there exists a y in $A_{k-1} \cap \pi_{k-1}(U)$. Consider an open set V of the preimage $\pi_{k-1}^{-1}(y)$. Since A_k is dense in $\pi_{k-1}^{-1}(A_{k-1})$ (by definition of restriction sampling), there exists an x in $A_k \cap V$ which is a subset of U. Since U was arbitrary, A_k is dense in I_k. □

Due to Theorem 1, we observe that restriction sampling differs from uniform sampling by removing states which cannot be feasible. Therefore, algorithms using restriction sampling maintain all their properties, which we can inherit.

7.3. Inheritance of probabilistic completeness

A motion planning algorithm is probabilistically complete, if the probability that the algorithm will find a path (if one exists) goes to one as time goes to infinity. This property has been proven for sampling-based planners, in the case of a graph (Svestka, 1996) including the case of a tree (Kuffner and LaValle, 2000).

Probabilistic completeness follows in our case directly from the assumptions and our proof that restriction sampling is dense. In particular, let us assume a given motion planning problem to be feasible and containing a solution in the interior of the free space. Since restriction sampling is dense, by assumption, we have a space-filling graph containing a path starting at the initial state and converging to the goal state.

In the grow functions of QRRT, QRRT*, QMP, and QMP*, we directly implement the corresponding versions of RRT, RRT*, PRM, and PRM*, which all have been shown to be probabilistically complete (see corresponding papers listed in Tab. 1). They therefore necessarily need to construct a space-filling graph (tree) (Kuffner and LaValle, 2011) and all bundle space planners, when using restriction sampling, inherit the probabilistic completeness property.

7.4. Inheritance of asymptotical optimality

An algorithm is (almost-surely) asymptotically (near-)optimal (AnO) (Karaman and Frazzoli, 2011; Salzman and Halperin, 2016) if it converges to a cost at most $(1 + \epsilon)$ times the cost of the optimal path. An algorithm is (almost-surely) asymptotically optimal if it is AnO with $\epsilon = 0$.

Similar to probabilistic completeness, we argue that QRRT* and QMP* are asymptotically optimal, since this property is inherited from RRT* and PRM* (Karaman and Frazzoli, 2011), respectively. This is true, since on the last level, we only change the sampling function from uniform to restriction sampling. Since we showed restriction sampling to be dense and we will select the last bundle space infinitely many times, we can be sure that the optimality properties are kept intact. Note that this line of reasoning is slightly different from the proof of asymptotic optimality for HBFMT (Reid et al., 2020), where Reid et al., (2020) define a probability l with which they switch to use uniform sampling, thereby guaranteeing optimality by actually reverting to BFMT. We, however, rely on the denseness property of restriction sampling, thereby avoiding an uniform extension step.

Detailed proofs of asymptotic optimality for sampling-based planner can be found in Karaman and Frazzoli (2011). See also Salzman and Halperin (2016) and Solovey and Kleinbort (2020) for a treatise of asymptotic near-optimality.

8. Evaluation

To show the wide applicability of fiber bundles and bundle algorithms, we apply them to a broad range of planning scenarios. In particular, we evaluate our algorithms on four low-dimensional and eight high-dimensional planning scenarios, including computer animation, pregrasping, multi-robot coordination, and nonholonomic constraints. The dimensionality of the state spaces in the high-dimensional case ranges from 21-dof (box folding) to 100-dof (hypercube). We compare our algorithms with available algorithms implemented in the Open Motion Planning Library (OMPL) as of May 2020 (Moll et al., 2015). References and details of those algorithms are shown in Table 1. All algorithms, except QMP, QMP*, QRRT, and QRRT*, do not use the additional information which fiber bundles provide. We like to show that fiber bundles are helpful to solve scenarios which are near unsolvable using classical sampling-based methods Kavraki et al. (1996); Kuffner and LaValle (2000)

Evaluation Metrics: For all scenarios, we let each algorithm run 10 times with a cut-off time limit of 60s. For the low-dimensional scenarios, we report a success-cost plot showing convergence rate and success rate over time. In this case, we let the algorithms run for 60s and query their current best cost with a 100Hz update frequency (i.e., every 0.01s). For the high-dimensional scenarios, we run two separate evaluations. First, a pure runtime evaluation, where we compare the average runtime on each scenario, comparing against all available OMPL planners. In this case, planners run until they either find a solution or the cut-off time limit has been reached. Second, we report on a success-cost plot for our algorithms against four well performing algorithms from OMPL, namely, RRTConnect, RRT*, BIT*, and LBTRRT. In this case, all algorithms are run for 60s with best cost queries at 100 Hz update frequency.

Hardware: Concerning hardware, we use a 4-core laptop running Ubuntu 18.04 with 20GB of RAM to run the runtime evaluation on the high-dimensional planning problems. For the low-dimensional planning problems and the cost function evaluation on the high-dimensional problems, we use a 4-core laptop running Ubuntu 16.04 with 8GB of RAM. Concerning parameters, our algorithms are set as follows. For the FINDSECTION method, we use $d_{\text{MAX}} = 3$ and $b_{\text{MAX}} = 10$. For path restriction sampling, we use the decay constant $\lambda = 10^{-3}$ and fixed probability $\beta_{\text{fixed}} = 0.1$. For QRRT, we use a maximal distance range of 0.2μ whereby μ is the measure of the space (same value as in RRT or RRTConnect). For QMP, we use $k = 10$ to compute nearest neighbors (same as in PRM). For QMP*, we use the optimal number of nearest neighbors in each iteration as in PRM* (Solovey and Kleinbort, 2020). The choice of primitive methods has been independently optimized using a meta-analysis (See Appendix C). We set any other parameters to be equivalent to the corresponding single-level planner.

State Spaces: In all scenarios, the state spaces of the robots are modeled using the following mathematical spaces. For rigid bodies, we use $SE(2)$ and $SE(3)$, the special Euclidean group in two and three dimensions, respectively (Selig, 2004). The spaces in those groups model all rotations and translations applicable to a rigid body in two or three dimensions. For rotating joints, we use $SO(2)$ and $SO(3)$, the special orthogonal group. The spaces in those groups model all rotations about a fixed point of the robot. For all other kinematic chains with rotational joints and joint limits, we use the Euclidean space $\mathbb{R}^n$ of n dimensions.

8.1. Low-dimensional motion planning

In the low-dimensional motion planning evaluation, we evaluate QMP, QMP*, QRRT, and QRRT* against RRTConnect, RRT*, BIT*, and LBTRRT. This is done on four low-dimensional planning problems as shown in Figure 6. We let all planners run until time out and collect both time to find the first solution and solution cost over time.

8.1.1. 2-dof disk problem (2 levels). The first scenario is a 2-dof disk problem, where a small disk robot needs to traverse a square with a narrow passage in the middle. For our bundle algorithms, we use a projection onto a smaller inscribed disk with half the radius of the original disk (see Figure 6). This creates a fiber bundle as

$$\mathbb{R}^2 \to \mathbb{R}^2 \quad (14)$$

The evaluation results are shown in Figure 5 (Upper left). RRTConnect performs best in terms of quickest convergence to one hundred percent success rate, while BIT* performs best by converging the fastest to the optimal solution. All bundle planners can successfully solve this problem with competitive results both in terms of success rate (QRRT, QMP), and in terms of cost convergence (QRRT*, QMP*).

8.1.2. 3-dof piano mover's problem (2 levels). The second scenario is the piano mover's problem (Schwartz and Sharir, 1983), where a piano has to be moved on a planar floor from one side of a house to the other side. As shown in Figure 6, we impose a fiber bundle by inscribing a simpler shape into the original piano, thus imposing a fiber bundle as

$$SE(2) \to SE(2) \quad (15)$$

The evaluation results are shown in Figure 5 (Upper right). BIT* and RRTConnect outperform in terms of success rate, while BIT* also converges quickest to a low-cost solution. All bundle planners perform slightly worse, but still competitive in terms of runtime and cost convergence.

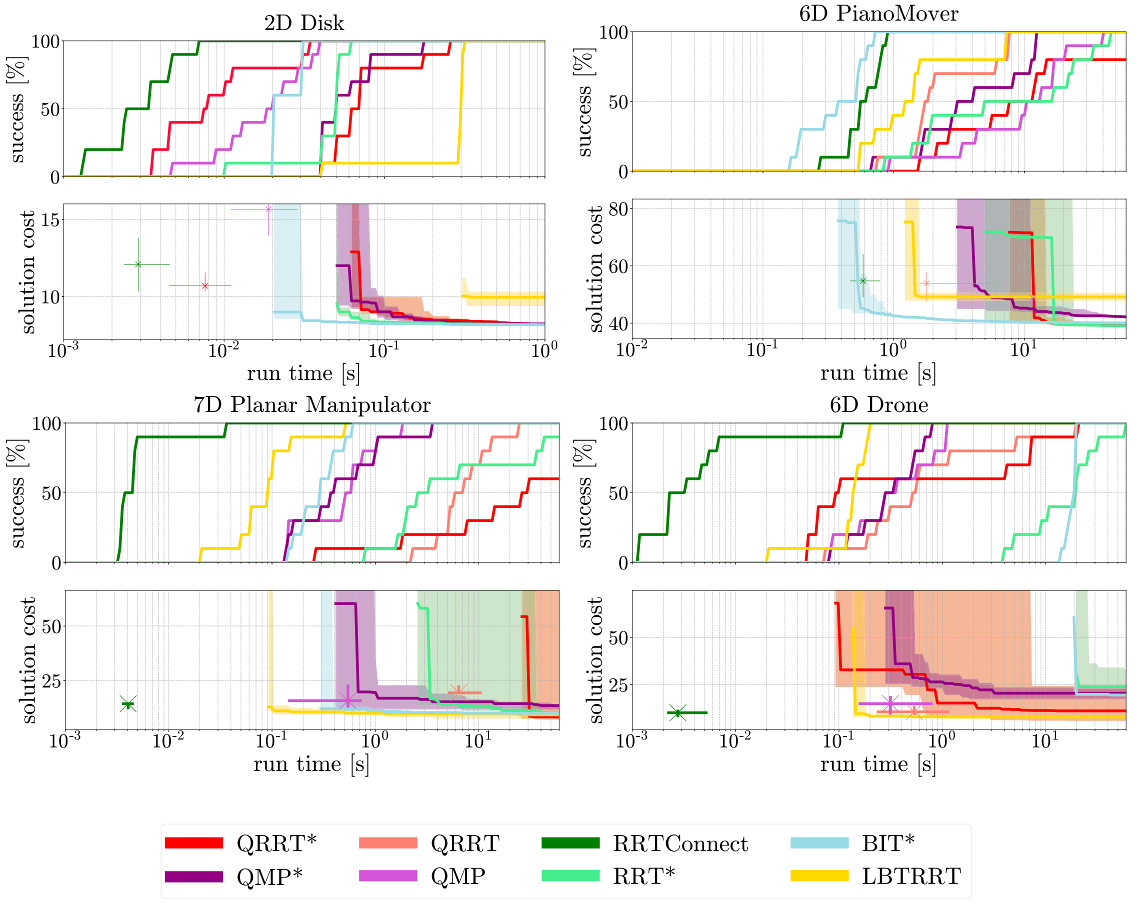

Figure 5. Success-cost plots of the four low-dimensional planning scenarios. (a) 02D disk, (b) 06D Piano Mover's problem, (c) 07D Planar Manipulator, and (d) 06D Double L-Shape.

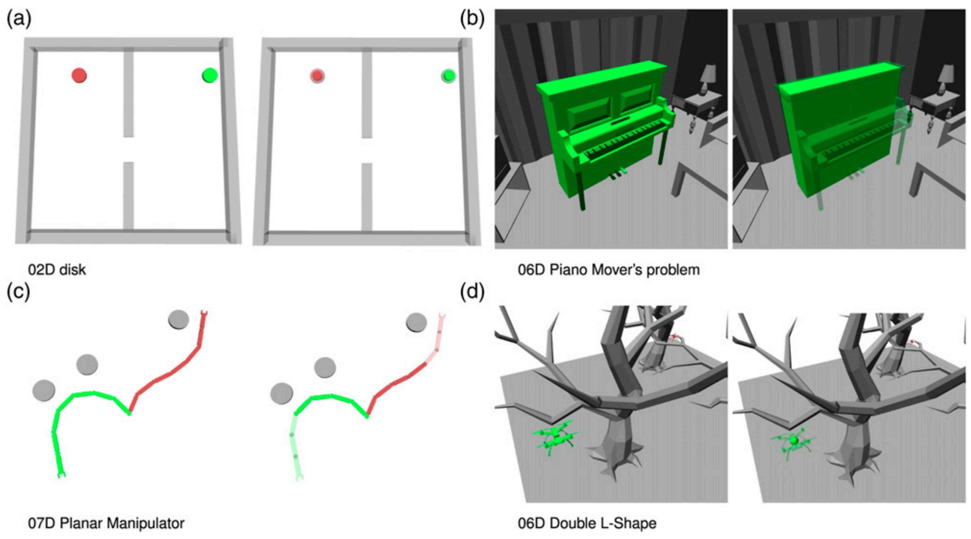

Figure 6. Four scenarios for low-dimensional planning. Start configuration of robot (green) is shown alongside goal configuration (red) when applicable. In each figure, the robot is shown on the original space (left), and with the first projection applied (right), where the original robot is shown with a transparent color.

8.1.3. 7-dof planar manipulator (4 levels). In the third scenario, we evaluate the planners on a 7-dof planar manipulator task, as shown in Figure 6 (Lower left). For this scenario, we impose four levels of abstractions, where we first project the 7-dof robot onto a 4-dof robot by removing the last three links. We then project onto a 2-dof robot by removing two links and finally we project onto a 1-dof robot by removing one link. The resulting fiber bundle can be written as

$$\begin{aligned} &SO(2)\times\mathbb{R}^6 \to SO(2)\times\mathbb{R}^3 \\ &\to SO(2)\times\mathbb{R}^1 \to SO(2). \end{aligned} \tag{16}$$

The evaluation results in Figure 5 (Lower left) show that RRTConnect and LBTRRT perform best in terms of success rate, while LBTRRT converges quickest in terms of solution cost. Both QMP and QMP* perform competitively in terms of success rate and QMP* terms of cost convergence. QRRT has slightly worse performance in terms of success rate, but still solves the problem. QRRT*, however, does not solve all runs of this problem.

8.1.4. 6-dof drone (2 levels). In the fourth scenario, a drone has to traverse two trees to reach a goal state. The state space is $SE(3)$ with additional constraints on roll and pitch, but not yaw (to prevent impossible maneuvers). We impose a fiber bundle as

$$SE(3) \to \mathbb{R}^3 \tag{17}$$

by inscribing a small sphere inside the drone, thereby reducing the state space to $\mathbb{R}^3$. The evaluation results in Figure 5 (Lower right) show RRTConnect to converge quickest in success rate, with LBTRRT being fastest in cost convergence. All bundle planners solve this problem, but QMP* returns a slightly larger final cost compared to the lowest cost found.

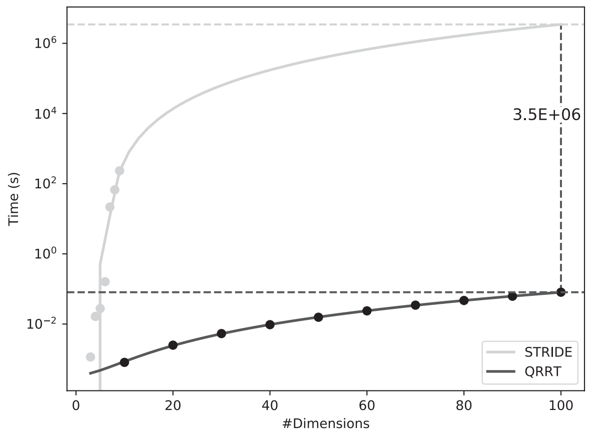

Figure 7. Hypercube scenario comparison of algorithms STRIDE and QRRT.

8.2. High-dimensional motion planning

For the high-dimensional planning scenarios, we conduct two evaluations. First, we run a large set of planners from OMPL until a first solution is found (or a timeout occurs) and report on the runtime. Those results are evaluated for all available planners in OMPL, if they are applicable to the problem at hand. This case is discussed in Section 8.2.1 up to Section 8.2.8. Second, we run the eight planners QMP, QMP*, QRRT, QRRT*, RRTConnect, RRT*, BIT*, and LBTRRT on each scenario until the timeout occurs. We collect both success rate and cost over time and plot those results as success-cost graphs. This case is discussed in Section 8.3. Note that each case uses a different hardware setup as mentioned in Section 8.

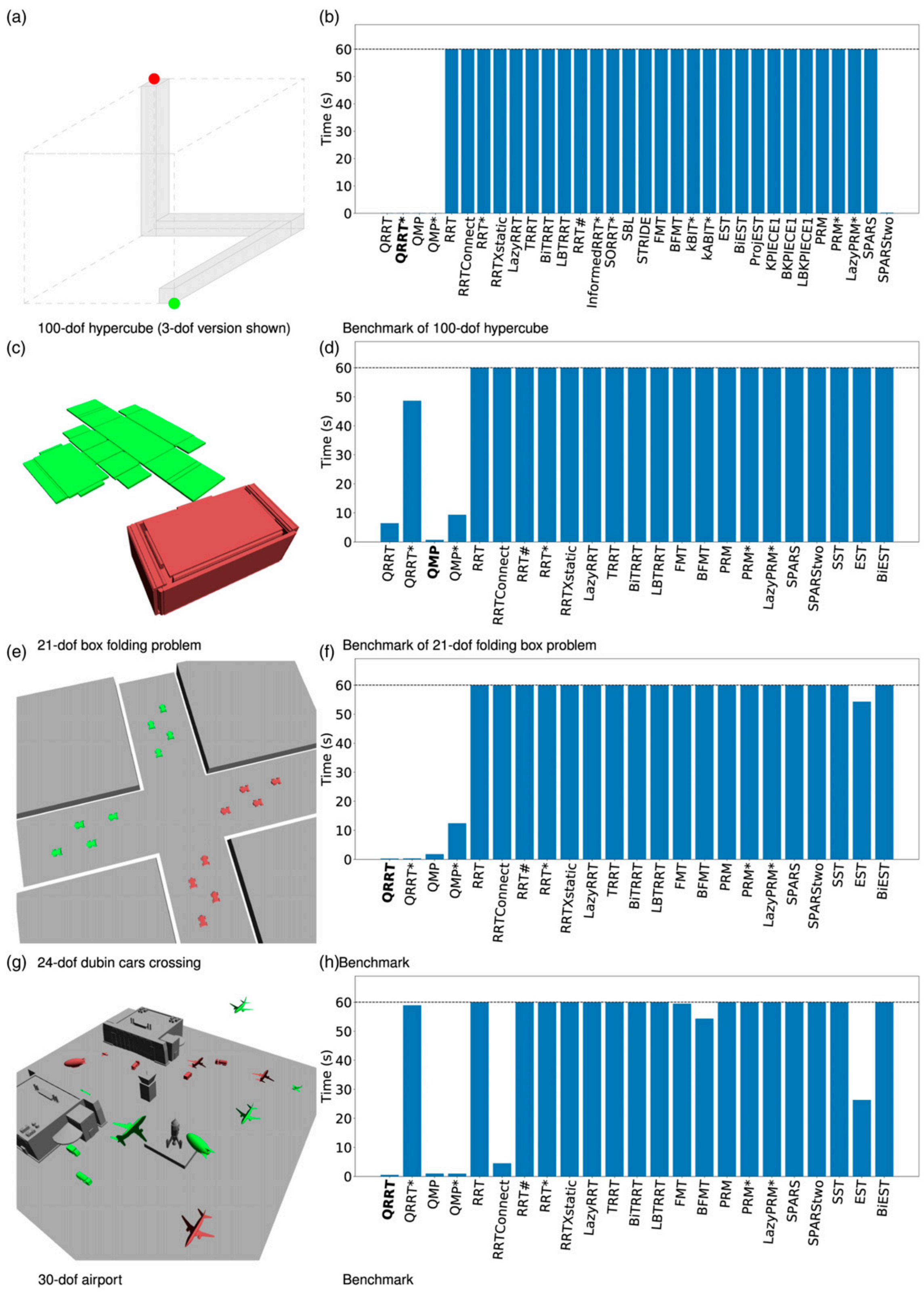

Figure 8. Runtime benchmarks on the first four high-dimensional planning scenarios. (a) 37-dof pre-grasp, (b) Benchmark, (c) 48-dof drones, (d) Benchmark, (e) 54-dof kraken animation, (f) Benchmark, (g) 72-dof manipulators, and (h) Benchmark.

8.2.1. 100-dof hypercube (98 levels). The hypercube (Gipson et al., 2013) is a classical motion planning benchmark, where we need to move a point robot in an n-dimensional cube $X = [0,1]^n$ from $x_I = (0, \ldots, 0)$ to $x_G = (1, \ldots, 1)$. We allow the robot to move only along corridors of size $\epsilon = 0.1$ along the edges of the cube as shown in Figure 8(a). For more details see Gipson et al. (2013). As a fiber bundle, we choose the sequence of reductions

$$[0, 1]^n \to [0, 1]^{n-1} \to \ldots \to [0, 1]^2 \quad (18)$$

where the constraint function is the constraint function of the corresponding cube.

Prior work showed solutions to 25-dimensional cubes in around 100s (Gipson et al., 2013). Here, we attempt to solve a 100-dimensional cube version. The benchmarks are shown in Figure 8(b). All bundle planners have an average runtime of less than 0.1s. Also the non-bundle planner SPARS2 terminates with a runtime of around 0.2s. However, we note that SPARS2 terminates with a probabilistic infeasibility proof, that is, they declare this problem infeasible. Only QRRT, QMP and their star versions can solve this problem in the time limit given. While we terminate all planner at 60s, we can provide a rough estimate of performance improvement of QRRT compared to STRIDE (which outperforms PRM, KPIECE, EST, and RRT (Gipson et al., 2013)). To do that, we let STRIDE run on the $n = \{3, \ldots, 9\}$ dimensional version of the cube, then we extrapolate the results by fitting a cubic curve (see Figure 7). Comparing the extrapolation to QRRT at the dimension 100, we observe that QRRT performs around six orders of magnitude better than STRIDE.

8.2.2. 21-dof box folding (5 levels). To automate deliveries or assemble production pieces, we often need to compute folding motions. Here we concentrate on computing the folding motion of a small packaging box with 21-dof (Figure 8(c)). Such problems are challenging, because parts of the box have to fit into small narrow passages, which is challenging for sampling-based planners. We use a fiber bundle sequence as

$$\begin{aligned} SE(2) \times \mathbb{R}^{18} &\to SE(2) \times \mathbb{R}^{16} \to SE(2) \times \mathbb{R}^{13} \\ &\to SE(2) \times \mathbb{R}^{10} \to SE(2) \times \mathbb{R}^{7} \\ &\to SE(2) \end{aligned} \quad (19)$$

which corresponds to the removal of (1) flaps on lid, (2) lid, (3) right side, (4) left side, (5) back/front elements. We show the benchmarks in Figure 8(d). The best performing algorithm is QMP with 0.68s of planning time. QRRT performs worse with around 6.4s. We discuss this performance difference in Section 9. Both QMP and QRRT together with QMP* outperform all other planning algorithms, that is, no OMPL algorithm was able to solve this scenario in our timelimit.

8.2.3. 24-dof Dubins cars crossing (3 levels). With several companies pushing towards autonomous driving, we need increasingly more efficient algorithms to coordinate multiple car-like robots under nonholonomic constraints. We concentrate here on the problem of planning motions for eight Dubins cars (Dubins, 1957), which are cars with constant forward speed, which we can steer left or right. The cars start on different ends of a crossroad (in reverse direction) and we need them to cross the road while avoiding each other (Figure 8(e)). We impose a fiber bundle as

$$SE(2)^8 \to (\mathbb{R}^2)^8 \to (\mathbb{R}^2)^4 \quad (20)$$

which corresponds to the reduction onto a disk inscribed in the car and the removal of the upper four robots, respectively. We show the benchmark in Figure 8(f). QRRT performs best with a planning time of 0.28s closely followed by QRRT* (0.29s) and QMP (1.77s). QMP* performs less well with 12.41s of planning time. Except EST with planning time of around 54s, there was no non-bundle algorithm able to solve this coordination problem in the timelimit given.

8.2.4. 30-dof airport (15 levels). While coordinating motions for multiple cars are essential for traffic coordination, we often need to coordinate multiple vehicles in 3D under nonholonomic constraints. One particular instance of this problem is an airport, in which we might need to coordinate cars, planes, and zeppelins, each with different state spaces and different possible nonholonomic constraints. Here, we use a scenario with three trucks, 1 zeppelin, 1 propeller plane, 1 airplane while taxiing[4] and two airplanes while flying (see Figure 8(g)). This scenario is particularly challenging, since all vehicles have nonholonomic constraints except the zeppelin. We model the dynamics of the trucks and the planes as Dubins car and Dubins airplane (LaValle, 2006), respectively. Note that arbitrary dynamically constraints could be imposed, but there are implementations of Dubins car and airplane spaces available in OMPL, which makes them also useable with other algorithms in the library. We use a prioritization-like abstraction as

$$\begin{aligned} &SE(2)^4 \times SE(3) \times (\mathbb{R}^3 \times SO(2))^3 \to \\ &\mathbb{R}^2 \times SE(2)^3 \times SE(3) \times (\mathbb{R}^3 \times SO(2))^3 \to \\ &SE(2)^3 \times SE(3) \times (\mathbb{R}^3 \times SO(2))^3 \to \\ &\mathbb{R}^2 \times SE(2)^2 \times SE(3) \times (\mathbb{R}^3 \times SO(2))^3 \to \\ &SE(2)^2 \times SE(3) \times (\mathbb{R}^3 \times SO(2))^3 \to \\ &\mathbb{R}^2 \times SE(2) \times SE(3) \times (\mathbb{R}^3 \times SO(2))^3 \to \\ &SE(2) \times SE(3) \times (\mathbb{R}^3 \times SO(2))^3 \to \\ &\mathbb{R}^2 \times SE(3) \times (\mathbb{R}^3 \times SO(2))^3 \to \\ &SE(3) \times (\mathbb{R}^3 \times SO(2))^3 \to \\ &(\mathbb{R}^3 \times SO(2))^3 \to \\ &\mathbb{R}^3 \times (\mathbb{R}^3 \times SO(2))^2 \to \\ &(\mathbb{R}^3 \times SO(2))^2 \to \\ &\mathbb{R}^3 \times (\mathbb{R}^3 \times SO(2)) \to \\ &(\mathbb{R}^3 \times SO(2)) \to \\ &\mathbb{R}^3 \end{aligned} \quad (21)$$

where the first four $SE(2)$ spaces represent the three trucks and the taxiing airplane. The $SE(3)$ space represents the zeppelin and the remaining three spaces of $\mathbb{R}^3 \times SO(2)$ represent the two flying airplanes and the propeller plane, respectively. Each projection either projects an $SE(2)$ space by using the simpler robots of a nested disk, by removing a robot completely (and its geometry) or by nesting an inscribed sphere. The benchmarks are shown in Figure 8(h). The best performing planner are QRRT (0.52s), QMP (0.99s) and QMP* (0.94s). QRRT* performs significantly worse with a planning time of around 58s, which suggest that it could not completely solve this problem in the time allocated. Besides the bundle planner, we also observe that RRTConnect shows competitive results with 4.5s of planning time.

8.2.5. 37-dof pregrasp (3 levels). Manipulation of objects is a challenging task for robots (Dafle et al., 2018; Driess et al., 2020), in particular if we have to deal with realistic hands with many dofs. We concentrate here on computing a pregrasp for a 37-dof shadow-hand robot mounted on a KUKA LWR robot. We define the problem as finding a pregrasp for the grasping of a small glass, as we depict in Figure 9(a).We impose a fiber bundle as

$$\mathbb{R}^{31} \rightarrow \mathbb{R}^{18} \rightarrow \mathbb{R}^{13} \tag{22}$$

which corresponds to a reduction by first removing all fingers except thumb and index finger and second removing the thumb. The benchmark for this problem is shown in Figure 9(b). Both QMP and QMP* perform best with around 6.81s and 12.36s of planning time. In this scenario, no non-bundle planner can solve this problem. Please note that the planner QRRT and QRRT* perform around 44s and 48s. We discuss this performance further in Section 9.

8.2.6. 48-dof drones (8 levels). Planning motions for multiple quadrotors (Hönig et al., 2018) is essential for drone delivery, in disaster response scenarios and for entertainment purposes. We consider here the problem of coordinating the motion of eight drones which have to traverse a small forest-like environment as shown in Figure 9(c). We use the fiber bundle

$$SE(3)^8 \rightarrow SE(3)^7 \rightarrow \cdots \rightarrow SE(3) \tag{23}$$

which corresponds to a prioritization of the drones, that is, in each projection we remove one robot. The benchmarks are shown in Figure 9(d). While the best algorithm is QRRT (0.14s) closely followed by QMP (0.15s) and QMP* (0.16s), we observe that also RRTConnect and BFMT show competitive performances with 0.59s and 6.05s, respectively.

8.2.7. 54-dof Kraken animation (17 levels). Computer animation is an important application of planning algorithms (Plaku et al., 2018). In animations for movies, an animator would probably insert keyframes to guide the planning of motions. However, if we like to compute animations online, for example, for a computer game, we require fast planning algorithms.

We show here the problem of animating a 54-dof Kraken-like robot (see Figure 9(e)), which has to wrap its arms around a sailing ship. We use a fiber bundle reduction as

$$\begin{aligned} SE(3)\times\mathbb{R}^{48} &\rightarrow SE(3)\times\mathbb{R}^{45} \rightarrow SE(3)\times\mathbb{R}^{42} \\ &\rightarrow \ldots \rightarrow \\ SE(3)\times\mathbb{R}^{6} &\rightarrow SE(3)\times\mathbb{R}^{3} \rightarrow SE(3) \end{aligned} \tag{24}$$

which corresponds to the removal of each arm (6-dof revolute joints) on each stage, whereby we first remove the last three links (removal of 3-dof) and then remove the remaining arm (3-dof). We show the benchmark in Figure 9(f). We observe that both QRRT (0.20 s) and QMP (0.23 s) perform below 1s to find a feasible solution. Next comes QMP* with a planning time of 6.21 s. The next best non-bundle planner is BiTRRT with a performance of around 22 s planning time. The performance of the bundle planner QRRT is thus two orders of magnitude better than the next best non-bundle planner.

8.2.8. 72-dof manipulators (3 levels). When automating construction work (Hartmann et al., 2020) or warehouse operations (Salzman and Stern, 2020; Eppner et al., 2016), we often need to coordinate multiple robots with many dofs. Here, we consider the coordination of eight KUKA manipulators on disk-shaped mobile bases. Each manipulator starts around a circle and needs to change position with its antipodal partner (see Figure 9(g)). We impose a fiber bundle as

$$\left(SE(2)\times\mathbb{R}^6\right)^8 \rightarrow \left(\mathbb{R}^2\right)^8 \rightarrow \left(\mathbb{R}^2\right)^4 \tag{25}$$

which corresponds to the removal of arms and the removal of the upper half of the robots. The benchmarks are shown in Figure 9(h). We observe that QRRT solves this problem in 3.65 s while QRRT* requires 19 s. Only one non-bundle planner is able to terminate on average before the timelimit: RRTConnect with around 39 s seconds of planning time. Note that this problem is difficult for QMP (57 s) and QMP* (50 s) which perform worse than RRTConnect.

8.3. Cost analysis of high-dimensional scenarios

So far, planners have been evaluated with respect to runtime. To also evaluate the cost convergence property, we compare both QRRT* and QMP* on all eight high-dimensional scenarios to QMP, QRRT, BIT*, RRT*, LBTRRT, and RRTConnect. The results are shown in Figure 10.

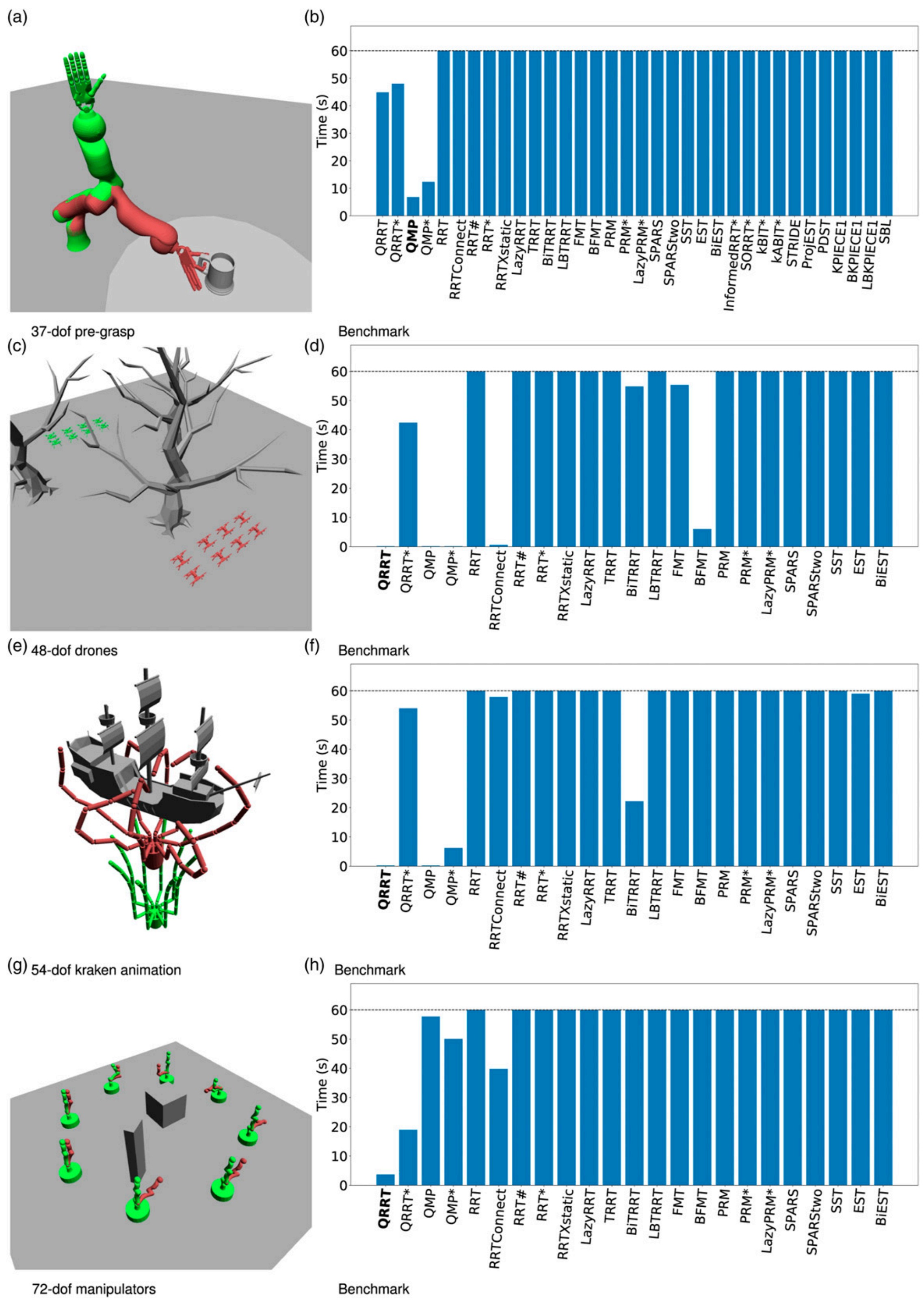

Figure 9. Runtime benchmarks on the last four high-dimensional planning scenarios.

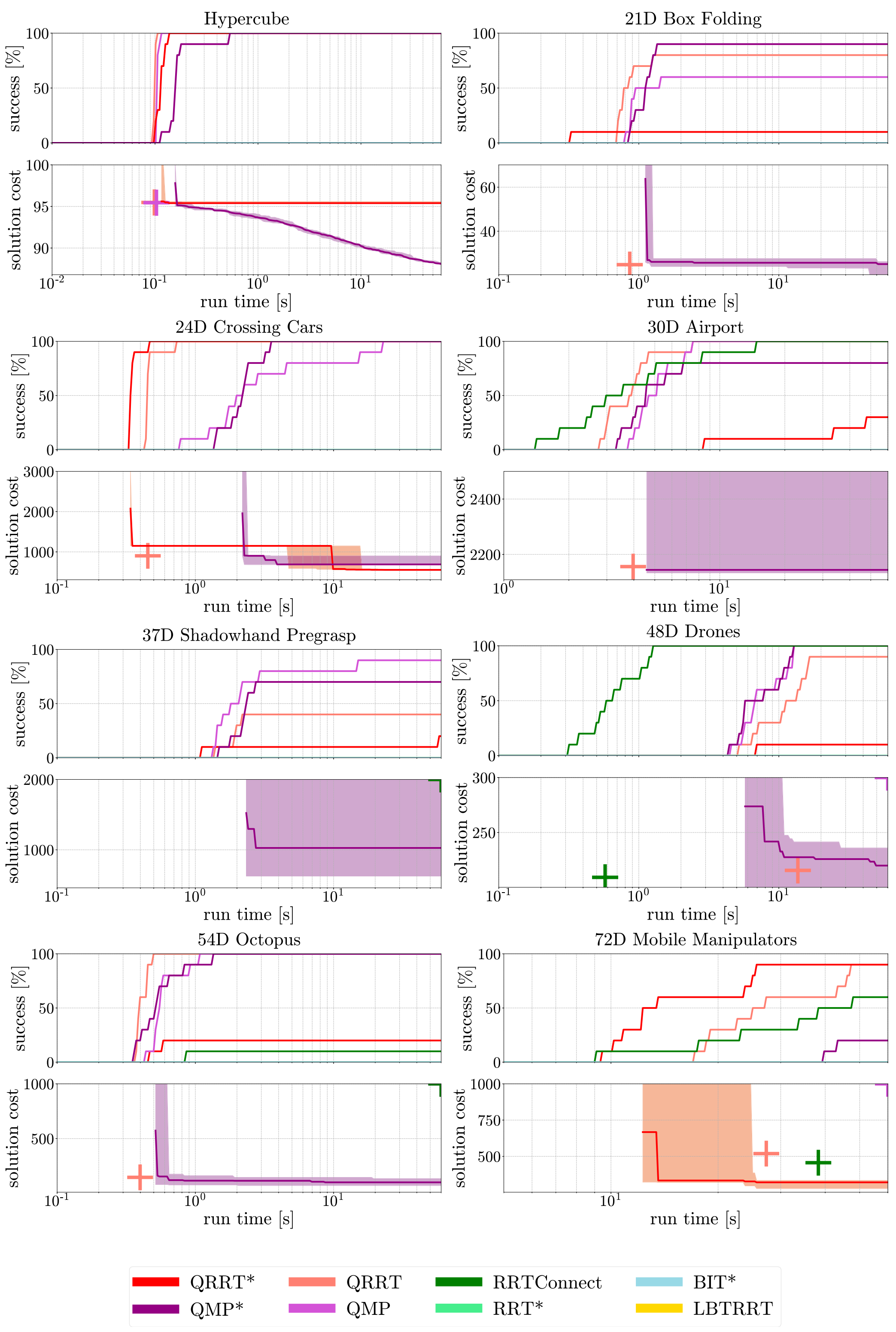

Figure 10. Success-cost plots of the eight high-dimensional planning scenarios.

Let us detail the performance of each algorithm class. First, the non-bundle space planners are only able to tackle two out of eight scenarios. RRTConnect is able to solve the airport and the drones scenario by quickly converging to 100% success rate. In the drones scenario, RRTConnect also finds good, low-cost solutions before any other planner has even found a single solution. However, apart from RRTConnect, the planners RRT*, BIT*, and LBTRRT are not applicable to any of the scenarios with no solved run during the time budget.

Second, the bundle space planners QMP, QMP*, QRRT, and QRRT* are able to tackle all eight scenarios. For the hypercube, QMP, QRRT, and QRRT* quickly find a solution, but are not able to improve upon it. QMP* finds a solution slightly later, but is able to continuously improve upon it. In the box folding task, QMP* is able to solve 90% of the cases while converging quickly to a low-cost solution. Both QRRT and QMP have lower success rates, but find on average a low-cost solution. QRRT*, however, is not able to adequately solve this problem with a success rate of 10%. For the crossing cars scenario, all bundle planners reach 100% success rate with both QRRT* and QMP* converging to low-cost solutions over time. For the airport scenario, QRRT and QMP reach 100% success rate, while both QRRT* and QMP* reach only 80% and 30%, respectively. In terms of cost convergence, QMP* is not able to improve the initial solution cost and has a large cost variance as indicated by the large shaded region around the average cost.

In the Shadow-hand scenario, QMP, and QMP* reach 90% and 70% success rate, while QRRT, and QRRT* reach only 40% and 20%. While QMP* is able to improve the solutions slightly, it has a large variance around the average cost. For the drones scenarios, both QMP and QMP* reach 100% with QMP* converging over time to good low-cost solutions. QRRT is competitive with 90% success rate and low-cost average solution as indicated by the cross in the cost plot. However, QRRT* is only able to solve 10% of the runs. For the octopus scenario, QMP, QMP*, and QRRT reach 100% success rate, while QRRT* only reaches 20%. QMP* shows quick, and low-variance convergence to an optimal solution. Finally, in the mobile manipulators scenario, QRRT* and QRRT reach 90% success rate, while QMP* reaches 20% and QMP fails to find any solutions. QRRT* is also able to converge quickly over time, reaching a solution cost significantly below solution costs from QRRT, and RRTConnect.

9. Discussion

From the preceding evaluation section, we have supporting evidence to draw three broad conclusions. First, it is difficult to solve high-dimensional planning problems with classical (non-bundle) motion planning algorithms. This should not be surprising, since the problem is known to be NP-hard (Hopcroft et al., 1984; Canny, 1988; Solovey, 2020) and the spaces to contain multiple narrow passages (Lozano-Pérez and Wesley, 1979; Salzman et al., 2013).

Second, we can often quickly and reliably solve high-dimensional planning problems by exploiting fiber bundles. We believe there are three primary contributing factors. First, we have expansions of narrow passages. If we project a narrow passage onto a base space, we often observe the narrow passage to increase its volume relative to the surrounding space. We thereby increase our chance to sample narrow passages on the base space, which we can use to guide sampling on the total space (Orthey and Toussaint, 2019). Second, we have the removal of infeasible preimages. If we find a point on the base space to be infeasible, we can remove their preimage from the bundle space, thereby removing *regions* which cannot be feasible (Orthey et al., 2018). Third, we have dedicated methods to exploit admissible heuristics. If we have a path on the base space, we can often quickly find solutions using the recursive path section method or by using path restriction sampling (Zhang et al., 2009). By staying on the path restriction, we exploit the information from the base space, similar to how we would exploit an admissible cost-to-go heuristic in a discrete search scenario (Pearl, 1984; Aine et al., 2016).

Third, the cost analysis showed that bundle space planners can successfully converge to low-cost solutions in high-dimensional spaces. However, this seems to only hold true for QMP*, which outperforms QRRT* in terms of cost convergence in seven out of eight scenarios, as shown in Section 8.3. QRRT*, however, has inferior performance compared to QMP* and only outperforms QMP* in the mobile manipulators scenario. We believe this is due to QRRT* using tree rewiring, which is an expensive operation. Instead, QMP* does not rely on such an operation and is better suited to tackle high-dimensional spaces.

While our evaluation seems to corroborate those statements, we also like to discuss two limiting issues. The first issue are evaluation outlier, which seemingly contradict our statements. We discuss what they are and what we can do about them. The second issue is our reliance on prespecification of fiber bundles, which we do for this work manually. We discuss options to automatically specify them in the future.

9.1. Evaluation outlier

From the evaluations, we observe that we often can find solutions over multilevel abstractions quickly and reliably. However, we observe three noteworthy exceptions. First, we observe that QRRT performs below 3s on every enviroment, except the 37-dof pregrasp (43s) and the box folding task (8s). The cost analysis further shows that QRRT is often not able to reach the 100% success rate. We believe those environments to be challenging for QRRT, because they are examples of ingress problems, that is, problems where we need to enter a narrow passage, similar to a Bugtrap (Yershova et al., 2005). Such problems could be overcome in future work by developing a bidirectional version of QRRT, by using biased sampling towards narrow passages (Yang and Lavalle, 2004), or by selectively expanding states at the frontier of the tree (Yershova et al., 2005; Denny et al., 2020).

Second, we observe QRRT* to perform worse by an order of magnitude compared to QRRT on five out of eight environments. The cost analysis corroborate this observation by showing that QRRT* performs worse in cost convergence on seven out of eight environments when compared against QMP*. We believe the rewiring of the tree in Alg. 6 slows down planning over multilevel abstractions.

In the future, we could overcome this by either postpone rewiring of the tree until a solution is found or by exploiting informed sets (Gammell et al., 2014), which are admissible lower bounds on the optimal solution. It could also be fruitful to investigate the connection between quotient space metrics and the geometric shape of informed sets, which we could use as admissible heuristics (Gammell et al., 2020).

Third, we observe that the non-bundle planner RRTConnect performs competitively on the 30-dof airport and the 48-dof drones environment. Also BFMT performs competitively on 48-dof drones. It seems, we could solve both problems without using fiber bundles. We believe this to happen because both scenarios involve $SE(3)$ state spaces, where narrow passages might be rarer than in $SE(2)$ scenarios. In those environments, we therefore have enough volume to quickly find valid samples, which we can exploit using RRTConnect, or BFMT. However, we believe fiber bundles are still needed. First, we do not know if RRTConnect or BFMT would still perform well if we further increase dimensionality. Second, only by using bundle planners can we consistently and reliably find solutions in all environments. Third, fiber bundles are often the only option if we want to rapidly establish infeasibility or organize local minima over high-dimensional state spaces (Orthey et al., 2020). It is, however, necessary to investigate how narrow passages slow down planning and how we could overcome them using fiber bundles. We previously conducted some evaluations in that direction for the QRRT planner (Orthey and Toussaint, 2019).

9.2. Specifying fiber bundles

For each problem, fiber bundles have to be specified manually. This is problematic, since there is no clear guideline on how to select fiber bundles for a specific problem. This could be overcome by optimizing over a primitive set of fiber bundles. To create a primitive set of fiber bundles, we could use the largest inscribed sphere for a rigid body, the removal of links from a chain, or the removal of nonholonomic constraints from a dynamical system. We can then search the landscape of such primitive fiber bundles to find an efficient fiber bundle for a specific robot and a specific set of environments. A recent study by Brandao and Havoutis (2020) shows promising results in that direction by using evolutionary algorithms to select an abstraction. It could also be promising to use workspace information to select a fiber bundle (Yoshida, 2005), either by choosing joints which can actuate links of interest through the workspace (Luna et al., 2020) or by choosing a bundle on-the-fly based on which links are closest to obstacles (Kim et al., 2015). We thereby could choose different fiber bundles for large rooms, for narrow passages or for ingress tasks. However, in those cases, we would need to consider fiber bundles with changing dimensions, which are in general given by the concept of a sheaf (Bredon, 2012).

10. Conclusion

We modeled multilevel motion planning problems using the framework of fiber bundles. To exploit fiber bundles, we developed a set of bundle primitives, and the bundle planners QRRT* and QMP*, which we showed to be probabilistically complete and asymptotically optimal. We also extended the existing bundle planners QRRT (Orthey and Toussaint, 2019) and QMP (Orthey et al., 2018) using an exponential importance criterion and a recursive L1 path section method (Figure 1). We conducted a meta-analysis to find the best implementation of the bundle primitives, including graph sampling, metric, importance selection, and path section methods. Using the bundle planners, we robustly and efficiently solved challenging high-dimensional motion planning problems, from 21-dof to 100-dof. We also showed competitive results for low-dimensional scenarios, and we showed QMP* to be superior in cost convergence for high-dimensional scenarios.

However, we believe there is still room for improvement. In particular, runtime could be further reduced by developing a bidirectional version of QRRT (LaValle and Kuffner Jr, 2001), by improving convergence using informed sets (Gammell et al., 2014), by investigating novel path section optimization methods (Zhang et al., 2009), and by automatically searching fiber bundles to exploit (Kim et al., 2015; Brandao and Havoutis, 2020)—that is, with respect to a given bundle algorithm (Orthey and Toussaint, 2019). We also believe it is worthwhile to investigate the connection to complementary approaches, like computing neighborhoods (Lacevic and Osmankovic, 2020) and exploiting sufficiency conditions (Grey et al., 2017).

However, despite room for improvements, we showed that bundle planners can efficiently exploit fiber bundles. By exploiting fiber bundles, bundle planners outperformed existing planners often by up to two orders of magnitude, occasionally up to six orders of magnitude. Thus, we believe to not only have contributed to solving multilevel planning problems in the now, but also to have contributed tools and insights to investigate high-dimensional state spaces in the future.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The authors disclose receipt of the following financial support for the research, authorship and publication of this article: This work was supported by the Alexander von Humboldt Foundation [individual grant], the Japan Society for the Promotion of Science [individual grant] and the Max-Planck Society [fellowship grant].

ORCID iD

Andreas Orthey https://orcid.org/0000-0002-1478-1405

Notes

1. https://github.com/aorthey/MotionExplorer
2. https://ompl.kavrakilab.org/multiLevelPlanning.html
3. We say that two states are path-connected if there exists a continuous path connecting them.
4. Taxiing refers to movements of an airplane on the ground, for example after landing or before take-off.

References

Aine S, Swaminathan S, Narayanan V, et al. (2016) Multi-heuristic a. *The International Journal of Robotics Research* 35(1–3): 224–243.

Amato NM, Bayazit OB, Dale LK, et al. (1998) Obprm: an obstacle-based prm for 3d workspaces. In: *Workshop on the Algorithmic Foundations of Robotics*, pp. 155–168.

Arslan O and Tsiotras P (2013) Use of relaxation methods in sampling-based algorithms for optimal motion planning In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 2421–2428.

Baginski B (1996) Local motion planning for manipulators based on shrinking and growing geometry models In: *IEEE International Conference on Robotics and Automation*. Citeseer, pp. 3303–3308.

Ballard DH (2015) *Brain Computation as Hierarchical Abstraction*. MIT press.

Bayazit OB, Xie D and Amato NM (2005) Iterative relaxation of constraints: a framework for improving automated motion planning In: *IEEE International Conference on Intelligent Robots and Systems*, pp. 3433–3440.

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.

Bhattacharya S, Likhachev M and Kumar V (2012) Topological constraints in search-based robot path planning. *Autonomous Robots* 33(3).

Bialkowski J, Otte M, Karaman S, et al. (2016) Efficient collision checking in sampling-based motion planning via safety certificates. *The International Journal of Robotics Research* 35(7): 767–796.

Bobrow JE, Dubowsky S and Gibson J (1985) Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research* 4(3): 3–17.

Bohlin R and Kavraki LE (2000) Path planning using lazy prm. *IEEE International Conference on Robotics and Automation* 1: 521–528.

Boor V, Overmars MH and Van Der Stappen AF (1999) The Gaussian sampling strategy for probabilistic roadmap planners. *IEEE International Conference on Robotics and Automation* 2: 1018–1023.

Boyd S and Vandenberghe L (2004) *Convex Optimization*. Cambridge University Press.

Brandao M and Havoutis I (2020) Learning sequences of approximations for hierarchical motion planning. *International Conference on Automated Planning and Scheduling* 30: 508–516.

Branicky MS, LaValle SM, Olson K, et al. (2001) Quasi-randomized path planning. *IEEE International Conference on Robotics and Automation* 2, pp. 1481–1487.

Bredon GE (2012) *Sheaf Theory.* Springer Science and Business Media, volume 170.

Bretl T (2006) Motion planning of multi-limbed robots subject to equilibrium constraints: the free-climbing robot problem. *The International Journal of Robotics Research* 25(4): 317–342.

Bungartz HJ and Griebel M (2004) Sparse grids. *Acta Numerica* 13: 147–269.

Burns B and Brock O (2005) Toward optimal configuration space sampling *Robotics: Science and Systems*. Cambridge, USA, pp. 105–112.

Canny JF (1988) *The Complexity of Robot Motion Planning*. MIT press.

Cortés J, Jaillet L and Siméon T (2008) Disassembly path planning for complex articulated objects. *Transactions on Robotics* 24(2): 475–481.

Dafle NC, Holladay R and Rodriguez A (2018) In-hand manipulation via motion cones *Robotics: Science and Systems*. Pittsburgh: Pennsylvania, pp. 19–31.

Deits R and Tedrake R (2014) Footstep planning on uneven terrain with mixed-integer convex optimization *IEEE International Conference on Humanoid Robots*. IEEE, pp. 279–286.

Denny J, Sandström R, Bregger A, et al. (2020) Dynamic region-biased rapidly-exploring random trees *Algorithmic Foundations of Robotics XII*. Springer, pp. 640–655.

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

Driess D, Ha JS and Toussaint M (2020) Deep visual reasoning: learning to predict action sequences for task and motion planning from an initial scene image *Robotics: Science and Systems*.

Dubins LE (1957) On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics* 79(3): 497–516.

Edelkamp S and Schroedl S (2011) *Heuristic Search: Theory and Applications*. Elsevier.

Eppner C, Höfer S, Jonschkowski R, et al. (2016) Lessons from the amazon picking challenge: four aspects of building robotic systems *Robotics: Science and Systems*. Michigan: AnnArbor.

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

Farber M (2008) *Invitation to Topological Robotics*. European Mathematical Society, volume 8.

Farber M (2017) Configuration spaces and robot motion planning algorithms. *Combinatorial And Toric Homotopy: Introductory Lectures* 35: 263.

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

Gammell JD, Srinivasa SS and Barfoot TD (2014) Informed RRT*: optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic *IEEE International Conference on Intelligent Robots and Systems*. IEEE, pp. 2997–3004.

Gammell JD, Barfoot TD and Srinivasa SS (2018) Informed sampling for asymptotically optimal path planning. *Transactions on Robotics* 34(4): 966–984.

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

Giles MB (2015) Multilevel Monte Carlo methods. *Acta Numerica* 24: 259–328.

Gipson B, Moll M and Kavraki LE (2013) Resolution independent density estimation for motion planning in high-dimensional spaces *IEEE International Conference on Robotics and Automation*. IEEE, pp. 2437–2443.

Gochev K, Safonova A and Likhachev M (2012) Planning with adaptive dimensionality for mobile manipulation *IEEE International Conference on Robotics and Automation*, pp. 2944–2951.

Gochev K, Safonova A and Likhachev M (2013) Incremental planning with adaptive dimensionality *International Conference on Automated Planning and Scheduling*.

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

Guo X, Srivastava A and Sarkar S (2019) *A Quotient Space Formulation for Statistical Analysis of Graphical Data*. arXiv e-prints.

Ha JS, Park SS and Choi HL (2019) Topology-guided path integral approach for stochastic optimal control in cluttered environment. *Robotics and Autonomous Systems* 113: 81–93.

Hart PE, Nilsson NJ and Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2): 100–107.

Hartmann VN, Oguz OS, Driess D, et al. (2020) Robust task and motion planning for long-horizon architectural construction planning. IEEE International Conference on Intelligent Robots and Systems.

Hastie T, Tibshirani R and Friedman J (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media.

Hauser K (2015) Lazy collision checking in asymptotically-optimal motion planning In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 2951–2957.

Henkel C and Toussaint M (2020) Optimized directed roadmap graph for multi-agent path finding using stochastic gradient descent *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20)*. Brno: Czech Republic.

Hönig W, Preiss JA, Kumar TS, et al. (2018) Trajectory planning for quadrotor swarms. *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. *The International Journal of Robotics Research* 3(4): 76–88.

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.

Hsu D, Jiang T, Reif J, et al. (2003) The bridge test for sampling narrow passages with probabilistic roadmap planners. *IEEE International Conference on Robotics and Automation* 3: 4420–4426.

Hsu D, Sánchez-Ante G, Cheng H, et al. (2006) Multi-level free-space dilation for sampling narrow passages in prm planning *IEEE International Conference on Robotics and Automation*. IEEE, pp. 1255–1260.

Husemoller D (1966) *Fibre Bundles*. Springer, volume 5.

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

Ivan V, Zarubin D, Toussaint M, et al. (2013) Topology-based representations for motion planning and generalization in dynamic environments with interactions. *The International Journal of Robotics Research* 32(9–10): 1151–1163.

Jaillet L and Porta JM (2013) Path planning under kinematic constraints by rapidly exploring manifolds. *Transactions on Robotics* 29(1): 105–117.

Jaillet L and Siméon T (2008) Path deformation roadmaps: compact graphs with useful cycles for motion planning. *The International Journal of Robotics Research* 27(11–12): 1175–1188.

Jaillet L, Cortés J and Siméon T (2010) Sampling-based path planning on configuration-space costmaps. *Transactions on Robotics* 26(4): 635–646.

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

Janson L, Ichter B and Pavone M (2018) Deterministic sampling-based motion planning: optimality, complexity, and performance. *The International Journal of Robotics Research* 37(1): 46–61.

Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research* 30(7): 846–894.

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

Kim DH, Choi YS, Park T, et al. (2015) Efficient path planning for high-dof articulated robots with adaptive dimensionality *IEEE International Conference on Robotics and Automation*. IEEE, pp. 2355–2360.

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

Kleinbort M, Solovey K, Littlefield Z, et al. (2019) Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation. *Robotics and Automation Letters* 4(2): 277–283.

Konidaris G (2019) On the necessity of abstraction. *Current opinion in behavioral sciences* 29: 1–7.

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

Kuffner JJ and LaValle SM (2011) Space-filling trees: a new perspective on incremental search for motion planning *IEEE International Conference on Intelligent Robots and Systems*. IEEE, pp. 2199–2206.

Lacevic B and Osmankovic D (2020) Improved c-space exploration and path planning for robotic manipulators using distance information *IEEE International Conference on Robotics and Automation*.

Lacevic B, Osmankovic D and Ademovic A (2016) Burs of free c-space: a novel structure for path planning *IEEE International Conference on Robotics and Automation*, pp. 70–76.

Ladd AM and Kavraki LE (2004) Fast tree-based exploration of state space for robots with dynamics *Algorithmic Foundations of Robotics VI*. Springer.

Lavalle SM (1998) *Rapidly-exploring Random Trees: A New Tool for Path Planning*. Iowa State University. Technical report.

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

LaValle SM and Kuffner JJ Jr (2001) Randomized kinodynamic planning. *The International Journal of Robotics Research* 20(5): 378–400.

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

Leskovec J and Faloutsos C (2006) Sampling from large graphs *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 631–636.

Li Y, Littlefield Z and Bekris KE (2016) Asymptotically optimal sampling-based kinodynamic planning. *The International Journal of Robotics Research*.

Lozano-Pérez T (1983) Spatial planning: a configuration space approach. *IEEE Trans. Computers* 32(2): 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, Moll M, Badger J, et al. (2020) A scalable motion planner for high-dimensional kinematic systems. *The International Journal of Robotics Research* 39(4): 361–388.

Ma H, Harabor D, Stuckey PJ, et al. (2019) Searching with consistent prioritization for multi-agent path finding. *AAAI Conference on Artificial Intelligence* 33: 7643–7650.

Mandalika A, Choudhury S, Salzman O, et al. (2019) Generalized lazy search for robot motion planning: interleaving search and edge evaluation via event-based toggles. *International Conference on Automated Planning and Scheduling* 29: 745–753.

Mavrogiannis CI and Knepper RA (2016) Decentralized multi-agent navigation planning with braids. In: *Workshop on the Algorithmic Foundations of Robotics*.

Möbius AF (1858). *Werke* 2: 519.

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

Munkres J (2000) *Topology*. Pearson.

Nguyen MK, Jaillet L and Redon S (2018) Art-rrt: as-rigid-as-possible exploration of ligand unbinding pathways. *Journal of Computational Chemistry* 39(11): 665–678.

Orthey A and Toussaint M (2019) *Rapidly-Exploring Quotient-Space Trees: Motion Planning Using Sequential Simplifications*. International Symposium of Robotics Research.

Orthey A and Toussaint M (2020) Visualizing local minima in multi-robot motion planning using multilevel morse theory. *Workshop on the Algorithmic Foundations of Robotics*.

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

Orthey A, Frész B and Toussaint M (2020) Motion planning explorer: visualizing local minima using a local-minima tree. *Robotics and Automation Letters* 5(2): 346–353.

Otte M and Frazzoli E (2016) Rrtx: asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research* 35(7): 797–822.

Palmieri L, Koenig S and Arras KO (2016) Rrt-based non-holonomic motion planning using any-angle path biasing *IEEE International Conference on Robotics and Automation*. IEEE, pp. 2775–2781.

Palmieri L, Bruns L, Meurer M, et al. (2019) Dispertio: optimal sampling for safe deterministic motion planning. *Robotics and Automation Letters* 5(2): 362–368.

Pappas GJ, Lafferriere G and Sastry S (2000) Hierarchically consistent control systems. *IEEE Transactions on Automatic Control* 45(6): 1144–1160.

Passino KM and Antsaklis PJ (1994) A metric space approach to the specification of the heuristic function for the a* algorithm. *IEEE transactions on systems, man, and cybernetics* 24(1): 159–166.

Pearl J (1984) *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addision Wesley.

Persson SM and Sharf I (2014) Sampling-based a* algorithm for robot path-planning. *The International Journal of Robotics Research* 33(13): 1683–1708.

Pham H and Pham QC (2018) A new approach to time-optimal path parameterization based on reachability analysis. *Transactions on Robotics* 34(3): 645–659.

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

Plaku E (2015) Region-guided and sampling-based tree search for motion planning with dynamics. *IEEE Transactions on Robotics* 31(3): 723–735.

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

Plaku E, Rashidian S and Edelkamp S (2018) Multi-group motion planning in virtual environments. *Computer Animation and Virtual Worlds* 29(6).

Pokorny FT, Hawasly M and Ramamoorthy S (2016a) Topological trajectory classification with filtrations of simplicial complexes and persistent homology. *The International Journal of Robotics Research* 35(1–3): 204–223.

Pokorny FT, Kragic D, Kavraki LE, et al. (2016b) High-dimensional winding-augmented motion planning with 2d topological task projections and persistent homology *IEEE International Conference on Robotics and Automation*, pp. 24–31.

Quinlan S (1994) *PhD thesis Real-time Modification of Collision-free Paths*. Stanford University Stanford.

Reid W, Fitch R, Göktoğan AH, et al. (2019) Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover. *Journal of Field Robotics* 37: 5.

Reid W, Fitch R, Göktogan AH, et al. (2020) Motion planning for reconfigurable mobile robots using hierarchical fast marching trees *Algorithmic Foundations of Robotics XII*. Springer, pp. 656–671.

Richter C, Bry A and Roy N (2016) Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments *Robotics Research*. Springer, pp. 649–666.

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

Röwekämper J, Tipaldi GD and Burgard W (2013) Learning to guide random tree planners in high dimensional spaces *IEEE International Conference on Intelligent Robots and Systems*. IEEE, pp. 1752–1757.

Roubíček T (2011) *Relaxation in optimization theory and variational calculus*. Walter de Gruyter, volume 4.

Russell S and Norvig P (2002) *Artificial Intelligence: A Modern Approach*.

Sánchez G and Latombe JC (2003a) A single-query bi-directional probabilistic roadmap planner with lazy collision checking *Robotics Research: The Tenth International Symposium*. Springer, pp. 403–417.

Sánchez G and Latombe JC (2003b) A single-query bi-directional probabilistic roadmap planner with lazy collision checking *Robotics Research*. Springer, pp. 403–417.

Saha M, Latombe JC, Chang YC, et al. (2005) Finding narrow passages with probabilistic roadmaps: the small-step retraction method. *Autonomous Robots* 19(3): 301–319.

Salzman O (2019) Sampling-based robot motion planning. *Communications of the ACM* 62(10): 54–63.

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

Salzman O and Stern R (2020) Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1711–1715.

Salzman O, Hemmer M and Halperin D (2013) On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages. In: E Frazzoli, T Lozano-Perez, N Roy, et al. (eds), *Algorithmic Foundations of Robotics X*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 313–329.

Schwartz JT and Sharir M (1983) On the “piano movers” problem. ii. general techniques for computing topological properties of real algebraic manifolds. *Advances in applied Mathematics* 4(3): 298–351.

Sekhavat S, Svestka P, Laumond JP, et al. (1998) Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *The International Journal of Robotics Research* 17(8): 840–857.

Selig JM (2004) *Geometric Fundamentals of Robotics*. Springer Science and Business Media.

Shome R, Solovey K, Dobson A, et al. (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.

Simon HA (1969) *The Sciences of the Artificial*. MIT press.

Solovey K (2020) *Complexity of Planning*. arXiv preprint *arXiv: 2003.03632*.

Solovey K and Halperin D (2014) k-color multi-robot motion planning. *The International Journal of Robotics Research* 33(1): 82–97.

Solovey K and Kleinbort M (2020) The critical radius in sampling-based motion planning. *The International Journal of Robotics Research* 39(2–3): 266–285.

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

Steenrod NE (1951) *The Topology of Fibre Bundles*.

Strub MP and Gammell JD (2020) Advanced bit*(abit*): sampling-based planning with advanced graph-search techniques *IEEE International Conference on Robotics and Automation*.

Şucan IA and Kavraki LE (2009) Kinodyn amic motion planning by interior-exterior cell exploration *Algorithmic Foundation of Robotics VIII*. Springer, pp. 449–464.

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

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

Svestka P (1996) *On probabilistic completeness and expected complexity for probabilistic path planning*, volume 1996. Utrecht University: Information and Computing Sciences.

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

Tonneau S, Prete AD, Pettré J, et al. (2018) An efficient acyclic contact planner for multiped robots. *Transactions on Robotics* 34(3): 586–601.

Toussaint M and Lopes M (2017) Multi-bound tree search for logic-geometric programming in cooperative manipulation domains In: *IEEE International Conference on Robotics and Automation*, pp. 4044–4051.

Toussaint M, Allen K, Smith K, et al. (2018) *Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning*. Robotics: Science and Systems.

Tu LW (2017) *Differential geometry: connections, curvature, and characteristic classes*. Springer, volume 275.

Vahrenkamp N, Scheurer C, Asfour T, et al. (2008) Adaptive motion planning for humanoid robots In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE, pp. 2127–2132.

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

Van Den Berg JP and Overmars MH (2005) Prioritized motion planning for multiple robots In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE, pp. 430–435.

Vega-Brown W and Roy N (2018) Admissible abstractions for near-optimal task and motion planning In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 4852–4859.

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

Vonásek V and Pěnîĝka R (2019) Sampling-based motion planning of 3d solid objects guided by multiple approximate solutions In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE, pp. 1480–1487.

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

Wilmarth SA, Amato NM and Stiller PF (1999) Maprm: a probabilistic roadmap planner with sampling on the medial axis of the free space. *IEEE International Conference on Robotics and Automation* IEEE, volume 2, pp. 1024–1031.

Xanthidis MP, Esposito JM, Rekleitis I, et al. (2018) *Analysis of Motion Planning by Sampling in Subspaces of Progressively Increasing Dimension*. arXiv preprint *arXiv:1802.00328*.

Yang L and Lavalle SM (2004) The sampling-based neighborhood graph: an approach to computing and executing feedback motion strategies. *Transactions on Robotics* 20(3): 419–432.

Yershova A, Jaillet L, Siméon T, et al. (2005) Dynamic-domain rrts: efficient exploration by controlling the sampling domain In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 3856–3861.

Yoshida E (2005) Humanoid motion planning using multi-level dof exploitation based on randomized method In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE, pp. 3378–3383.

Yu H, Lu W and Liu D (2019) A unified closed-loop motion planning approach for an i-auv in cluttered environment with localization uncertainty In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 4646–4652.

Yu H, Lu W, Han Y, et al. (2020) Heterogeneous dimensionality reduction for efficient motion planning in high-dimensional spaces. *IEEE Access* 8: 42619–42632.

Zarubin D, Ivan V, Toussaint M, et al. (2012) *Hierarchical Motion Planning in Topological Representations*. Robotics: Science and Systems.

Zhang L, Pan J and Manocha D (2009) *Motion planning of human-like robots using constrained coordination* IEEE International Conference on Humanoid Robots, pp. 188–195.

Zucker M, Kuffner J and Bagnell JA (2008) Adaptive workspace biasing for sampling-based planners In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 3757–3762.

Appendix

A. Background Fiber Bundles

Fiber bundles are based upon the concepts of equivalence relations, and quotient spaces, with close ties to constraint relaxation, and admissible heuristics. We provide here a short overview about those concepts.

A.1. Equivalence Relations. An equivalence relation $\sim$ is a binary relation on a space X such that for any elements $x, y, z \in X$ we have $x \sim x$ (reflexive), if $x \sim y$ then $y \sim x$ (symmetric) and if $x \sim y$ and $y \sim z$ then $x \sim z$ (transitive) (Munkres 2000).

An equivalence relation partitions the space X into disjoint subsets we call equivalence classes (Munkres 2000). Given an element x in X, the equivalence class of x is the set of elements $[x] = \{y | y \sim x\}$.

A.2. Quotient Spaces. We often like to simplify a space X under an equivalence relation $\sim$ by taking the quotient. Taking the quotient means that we compute the quotient space $Q = X/\sim$ under the quotient map $\pi\colon X \to Q$. The quotient space is the set of all equivalence classes imposed by $\sim$ on X. To manipulate those equivalence classes, we can often *represent* the quotient space by assigning an equivalence class to a point of a representative space. We define this representative space as a space B under a (bijective) representative mapping ν: $Q \to B$ (Lee 2003).

Let us consider an example. In Figure 11 (Left) we show the plane $\mathbb{R}^2$ with elements $x = (x_1, x_2)$ under the equivalence relation of vertical lines, that is, $x \sim x'$ if $x_1 = x_1'$. An equivalence class $[x] = \{x' | x' \sim x\}$ represents a vertical line, that is, the set of points in $\mathbb{R}^2$ with equivalent

x_1 value. Taking the quotient, we obtain the quotient space $Q = \mathbb{R}^2/\sim$, the set of vertical lines in $\mathbb{R}^2$ (Figure 11 Middle). We can then *represent* Q by the representative space $\mathbb{R}^1$ by associating to each equivalence class (vertical line) the real value x_1 using the representative mapping $v : Q \to \mathbb{R}^1$ we define as $v([x]) = x_1$ (Figure 11 Right).

A.3. Constraint Relaxation. To approximate a complex problem, we can often use the concept of constraint relaxation. Let X be a space and $\phi : X \to \mathbb{R}$ be a constraint function on X. To solve a planning problem on X, we need to search through the free space X_f, which might have zero-measure constraints or narrow passages. To simplify such a problem, we replace the constraint function ϕ by a constraint relaxation function ϕ_R under the condition

$$\phi_R(x) \leq \phi(x) \tag{26}$$

for any x in X.

We can explain this condition geometrically as an expansion of the free space X_f when using ϕ_R (Orthey and Toussaint 2019). Constraint relaxations (Roubíček 2011) are advantageous, because we can use solutions of the relaxed problem as certified lower bounds on the solution of the original problem.

A.4 Admissible Heuristics

In a search problem, we like to find paths through a state space X to move from an initial element $x_I \in X$ to a goal element $x_G \in X$. When casting this as a search problem, we often like to know which state to expand next. A helpful tool is the cost-to-go (or value) function $h^* : X \to \mathbb{R}$ which defines the cost of the optimal path from any point to the goal. An admissible heuristic is an estimate $h : X \to \mathbb{R}$ which lower bounds $h*$ as

$$h(x) \leq h^*(x) \tag{27}$$

for any x in X (Pearl 1984; Edelkamp and Schroedl 2011; Aine et al. 2016). Admissible heuristics are important because we can use them to guarantee optimality and completeness in algorithms like A* (Hart et al. 1968; Pearl 1984) and to often decrease planning time significantly (Aine et al. 2016).

B. Exponential Change

To model quick but smooth transitions between two parameter values, we use an exponential decay function. Let κ_0 be the start and κ_1 be the final parameter value. We model the change between κ_0 and κ_1 using the exponential decay function

$$\kappa(t) = (\kappa_0 - \kappa_1)\exp(-\lambda t) + \kappa_1 \tag{28}$$

with $t \in \mathbb{R}_{\geq 0}$ being the time or iteration number, $\kappa(0) = \kappa_0$, $\lim_{t\to\infty}\kappa(t) = \kappa_1$, exp being the exponential function and $\lambda \in \mathbb{R}_{\geq 0}$ being the decay parameter.

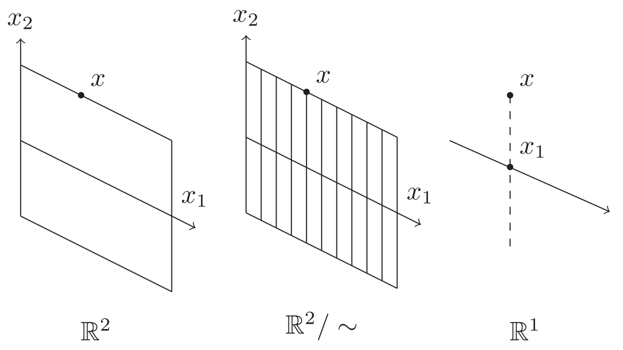

Figure 11. Quotient space example. **Left:** Space $\mathbb{R}^2$. **Middle:** Quotient space $Q = \mathbb{R}^2/\sim$, the set of equivalence classes of vertical lines. **Right:** Representative space $\mathbb{R}^1$ under representation mapping $v : Q \to \mathbb{R}^1$ (Adapted from (Orthey et al. 2018)).

C. Meta-Analysis of Primitive Methods. As discussed in Section 5, each bundle space primitive can be implemented in multiple ways. To find out which method works best for a specific algorithm, we perform a meta-analysis. In this meta-analysis, we select each bundle algorithm QRRT, QRRT*, QMP, and QMP* and vary its primitive methods. We vary those methods by taking the runtime average over the same set of environments as in Section 8 (except the hypercube). We then present the results as ratios of the best runtime. This means, to find the best sampling method for QRRT, we let QRRT run on all environments with different sampling method, then average the results for each method. We then take the method with the lowest runtime and assign it the ratio 1. All other runtimes are represented as multiples of the lowest runtime.

The results are shown in Figure 12. We divide the results into four groups. First, we compare the intrinsic metric to the quotient space (QS) metric (left group). Second, we compare the importance selection of a bundle space by comparing uniform, exponential and epsilon greedy (middle left). Third, we compare the graph sampling strategies, namely, random vertex, random edge and degree vertex (middle right). Finally, we compare the algorithms with enabled find section method and without (right).

In the case of QRRT, we observe the best metric to be the intrinsic metric (left) and that using the recursive find section method, we can lower the runtime significantly (right). However, for sampling and selection, we do not have a clear best strategy. Instead, we observe that a change in sampling or importance has a marginal influence on the performance. For the other three algorithms QRRT*, QMP, and QMP*, we observe similar results. One exception is QRRT*, where we observe the QS metric and the no find section method to perform only 1.25 times worse.

C.1. Discussion of results. The results indicate that both for sampling and importance selection, there is no clear advantage of using either strategy. This suggests that either

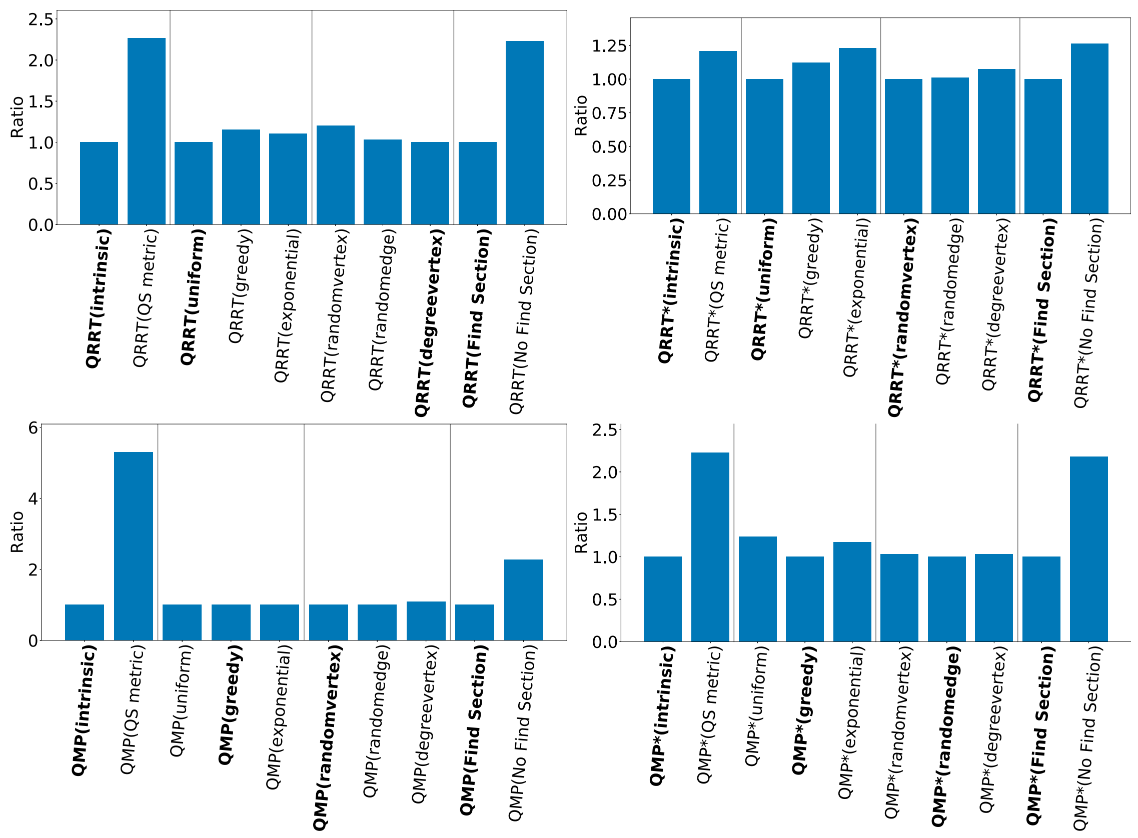

Figure 12. Meta-analysis of different implementations of bundle space primitives. Each graph shows the performance of QRRT, QRRT*, QMP, and QMP* on the high-dimensional benchmark set by comparing four different primitives, the metric (intrinsic vs quotient space), the importance selection (uniform, greedy, or exponential), the sampling strategy (random vertex, random edge, random degree vertex), and having the sidestep section method. See Section 5 for details. The results are displayed as ratio compared to the value of the best performing implementation in each category.

strategy can be chosen for the scenarios under investigation. Further investigation is required to understand the influence of sampling strategies over different types of bundle spaces.

Concerning the metric and section method, the difference in performance is significant. In detail, for all bundle planners, both the intrinsic method, and the find section method perform significantly better. The reason why the intrinsic metric is better lies in its simplicity. While the intrinsic metric can rapidly return values, the QS metric requires an expensive graph search. While the QS metric is more accurate, this is offset by its computational burden. The reason why the find section method performs better is due to independent movements of links caused by the L1 interpolation. This is often a decisive factor to ensure that colliding links are moved out of the way to clear the way towards the goal. Most of the problems in our evaluations benefit from this movement. An example is the box folding task, where moving outer links towards the goal positions increased our chances to find collision-free motions.