Motion Planning Explorer: Visualizing Local Minima Using a Local-Minima Tree

Andreas Orthey, Benjamin Frész, and Marc Toussaint

***Abstract*—Motion planning problems often have many local minima. Those minima are important to visualize to let a user guide, prevent or predict motions. Towards this goal, we develop the motion planning explorer, an algorithm to let users interactively explore a tree of local-minima. Following ideas from Morse theory, we define local minima as paths invariant under minimization of a cost functional. The local-minima are grouped into a local-minima tree using lower-dimensional projections specified by a user. The user can then interactively explore the local-minima tree, thereby visualizing the problem structure and guide or prevent motions. We show the motion planning explorer to faithfully capture local minima in four realistic scenarios, both for holonomic and certain non-holonomic robots.**

***Index Terms*—Motion and Path Planning, Nonholonomic Motion Planning, Foundations of Automation.**

I. Introduction

In MOTION planning, we develop algorithms to move robots from an initial configuration to a desired goal configuration. Such algorithms are essential for manufacturing, autonomous flight, computer animation or protein folding [17].

Most motion planning algorithms are black-box algorithms.[1] A user inputs a goal configuration and the algorithm returns a motion. In real-world scenarios, however, black-box algorithms are problematic. Human users cannot interact with the algorithm. There is no way to guide or prevent motions. Humans users cannot visualize the internal mechanism of the algorithm. There is no intuitive way to understand or debug the algorithm. Human users cannot predict the outcome of the algorithm. There is no way for coworkers to avoid or plan around a robot. Black-box algorithms are therefore an obstacle for having robots move in a safe, predictable and controllable way.

In an effort to make robotic algorithms visualizable, predictable and interactive, we develop the motion planning explorer. Using the planning explorer, we enumerate and visualize local minima. Using ideas from Morse theory [21], we define a local minimum as *a path which is invariant under minimization of a cost functional*. To each local minimum we can associate an equivalence class, the equivalence class of all paths converging to the local minimum.

Using this equivalence relation, we utilize a fiber bundle construction—a sequence of admissible lower-dimensional projections [25]—to organize the local-minima into a tree. Since the number of leaves of this tree is usually countable infinite, we do not compute the tree explicitly, but let users interactively explore the tree.

This local-minima tree is primarily a tool to visualize the problem structure. However, we believe it to be more widely applicable. The tree is a visual guide to the (topological) complexity of the problem [31]. The tree visualizes where a deformation algorithm [34] converges to. The tree allows us to interact with the algorithm, useful for factory workers guiding their robot or the control of computer avatars. The tree can be used to give high-level instructions to a robot—crucial when bandwidth is limited. The tree provides alternatives for efficient replanning [5]. Finally, the tree can be a source of symbolic representations [33].

A. Contributions

We make three original contributions

1) We propose a new data structure, the local-minima tree, to enumerate and organize local minima
2) We propose an algorithm, the motion planning explorer, which creates a local-minima tree from input by a user
3) We demonstrate the performance of the motion planning explorer on realistic planning problems and on pathological environments

Our algorithm requires, for each robot, the specification of a fiber bundle by a user. We can then handle any holonomic robotic system [17] and provide a first generalization to non-holonomic systems.

II. Related Work

We can visualize a motion planning problem by visualizing its decomposition. However, there is no clear consensus among researchers on the notion of decomposition.

Often the problem is decomposed topologically [9]. In a topological decomposition, we partition the pathspace into homotopy classes, sets of paths continuously deformable into each other [22] (See Fig. 1 Left). We can compute homotopy classes by computing an H-signature of paths [4] which counts, for each obstacle, the number of times a path crosses a line emanating from that obstacle. This can be generalized to higher

Manuscript received September 10, 2019; accepted November 23, 2019. Date of publication December 9, 2019; date of current version December 20, 2019. This letter was recommended for publication by Associate Editor D. Halperin and Editor N. Amato upon evaluation of the reviewers' comments. This work was supported by a grant from the Alexander von Humboldt Foundation. *(Corresponding author: Andreas Orthey.)*

A. Orthey and B. Frész are with the Institute of Parallel and Distributed Systems, University of Stuttgart, 70174 Stuttgart, Germany (e-mail: andreas.orthey@gmx.de; b.fresz@gmx.de).

M. Toussaint is with the Institute of Parallel and Distributed Systems, University of Stuttgart, 70174 Stuttgart, Germany, and also with the Max Planck Institute for Intelligent Systems, 70569 Stuttgart, Germany (e-mail: marc.toussaint@informatik.uni-stuttgart.de).

Digital Object Identifier 10.1109/LRA.2019.2958524

[1] We call an algorithm a black-box algorithm whenever the internal mechanism is hidden from the user [1].

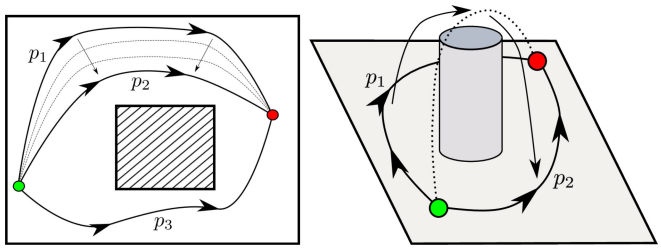


Fig. 1. **Left:** Homotopic paths in 2D. **Right:** Distinct local minima which are homotopic in 3D.

dimensions, where we measure how often a path passes through holes in configuration space [3]. If the configuration space is not too high-dimensional, we can also compute homotopy classes using simplicial complices [28] or lower-dimensional task projections [29].

Topological decompositions, however, do not adequately capture the intricate geometry of configuration space constraints and are often computationally inefficient. Many alternative definition have been proposed to obtain computationally-efficient homotopy-like decompositions. Examples include digital homotopy relations [30], K-order deformability [12] and convertibility of paths [24].

However, computationally-efficient homotopy decompositions fail to give a proper pathspace partitioning. All previously named efficient decompositions are violating the transitivity relation[2] and do not constitute an equivalence relation. This makes it difficult to have clear lines of demarcation between path subsets. It is also unclear how to visualize overlapping path sets.

We believe a more appropriate decomposition is the cost-function decomposition. In a cost-function decomposition, we group paths together whenever *they converge under optimization to the same local minimum*. With such an approach, we can leverage optimization methods for computational efficiency and compute partitions of the pathspace. In Fig. 1 (Right) we show a partition into two local-minima classes (ignoring minima wrapping around the obstacle).

The computation of local minima of cost functions belongs to the topic of optimal motion planning [14]. In optimal motion planning we like to find the global cost-function minimum. Recently, several sampling-based algorithms have been proposed which are asymptotically optimal, i.e. we will find the global optimum if time goes to infinity [14]. Recent extensions of those algorithms exploit graph sparsity [7], improve upon convergence time [13] and solve kinodynamic problems [19].

However, most optimal planning algorithms will find only the global optimal path, but not necessarily all local optimal paths. Our work differs by interactively computing local minima and arranging them into a local-minima tree. To build the tree, we require fiber bundle simplifications of the configuration space [25]. Those simplifications help us to organize the local-minima of a planning problem and visualize its path space.

Visualization of path spaces is closely related to topological data analysis and Morse theory. We briefly discuss those approaches and how they differ from our approach.

[2]Transitivity holds for a pathspace decomposition if whenever a path a is equivalent to a path b, and b is equivalent to a path c, then a is equivalent to c.

1) *Topological Data Analysis:* In topological data analysis [6], we use structures from algebraic topology (e.g. simplicial complices [8]) to model and visualize the topology of high-dimensional data. Our approach differs by computing paths directly (not modelling the topology) and by making the visualization interactive.

2) *Infinite-Dimensional Morse Theory:* The goal of infinite-dimensional Morse theory [21] is studying solution spaces of optimization problems and investigating critical solution paths [20]. Our approach can be seen as applying Morse theory to motion planning while ignoring second-order behavior of the paths.

III. Background

Let (X, ϕ) be the *planning space*, consisting of the configuration space X of a robot and the *constraint function* $\phi : X \to \{0, 1\}$ which on input $x \in X$ outputs zero when x is constraint-free and one otherwise. We extend ϕ such that on input of subsets $U \subseteq X$ outputs zero when at least one $x \in U$ is constraint-free and to one otherwise. The constraint function defines the *free configuration space* $X_{\mathrm{f}} = \{x \in X \mid \phi(x) = 0\}$. Given an initial configuration $x_I \in X_{\mathrm{f}}$ and a goal configuration $x_G \in X_{\mathrm{f}}$, we are interested in finding a path in X_{f} connecting them. We call $(X_{\mathrm{f}}, x_I, x_G)$ a *motion planning problem* [25].

The space of solutions to a planning problem is given by its *path space*. The path space P is the set of continuous paths $\mathbf{p} : I \to X_{\mathrm{f}}$ from $I = [0, 1]$ to X_{f} such that $\mathbf{p}(0) = x_I$ and $\mathbf{p}(1) = x_G$. We equip the pathspace P with a cost functional $c : P \to \mathbb{R}_{\geq 0}$ on P. Examples of cost functionals are minimum-length, minimum-energy, or maximum-clearance.

A. Admissible Fiber Bundles

We can often simplify planning spaces using *fiber bundles* [25]. A fiber bundle is a tuple (X, Y, π, π_ϕ) consisting of a mapping

$$\pi : X \to Y \tag{1}$$

which maps open sets to open sets and a mapping $\pi_\phi : \phi \to \phi_Y$, which map a planning space (X, ϕ) to a lower-dimensional space (Y, ϕ_Y). We say that π_ϕ is *admissible* if the admissibility condition $\phi_Y(y) \leq \phi(\pi^{-1}(y))$ holds for all $y \in Y$, whereby we call $\pi^{-1}(y)$ the *fiber* of y in X. We then call π an admissible lower-dimensional projection, X the bundle space, and Y the *quotient space* of X under π [18].

Often it is advantageous to define chains of K fiber bundles $(X_k, X_{k-1}, \pi_k, \pi_{\phi_k})$ with admissible mappings

$$\{\pi_k : X_k \to X_{k-1}\}_{k=1}^{K} \tag{2}$$

such that $X_K = X$ and the constraint functions are admissible such that $\phi_{k-1}(x_{k-1}) \leq \phi_k(\pi_k^{-1}(x_{k-1}))$ for all $x_{k-1} \in X_{k-1}$. Admissible fiber bundles have been shown to be a generalization of constraint relaxation, a source of admissible heuristic and can reduce planning time by up to one order of magnitude [25].

There are multiple ways of simplifying a configuration space to construct a fiber bundle. We can often construct simpler robotic system by removing constraints, through nesting lower degree-of-freedom (dof) robots [26], removing links [2] or shrinking obstacles [10].

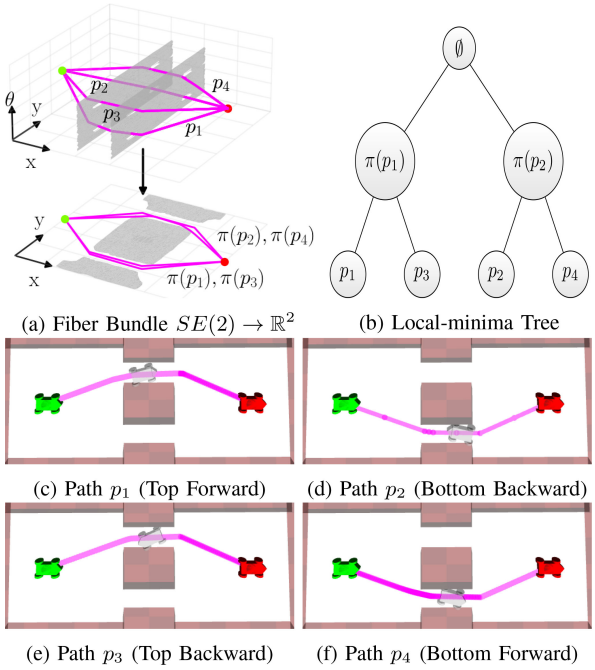


Fig. 2. Car in 2D with configuration space $SE(2)$. The planning problem can be decomposed into four parts.

If the projection mappings π and π_ϕ are obvious from the context, we will often denote the fiber bundle simply as $X \to Y$. As an example, we write $SE(2) \to \mathbb{R}^2$ for the car in Fig. 2, whereby we mean that the car has been simplified by a nested disk. The nested disk is an abstraction of the car, removing the orientation. The mapping π in that case maps position and orientation onto position, and $\phi_{\mathbb{R}^2}$ is zero whenever the disk is collision-free.

IV. Method

In this section, we describe the *local-minima tree*. First, we define local minima as paths which are invariant under minimization of a cost functional. We then associate an equivalence class to each local minimum, consisting of all paths converging to the same local minimum. Using this equivalence relation, we then construct a local-minima space. To visualize the local-minima space, we finally group local-minima into a tree using the fiber bundle construction [25].

A. Assumptions

Let (X_f, x_I, x_G) be a motion planning problem, P its path space, and $c : P \to \mathbb{R}_{\geq 0}$ be a cost functional on the pathspace. We assume that there exists a path optimization algorithm that we represent as a mapping $f_c : P \to P$, which takes any path and transforms the path into a path having a locally minimal cost. We make no further assumptions about the optimizer, such as that the output optimum is close to the initialization. Instead, our notion of path equivalence will be relative to a given f_c. Further, we let a user provide an admissible fiber bundle $X_K \to X_{K-1} \to \cdots \to X_0$ with $X_K = X$, which simplifies the configuration space X. The fiber bundle implicitly defines lower-dimensional projections $\pi_K, \ldots, \pi_1$.

B. Local-Minima Space

The minimization function f_c *partitions*[3] the pathspace. The partition is given by an equivalence relation we call path equivalence. Given the path-equivalence, we can construct the quotient of the pathspace under path-equivalence, which we call the local-minima space.

Let us start by defining path-equivalence. If two paths converge, under the optimizer f_c of the cost c, to the same path, we say they are path-equivalent. Formally, given two paths $\mathbf{p}, \mathbf{p}' \in P$, we say that they are *path-equivalent*, written as $\mathbf{p} \sim_{f_c} \mathbf{p}'$, if

$$f_c(\mathbf{p}) = f_c(\mathbf{p}') \tag{3}$$

It is straightforward to check that path-equivalence is an equivalence relation (i.e. reflexive, symmetric, transitive). The optimizer f_c therefore partitions the pathspace [22].

To better understand this partition, we construct the local-minima space as the quotient space of all equivalence classes of P under f_c, denoted as

$$Q = P/\sim_{f_c} \tag{4}$$

Elements of the space Q are equivalence classes of paths. We will, however, *represent* each equivalence class by the path which is invariant under minimization of the cost. We call those paths local minima.

To simplify matters, we will only consider *simple* local minima. A simple local minimum is a local minimum without self-intersections. Simple paths are easier to compute and often capture all important local minima in a problem. However, we note that there are certain pathological cases, where non-simple paths are required to solve the problem [26].

C. Sequential Projections of the Local-Minima Space

To efficiently represent the local-minima space, we propose to sequentially partition the space using the fiber bundle projections. This works as follows: Two distinct local minima of Q are projected onto a quotient-space X_k using the mapping π_k. We then consider them to be projection-equivalent, when, under minimization f_c, they converge to the same path.

More formally, given two local minima $\mathbf{q}, \mathbf{q}' \in Q$, we say that they are *projection-equivalent*, written as $\mathbf{q} \sim_{\{f_c, \pi_k\}} \mathbf{q}'$, if

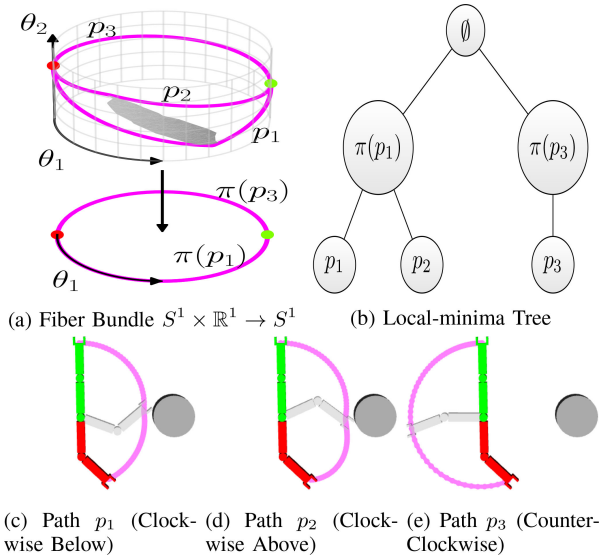
$$f_c(\pi_k(\mathbf{q})) = f_c(\pi_k(\mathbf{q}')) \tag{5}$$

Projection-equivalence is again an equivalence relation and therefore partitions the local-minima space. We denote the quotient of Q under the projection-equivalence as Q_{K-1}. We then iterate this process for each projection mapping. Thus, given an admissible fiber bundle $X_K \to X_{K-1} \to \cdots \to X_0$, we construct a sequence of local-minima spaces $Q_K, \ldots, Q_0$ with $Q_K = Q$. In other words, the local-minima space Q_{k-1} is obtained from Q_k as the quotient-space

$$Q_{k-1} = Q_k/\sim_{\{f_c, \pi_k\}} \tag{6}$$

Elements of Q_{k-1} are equivalence classes of local minima of Q_k. We will, however, *represent* each equivalence class by the path to which all its elements (after projection) will converge to.

[3]A partition of a set X is a family of disjoint non-empty sets such that every element of X is in exactly one such set.



(a) Fiber Bundle $S^1 \times \mathbb{R}^1 \to S^1$ (b) Local-minima Tree

(c) Path p_1 (Clockwise Below) (d) Path p_2 (Clockwise Above) (e) Path p_3 (Counter-Clockwise)

Fig. 3. 2D manipulator with environment which can be decomposed into three parts. Note that the three shown paths cannot be deformed into each other.

D. Local-Minima Tree

Finally, we use the sequence of local-minima spaces to construct the local-minima tree. The tree consist of all elements of $Q_0, \ldots, Q_K$ as nodes. Two nodes are connected by a directed edge, if the first node is a local minimum $\mathbf{q}_k$ of Q_k, the second node is a local minimum $\mathbf{q}_{k+1}$ of Q_{k+1}, and we have $f_c(\pi_{k+1}(\mathbf{q}_{k+1})) = \mathbf{q}_k$. Additionally, we add one empty-set root node which is connected to every element of Q_0.

Note that a complete description of the local-minima tree is only possible in trivial cases. In any real-world scenario, we can only hope to visualize certain subsets of the tree.

E. Examples

To make the preceding discussion concrete, we visualize the local-minima tree for two examples.

First, we use a free-floating 3-dof planar car with fiber bundle $SE(2) \to \mathbb{R}^2$, which represents the removal of orientation by projection onto a circular disk. The environment is shown in Fig. 2(c-f) and the fiber bundle is shown in Fig. 2(a). The planning problem is to find a path to go from the green initial configuration to the red goal configuration. We observe that there are four simple local-minima, depending on if the car is going through the top or bottom slit, and going forward or backward. The two top slit paths are projection equivalent and we group them together. The same for the bottom paths. The local-minima tree is then shown in Fig. 2(b). Note that we ignore non-simple local-minima which would occur when moving the car in a circle around the middle obstacle.

Second, we use a fixed-based 2-dof manipulator robot with fiber bundle $S^1 \times \mathbb{R}^1 \to S^1$ (S^1 is the circle space), which represents the removal of the last link. The environment is shown in Fig. 3(c-e) (obstacle in grey) with fiber bundle shown in Fig. 3(a). There are three simple local-minima, two going clockwise below (c) and above (d) the obstacle, and one going counterclockwise (e). We group them according to their projection-equivalence as counterclockwise and clockwise, respectively. The local-minima tree is shown in Fig. 3(b).

Algorithm 1: Motion Planning Explorer$(x^I, x^G, X_1, \ldots, X_K, N, t_{\max}, \delta_S, \epsilon)$.

1: $T = \emptyset$
2: $\mathbf{G}_1, \mathbf{S}_1, \ldots, \mathbf{G}_K, \mathbf{S}_K =$ InitRoadmaps$(X_1, \ldots, X_K)$
3: **while** True **do**
4: $\quad \mathbf{q}_k =$ SelectLocalMinima(T)
5: $\quad$ UpdateMinimaTree$(T, \mathbf{q}_k, \mathbf{G}_k, \mathbf{G}_{k+1}, \mathbf{S}_{k+1})$
6: **end while**
7: **return** $\emptyset$

V. Algorithm

To compute the local-minima tree, we develop the *motion planning explorer* (Algorithm 1). The motion planning explorer takes as input a planning problem $(X_{\mathrm{f}}, x_I, x_G)$, a minimization method f_c and a fiber bundle represented as a sequence of quotient spaces $X_0, \ldots, X_K$ with $X_K = X$. The explorer depends on four parameters, namely $N \in \mathbb{N}$, the maximum number of local-minima to display, $t_{\max} \in \mathbb{R}_{>0}$, the maximum time to sample in one iteration, $\delta_S \in \mathbb{R}_{>0}$, the fraction of space to be visible for the underlying sparse roadmap and $\epsilon \in \mathbb{R}_{\geq 0}$, the ϵ-neighborhood of a local minimum to sample. Given the input, we return a browsable local-minima tree T. A user can navigate this tree by clicking on local minima and by collapsing or expanding the minimum, similar to how one navigates a unix directory structure.

Our algorithm consists of an alternation of two phases. In phase one (Line 1.4), a human user can navigate the local-minima tree and select one local minima. In the beginning the user has only one choice, selecting the root node (the empty-set minimum). In the second phase (Line 1.5), the user presses a button and the algorithm uses the selected local minimum $\mathbf{q}_k$ on Q_k to find all local minima on Q_{k+1} which, when projected, would be equivalent to $\mathbf{q}_k$. For each local minimum we find, we add a directed edge from $\mathbf{q}_k$ to the local minimum. Note that we construct the local-minima tree in a top-down fashion, which differs from the bottom-up description in Sec. IV. This construction is more computationally efficient, but we might create *spurious local-minima*, which are local minima which do not have any children. In other words, there are no local-minima which, when projected, would be equivalent to the spurious local minimum. This second phase is run for a predetermined maximum timelimit $t_{\max}$, and can be run multiple times until the user has found sufficiently many local minima.

For phase one of the explorer, we develop a graphical user interface (GUI). The GUI is shown in Fig. 4, where we show the local minima tree (1), the last button pressed (2), the initial configuration (3), the goal configuration (4), and a configuration along a local minimum (environment is hidden to remove distractions). Pressing the button `left` or `right` switches local minima on the same level. Pressing `up` collapses the current local minimum and displays the local minimum on the next lower-dimensional quotient space, which is obtained by projection and subsequent optimization of the current local minimum. Pressing `down` expands the current local minimum. Pressing the button `u` executes the current local minimum path by sending it to the robot and pressing the button `w` starts the search for more local minima.

In the second phase (Algorithm 2) we update the minima tree by performing two steps. First, we take the selected local

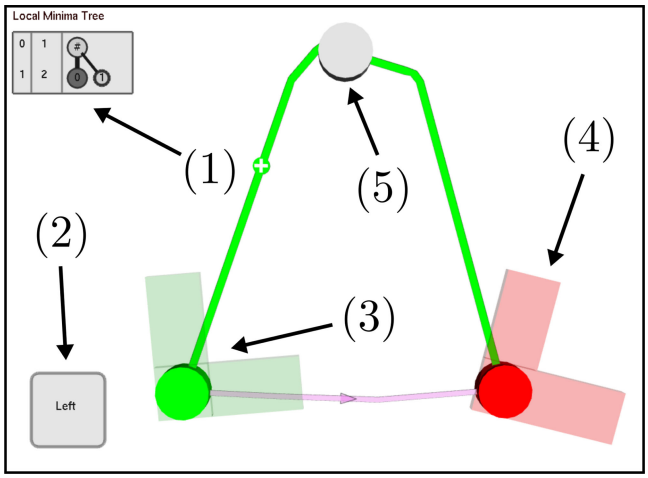


Fig. 4. **Motion Planning Explorer GUI**: (1) local-minima tree depiction. Columns show fiber bundle level, number of nodes on level and nodes of tree, respectively. (2) Last button pressed by user. (3) Initial (green) configuration on quotient-space (non-transparent disk) and on bundle space (transparent). (4) Same for goal (red) configuration. (5) Local minima selected by user and highlighted in tree.

Algorithm 2: UpdateMinimaTree($T, \mathbf{q}_k, \mathbf{G}_k, \mathbf{G}_{k+1}, \mathbf{S}_{k+1}$).

1: **while** $\neg$PTC($t_{\max}$) **do**
2: GROWROADMAP($X_{k+1}, \mathbf{q}_k, \mathbf{G}_k, \mathbf{G}_{k+1}, \mathbf{S}_{k+1}$)
3: **end while**
4: $\{q_1, \ldots, q_M\} \leftarrow$ ENUMERATEPATHS($\mathbf{S}_{k+1}$)
5: $Q_{new} \leftarrow \emptyset$
6: **for** each $\mathbf{q}$ in $\{q_1, \ldots, q_M\}$ **do**
7: **if** $\neg$ MINIMAEXISTS($\mathbf{q}, Q_{new}$) **then**
8: $Q_{new} \leftarrow Q_{new} \cup \mathbf{q}$
9: **end if**
10: **if** SIZE(Q_{new}) $\geq N$ **then**
11: **break**
12: **end if**
13: **end for**
14: ADDMINIMATOTREE(Q_{new}, T)

Algorithm 3: MinimaExists($\mathbf{q}, Q_{new}$).

1: **for** each $\mathbf{q}'$ in Q_{new} **do**
2: **if** ISVISIBLE($(\mathbf{q}, \mathbf{q}')$) **then**
3: **return** True
4: **end if**
5: **end for**
6: **return** False

minimum $\mathbf{q}_k$ on Q_k and grow a sparse graph $\mathbf{S}_{k+1}$ on the space X_{k+1} (or X_K if $k = K$) biased towards $\mathbf{q}_k$. Second, we compute up to N local-minima from the sparse graph $\mathbf{S}_{k+1}$. We first describe both steps in the case of a holonomic robot, and then describe the modifications in the non-holonomic case.

In the first step, we grow the sparse graph $\mathbf{S}_{k+1}$ on X_{k+1} for up to $t_{\max}$ seconds (or some other Planner Terminate Condition (PTC)). The algorithm `GrowRoadmap` (Line 2.2) is further detailed in Algorithm 4, which closely follows the Quotient-Space roadMap Planner (QMP) algorithm [26]. It differs from QMP by computing both a dense graph $\mathbf{G}_{k+1}$ and a sparse graph $\mathbf{S}_{k+1}$. To build the graphs, we first sample a configuration on the graph $\mathbf{G}_k$ biased towards an ϵ-neighborhood of $\mathbf{q}_k$. This configuration indexes a fiber through the inverse mapping π_k^{-1}. We then sample this fiber to obtain a configuraton on X_{k+1} (Line 4.1), compute the nearest configuration on $\mathbf{G}_{k+1}$ (Line 4.2) and connect if possible (Line 4.3). The new configuration is then conditionally added to the sparse graph (Line 4.4). Our implementation utilizes previous work from the Sparse Roadmap Spanners (SPARS) algorithm[4] [7]. The sparse graph $\mathbf{S}_{k+1}$ utilizes the parameters δ_S which determines the maximum visibility radius of a configuration. This method biases sampling towards paths which, when projected onto X_k, will be projection-equivalent to $\mathbf{q}_k$. If we find a path not projection-equivalent to $\mathbf{q}_k$, we ignore the path.

Algorithm 4: GrowRoadmap($X_{k+1}, \mathbf{q}_k, \mathbf{G}_k, \mathbf{G}_{k+1}$).

1: $x_{\text{rand}} \leftarrow$ SAMPLEFIBER($X_{k+1}, \mathbf{G}_k, \mathbf{q}_k, \epsilon$)
2: $x_{\text{near}} \leftarrow$ NEAREST($x_{\text{rand}}, \mathbf{G}_{k+1}$)
3: $x_{\text{new}} \leftarrow$ CONNECT($x_{\text{near}}, x_{\text{rand}}, \mathbf{G}_{k+1}$)
4: $\mathbf{S}_{k+1} \leftarrow$ ADDCONDITIONAL($x_{\text{near}}, \mathbf{S}_{k+1}, \delta_S$)

In the second step, we enumerate $M \leq 2N$ paths on $\mathbf{S}_{k+1}$ (Line 2.4). Those paths are found using a depth-first graph search on $\mathbf{S}_{k+1}$. For each path found, we use the optimizer f_c to let the path converge to the nearest local minimum. We then try to add this path to a set of local minima paths (Line 2.6 to 2.13). The path is added if it is not visible from any path in the set (Line 2.7). We implement the visibility function following the algorithm by [12]. Another option would be to compute a distance between two paths. However, we found this to not work well with the particular minimization method we used in the demonstrations. Note that other minimization methods might require different methods to check convergence.

In the case of a non-holonomic robot, we replace the function `GrowRoadmap` using an iteration of kinodynamic RRT [15]. We then populate the sparse graph only with the current shortest path to the goal. This allows us to find a dynamically feasible path given a geometrically feasible path on the quotient space (using the path bias through $\mathbf{q}_k$). However, for this work we did not implement an optimizer for dynamical systems and therefore can only return a single non-optimal path. In future work we need to use a sparse optimal graph spanner for kinodynamic systems like [19] and dynamical optimization functions for non-holonomic systems like [16].

Once all simple paths have been enumerated, and the local minima saved, we stop the phase, add all found local minima to the local-minima tree (Line 2.14) and display them to the user in the GUI. Then we return to phase one.

The motion planning explorer has been implemented in C++ and uses the Klampt library [11] for simulation and visualisation, and the Open Motion Planning Library (OMPL) [32] for roadmap computation and fiber bundle projection. The implementation is freely available at github.com/aorthey/MotionPlanningExplorerGUI.

[4]In particular, we add a configuration to the sparse roadmap whenever the configuration increases visibility, increases connectivity or constitutes a useful cycle.

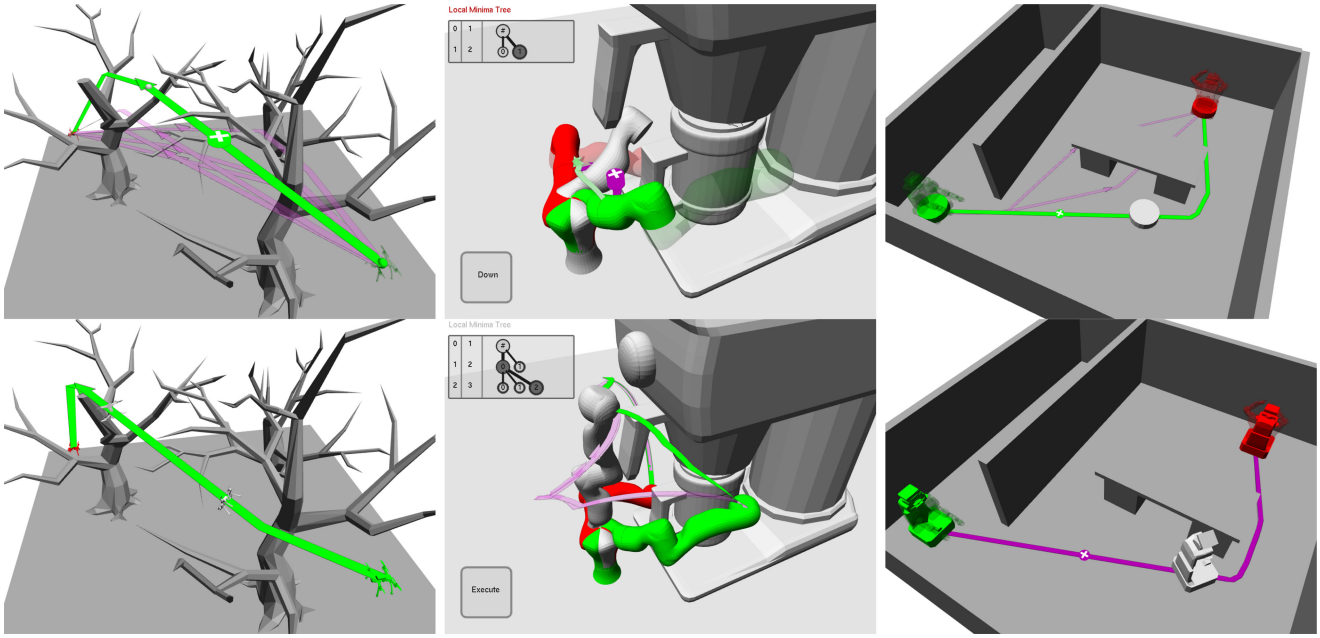


Fig. 5. **Left**: 6-dof Drone **Middle**: 7-dof KUKA LWR **Right**: 34-dof PR2.

TABLE I
TIME (S) TO GENERATE THE LOCAL-MINIMA TREE IN THE DEMONSTRATION CASES SHOWN IN VIDEO

Scenario	Time Q_0 (s)	Time $Q_{>0}$ (s)	Total Time (s)
Planar Manipulator	0.51	0.82	1.33
Planar Car	1.57	3.16	4.73
Drone in Forest	9.57	0.89	10.46
Robotic Arm	2.03	10.57	12.6
PR2	4.61	292.29	296.9
Dubin's Airplane	2.15	12.34	14.49

VI. DEMONSTRATIONS

We demonstrate the motion planning explorer on four realistic and two pathological scenarios, using the following setup. We use a minimal-length cost function and a path optimizer implemented in OMPL.[5] For each configuration space, we manually choose an admissible fiber bundle. To choose the fiber bundle, we use a heuristic considering runtime and meaningfulness of local-minima classes. In each scenario, we have used the parameters $N = 7$ (the maximum amount of visualized paths), the sparsity parameter $\delta_S = 0.1$ (the fraction of space visible from a vertex). We further set ϵ to 0.1 times the measure of the space, and we have adjusted $t_{\max}$ to be between 1s to 10 s. We perform each visualization on a 4×2.50 GHz processor laptop using 8 GB Ram and operating system Ubuntu 16.04.

We do not compare to existing methods, because we are not aware of any other algorithm which can (1) visualize local optima for any motion planning problem and (2) let a human user interact with it.

For the four scenarios we have summarized the runtimes in Table I. The runtimes show the time to compute the local minima space Q_0 (Column 1), and the time to compute the remaining local minima spaces $Q_{>0}$ (Column 2) together with their sum (Column 3). Note that those times do not include the interaction by the user, and might differ depending on which minima have been selected.

The first scenario is a drone in a forest. The configuration space is simplified using a fiber bundle $SE(3) \to \mathbb{R}^3$, corresponding to a sphere nested inside the drone. The outcome is shown in Fig. 5 (Left). The upper Figure shows seven local minima on the quotient space (magenta). Note that the quotient space is topologically trivial, but computing homotopical deformations would be computationally inefficient [12].

The user selects the green path in phase one. We then compute local minima on the configuration space which project onto this path. In this case we find one single local-minima on $SE(3)$, which we then execute.

Second, we use a robotic arm (Fig. 5 Middle) in an environment with a large coffee machine which has a visible geometric protrusion. The configuration is simplified using the fiber bundle $\mathbb{R}^7 \to \mathbb{R}^3$, obtained by removing the first three links of the robotic arm. The explorer finds two local minima on the quotient space which belong to a motion below the protrusion and above the protrusion, respectively. Finally, the user selects the path going above the protrusion, and the explorer finds three local minima which belong to different rotations of the manipulator around its axes.

Third, we use the PR2 robot in a navigation scenario. We use a fiber bundle $SE(2) \times \mathbb{R}^{31} \to SE(2) \times \mathbb{R}^7 \to \mathbb{R}^2$, which corresponds to the removal of arms, and upper torso, respectively. On the lowest-dimensional quotient space, we find three local minima (Fig. 5 Right), which correspond to going left or right around the table, and one going underneath the table. Note that the path underneath the table is spurious (see Section V). The computation of the first three local minima takes 4.61 s, while the remaining local-minima take together 292 s. This high runtime results from the high-dimensionality of the original configuration space combined with a possible narrow

[5]See ompl::geometric::PathSimplifier.

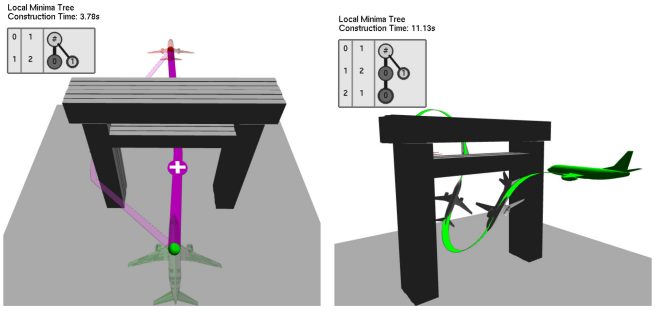


Fig. 6. Dubin's airplane with constant forward velocity of 0.5 ms^{-1} and bounds on first derivative of yaw and pitch of ± 0.1 ms^{-1}. The dynamics are modelled as a driftless airplane [27].

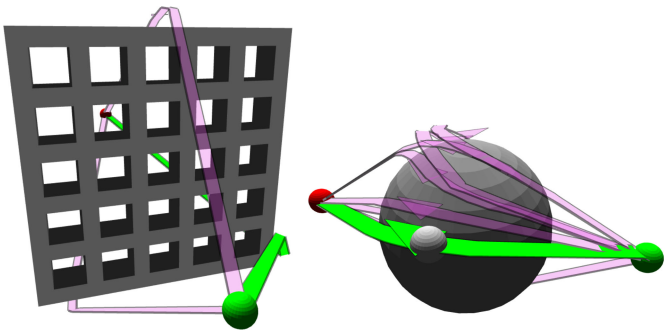


Fig. 7. Limitations: (A) low-cost small-measure local-minima are often ignored in favor of high-cost but large-measure local-minima. (B) Only a countable number of paths is found from an uncountable number of local minima.

passage occurring when the robot has to traverse the corner of the table.

In the last scenario, we visualize the flight paths of dubin's airplane [17] through an archway. Dubin's airplane is a rigid body in 3D with velocity constraints such that it flies at a constant forward velocity of $+0.5$ ms^{-1} and has bounds on the first derivative of yaw and pitch of ± 0.1 ms^{-1}. The fiber bundle is $SE(3) \times \mathbb{R}^6 \to \mathbb{R}^3$, which corresponds to the removal of dynamical constraints and orientation. We see that the algorithm finds two distinct local minima on the quotient space, which corresponds to going through or around the archway. We then select the local minimum going through the archway and the algorithm finds a dynamically feasible path (Fig. 6 Right).

This demonstrates that our method can be extended to dynamical systems, even when the shortest path of the geometrical system is neither dynamically feasible nor near to a dynamically feasible path. However, as detailed in Sec. V, we currently do not have an adequate optimization function to compute a dynamically optimal path. The resulting dynamical path is therefore non-optimal.

A. *Pathological Scenarios and Limitations*

While the motion planning explorer works well on realistic scenarios, it might not work well on pathological cases. To test this, we demonstrate the performance on two scenarios which have been crafted to break the algorithm.

In the first scenario (Fig. 7 Left), we need to move a ball with configuration space $\mathbb{R}^3$ from an initial (green) to a goal (red) configuration. Between the configurations we place a lattice with openings slightly larger than the radius of the ball. All local minima through the lattice have a neighborhood in pathspace with a vanishingly small measure. Our algorithm, however, rarely detects those minima, because the probability of finding samples inside an opening is smaller than finding samples above or below the lattice. Therefore, the algorithm usually finds minima with a higher cost going around the lattice.

In the second scenario (Fig. 7 Right), we place a spherical obstacle between the initial and goal configuration of the ball (As described by Karaman and Frazzoli [14]). The number of local minima is uncountable infinite. Our algorithm, however, can only find a finite number of local minima and is unable to describe the complete uncountable set.

VII. Conclusion

We introduced the motion planning explorer, an algorithm taking a planning problem as input and computing a local-minima tree. Local minima are defined as paths which are invariant under minimization of a cost functional. The local-minima are grouped into a tree, where two paths are grouped together if they are projection-equivalent under a lower-dimensional fiber bundle projection. We showed that the resulting local-minima tree faithfully captures the structure of holonomic and certain non-holonomic problems.

The implementation of the local-minima tree has, however, three limitations. First, we restrict computation to N simple paths, which makes the tree non-exhaustive. We could alleviate this by letting the user add additional local-minima and by enumerating non-simple paths. Second, the runtime is sometimes prohibitive for real-time application. We believe this could be addressed by specifically tailored hardware [23], code optimization and a sampling-bias towards narrow passages. Third, the construction of the tree depends on pre-specified lower-dimensional projections. We could remove this dependency by enumerating all projections [25] and use a specific projection only if it will group at least two local minima together.

Most importantly, however, the computation time spent constructing the local-minima tree is negligible compared to having a tool which allows us to visualize, debug and interact with a planning problem.

Acknowledgment

The authors would like to thank the anonymous reviewers for clarifying the connection to Morse theory. They would also like to thank M. Moll and Z. Kingston for independent code reviews of Fiber Bundle Projections and the website TurboSquid for providing three-dimensional models.

References

[1] "Black box." [Online]. Available: https://www.merriam-webster.com/dictionary/black box. Accessed on: Nov. 23, 2019.
[2] O. B. Bayazit, D. Xie, and N. M. Amato, "Iterative relaxation of constraints: A framework for improving automated motion planning," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2005, pp. 3433–3440.
[3] S. Bhattacharya and R. Ghrist, "Path homotopy invariants and their application to optimal trajectory planning," *Ann. Math. Artif. Intell.*, vol. 84, no. 3/4, pp. 139–160, 2018.
[4] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Auton. Robots*, vol. 33, no. 3, pp. 273–290, 2012.

[5] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *Int. J. Robot. Res.*, vol. 21, pp. 1031–1052, 2002.
[6] G. Carlsson, "Topology and data," *Bull. Amer. Math. Soc.*, vol. 46, no. 2, pp. 255–308, 2009.
[7] A. Dobson and K. E. Bekris, "Sparse roadmap spanners for asymptotically near-optimal motion planning," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 18–47, 2014.
[8] H. Edelsbrunner and J. Harer, *Computational Topology: An Introduction*. Providence, RI, USA: Amer. Math. Soc., 2010.
[9] M. Farber, "Topology of random linkages," *Algebr. Geometric Topology*, vol. 8, pp. 155–171, 2008.
[10] P. Ferbach and J. Barraquand, "A method of progressive constraints for manipulation planning," *Trans. Robot.*, vol. 13, no. 4, pp. 473–485, 1997.
[11] K. Hauser, "Robust contact generation for robot simulation with unstructured meshes," in *Proc. Int. J. Robot. Res.*, 2016, pp. 357–373.
[12] L. Jaillet and T. Siméon, "Path deformation roadmaps: Compact graphs with useful cycles for motion planning," *Int. J. Robot. Res.*, vol. 27, pp. 1175–1188, 2008.
[13] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *Int. J. Robot. Res.*, vol. 34, no. 7, pp. 883–921, 2015.
[14] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
[15] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, vol. 2, pp. 995–1001.
[16] F. Lamiraux, D. Bonnafous, and O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots," *IEEE Trans. Robot.*, vol. 20, no. 6, pp. 967–977, Dec. 2004.
[17] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[18] J. Lee, *Introduction to Topological Manifolds*, vol. 202. Berlin, Germany: Springer, 2010.
[19] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *Int. J. Robot. Res.*, vol. 35, pp. 528–564, 2016.
[20] J. Milnor, "Morse theory," *Ann. Math. Studies*, vol. 51, 1963.
[21] M. Morse, *The Calculus of Variations in the Large* (Colloquium Publications), vol. 18. Providence, RI, USA: Amer. Math. Soc., 1934.
[22] J. Munkres, *Topology*. London, U.K.: Pearson, 2000.
[23] S. Murray, W. Floyd-Jones, Y. Qi, D. J. Sorin, and G. Konidaris, "Robot motion planning on a chip," in *Proc. Robot.: Sci. Syst.*, 2016. [Online]. Available: http://www.roboticsproceedings.org/rss12/p04.html
[24] D. Nieuwenhuisen and M. H. Overmars, "Useful cycles in probabilistic roadmap graphs," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, vol. 1, pp. 446–452.
[25] A. Orthey and M. Toussaint, "Rapidly-exploring quotient-space trees: Motion planning using sequential simplifications," in *Proc. Int. Symp. Robot. Res.*, 2019.
[26] A. Orthey, A. Escande, and E. Yoshida, "Quotient-space motion planning," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2018, pp. 8089–8096.
[27] A. Orthey, O. Roussel, O. Stasse, and M. Taïx, "Motion planning in irreducible path spaces," *Robot. Auton. Syst.*, vol. 109, pp. 97–108, 2018.
[28] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy, "Topological trajectory classification with filtrations of simplicial complexes and persistent homology," *Int. J. Robot. Res.*, vol. 35, no. 1–3, pp. 204–223, 2016.
[29] F. T. Pokorny, D. Kragic, L. E. Kavraki, and K. Goldberg, "High-dimensional winding-augmented motion planning with 2D topological task projections and persistent homology," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 24–31.
[30] E. Schmitzberger, J.-L. Bouchet, M. Dufaut, D. Wolf, and R. Husson, "Capture of homotopy classes with probabilistic road map," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2002, vol. 3, pp. 2317–2322.
[31] S. Smale, "On the topology of algorithms, I," *J. Complexity*, vol. 3, pp. 81–89, 1987.
[32] I. A. Şucan, M. Moll, and L. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.
[33] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 4044–4051.
[34] M. Zucker *et al.*, "CHOMP: Covariant Hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, pp. 1164–1193, 2013.