Visualizing Local Minima in Multi-Robot Motion Planning using Multilevel Morse Theory

Andreas Orthey[1] and Marc Toussaint[1,2]

[1] Max Planck Institute for Intelligent Systems, Stuttgart, Germany
aorthey@is.mpg.de
[2] Technical University of Berlin, Germany

Abstract. Multi-robot motion planning problems often have many local minima. It is essential to visualize those local minima such that we can better understand, debug and interact with multi-robot systems. Towards this goal, we present the multi-robot motion explorer, an algorithm which extends previous results on multilevel Morse theory by introducing a component-based framework, where we reduce multi-robot configuration spaces by reducing each robots component space using fiber bundles. Our algorithm exploits this component structure to search for and visualize local minima. A user of the algorithm can specify a multilevel abstraction and an optimization algorithm. We use this information to incrementally build a local minima tree for a given problem. We demonstrate this algorithm on several multi-robot systems of up to 20 degrees of freedom.

Keywords: Multi-robot motion planning, Morse Theory, Fiber bundles, Visualization of Local Minima

1 Introduction

Coordinating multiple robots is essential to automate surveillance of climate changes, to collaborate on construction sites and to route autonomous vehicles. Problems of coordinating multiple robots often involve many local minima. A local minimum is a solution path to a multi-robot planning problem, such that we cannot (locally) improve upon the path, i.e. the path is *invariant under optimization of a cost functional* [19]. Existing algorithms, however, usually do not compute or visualize those local minima and often search for only a single solution [11].

To solve multi-robot motion planning problems, we argue it to be essential to visualize local minima. By visualizing local minima, we can obtain conceptual understanding about the underlying topological complexity [24], extract symbolic representations [31] and analyse the convergence of optimization algorithms [37]. By visualizing local minima, we allow interaction by non-expert users, to either guide or prevent motions [19]. By visualizing local minima, we can create high-level options [20], usable to make high-level decisions [27] or perform rapid re-planning [35].

Visualizing local minima is therefore an important requirement for multi-robot motion planning. To enumerate local minima, we develop a new algorithm we call the *multi-robot motion explorer*. The explorer extends previous works on single-robot motion planning [19]. In particular, we evoke Morse theory [16] to enumerate local minima. To each local minimum we assign an equivalence class of paths converging, under optimization, to the same local minimum. Using those equivalence classes, we use a predefined multilevel representation, described by fiber bundles [18], to organize local minima into a local-minima tree. Eventually, non-expert users can interact with this tree by navigating through it similar to navigating through a unix filesystem.

Our contributions are

1. An algorithm, we call the multi-robot motion explorer, using a component-based framework to incrementally build local minima trees for multi-robot planning problems
2. Demonstration of the multi-robot motion explorer on six multi-robot planning problems of up to 20 degrees of freedom

2   Related Work

To visualize local minima, we need to solve two associated problems. First, we need to find a representation of the configuration space. Second, we need to utilize this representation to extract local minima.

2.1   Multi-Robot Motion Planning

To represent a multi-robot motion planning problem, we can consider the robots as one generalized robot under robot-robot collision constraints in a composite configuration space [11]. In general, we can use such an approach only for a few low-dimensional robots, mainly because the problem itself is NP-hard [8]. Since the problem is NP-hard, it becomes necessary to reduce the composite configuration space. Depending on the type of robots, we can group reduction methods into two classes.

First, we have reductions for the case where all robots are equivalent (homogeneous). For homogeneous multi-robot systems, we can project all start and goal configurations into the configuration space of the first robot and find a graph connecting all configurations. To coordinate the motion of the robots along that graph we need to solve the pebbles-on-a-graph problem, which we can solve efficiently [10], either by converting it to an integer linear program [36], by partitioning the graph into regions densely or sparsely connected [22] or by utilizing simple push and swap strategies [13]. By utilizing solvers for pebbles-on-a-graph, we can for example create a larger framework to compute motions for swarms of drones [7].

Second, we have reductions for the case where robots are not equivalent (non-homogeneous). For non-homogeneous multi-robot systems, we can first compute

graphs on each component configuration space and then merge them into a graph on the composite configuration space [11]. To merge graphs, we can either use path coordination or graph coordination.

In path coordination [23], we compute paths for each robot separately. We then coordinate the execution of those paths, either by searching over the space of reparameterizations [23] or by prioritizing the robots [5].

In graph coordination, we compute graphs for each robot separately. We then combine the graphs into an (implicit) composite configuration space graph [30]. To compute this graph, we can use two methods.

First, we can create the *tensor product* of graphs, whereby all edges are combined [26]. To explore the tensor product, we can either use a random search utilizing a direction-oracle [26] or we can execute shortest paths optimistically until conflicts arise. When conflicts arise, we can expand locally the dimensionality to resolve conflicts [33]. We can combine both methods using prioritization of robots, for example by analyzing possible start and goal conflicts [32] or by the number of topologically varying paths a robot can execute [34].

Second, we can create the *cartesian product* of graphs, whereby only edges are used where at most one robot moves. We can think of the cartesian product as an approximation to the tensor product, in the sense that every multi-robot path can be arbitrarily close approximated by a path where at most one robot moves at a time [30]. However, if the underlying graphs are not dense enough, we can miss valid paths and thereby sacrifice completeness.

It is important to note that non-homogenous and homogenous robots are not mutually exclusive, but we can often further simplify non-homogenous robot problems by decomposing them into problems of groups of homogenous robots [25].

2.2   Multi-Path Multi-Robot Motion Planning

From a given representation of the composite configuration space, we like to extract local minima. Local minima can often be defined as representative paths of equivalence classes, where we define an equivalence relation on the path space for the purpose of grouping paths. Grouping paths can be done using different approaches, of which we discuss three fundamental ones.

First, we can group paths topologically [17]. In a topologically grouping, we use the notion of homotopy to define two paths to be equivalent if they can be continuously deformed into each other. To find paths which differ homotopically, we can compute a simplicial complex of the configuration space and extract paths [21] or by computing an H-score determining the number of times a path crosses subsets of the configuration space [2].

Second, we can group paths based on braid patterns [1]. In a braid pattern grouping, we define two paths to be equivalent if pairwise robot crossings are equivalent. By ignoring the type of crossing, we obtain a permutation group of robots. Using this permutation group, we can compute representative paths of varying braid pattern [14]. We can alternatively find paths of varying braid

pattern by planning a minimal-cost path constrained to a pattern [15] or by following a braid pattern controller, for example with safety separations [3].

Third, we can group paths based on Morse theory [16]. In morse theory, we define two paths to be equivalent if they converge, under optimization, to the same local minimum [19]. This differs from braid theory and topology (1) by being finer in the sense that two equivalent paths under braid theory or topology can converge to two different minima, (2) by being defined relative to an optimizer and (3) by being often faster to compute in higher dimensions [9]. In previous work we used Morse theory in single-robot motion planning [19]. In this work, we generalize this approach to multi-robot motion planning.

3   Foundations

Let $r_1, \ldots, r_M$ be M robots with associated *component* configuration spaces $Y_1, \ldots, Y_M$ of dimensionality $n_1, \ldots, n_M$. We define the (composite) configuration space $X = Y_1 \times \cdots \times Y_M$ of dimensionality $n = n_1 + \cdots + n_M$. The space of constraint-free configurations is denoted as X_f. The motion planning problem (X_f, x_I, x_G) asks us to find a path from a start configuration $x_I \in X_f$ to a goal configuration $x_G \in X_f$.

The space of solutions to the motion planning problem is given by the associated path space. The path space P_f is the space of all continuous paths $p : I \to X_f$ on X_f starting at x_I and ending at x_G. To analyse P_f, we enumerate local minima using Morse theory and use multilevel abstractions represented by fiber bundles to organize local minima into a local minima tree.

3.1   Morse Theory

We utilize Morse theory [16] to identify local minima. A local minimum is an invariant of the path space P_f under optimization of a cost functional. A cost functional J maps a path $p \in P_f$ to a real number $\mathbb{R}$ as

$$J[p] = \int_0^1 L(x, p(x), p'(x))dx \tag{1}$$

whereby L is a loss term. To solve Eq. 1, we can take one of two views. In the first view, we interpret Eq. (1) as a problem of optimal control or calculus of variation in the small [6], where we like to find *one* global minimal solution. In the second view, however, we interpret Eq. (1) as a problem of Morse theory or calculus of variation in the large [16], where we like to find *all* local minimal solutions.

In this paper, we adopt the Morse theoretic view to enumerate local minima. We define a local minimum as an invariant of Eq. (1) under optimization. To optimize, we use a local optimizer $\Phi_J : P_f \to P_f$ which we assume to be given. We require Φ_J to be different from an identity mapping, but make no other assumption about its behavior.

Following Morse theory, we interpret the optimizer Φ_J as an equivalence relation[3] on the path space [19]. In particular, given two paths $p, p' \in P_f$ we define them to be equivalent, written as $p \sim_{\Phi_J} p'$, if $\Phi_J(p) = \Phi_J(p')$. We then take the quotient $Q = P_f / \sim_{\Phi_J}$ which represents equivalence classes of paths under optimization. Each equivalence class will be *represented* by one path invariant under optimization, i.e. a *local minimum* $p^\star$ for which we have $\Phi_J(p^\star) = p^\star$. Under this representation, we associate to every X_f its local-minima space Q containing all local-minima of P_f.

3.2   Multilevel Abstractions using Fiber Bundles

Finding and interpreting local minima is often too difficult in high-dimensional configuration spaces. To simplify those spaces, we use multiple levels of abstractions which we model using the language of fiber bundles [28]. Fiber bundles are sets of admissible lower-dimensional projections we use to decrease planning time [18] and to group local minima into more meaningful classes [19].

Formally, a fiber bundle is a tuple (X, F, B) with a projection mapping

$$\pi : X \longrightarrow B \qquad (2)$$

whereby X is the bundle space and B the base space. Both bundle and base space have associated constraint-free subspaces X_f and B_f.

We impose three restriction on the fiber bundle. First, the preimage $\pi^{-1}(x_B)$ (called the fiber over x_B) of an element $x_B \in B$ is required to be isomorphic to the fiber space F. Second, the bundle space X needs to be (locally) a product space $B \times F$ [12]. Third, we require the projection map to be admissible, meaning that the projection of X_f is a subset of B_f [18].

Fig. 1: Fiber bundle $D^2 \times \mathbb{R}^1 \to D^2$

With the last requirement we ensure that no local minima are removed after a projection [19].

We will often abbreviate a fiber bundle using the shorthand $X \xrightarrow{\pi} B$. As an example, we visualize in Fig. 1 the fiber bundle $D^2 \times \mathbb{R}^1 \to D^2$ whereby D^2 is the 2-d disk. The fiber space in this case is $\mathbb{R}^1$ which is isomorphic to the preimage $\pi^{-1}(x_B)$ of a point x_B in D^2. To get a better understanding of fiber bundles, we like to think of the preimages as equivalence classes of X which are collapsed during projection onto the (quotient) base space B.
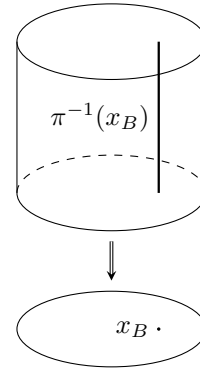
---

[3] Recall that an equivalence relation $\sim$ on a path space P is a binary relation such that for any paths $a, b, c \in P$ we have $a \sim a$ (Reflexive), if $a \sim b$ then $b \sim a$ (Symmetric) and if $a \sim b$ and $b \sim c$ then $a \sim c$ (Transitive) [17].

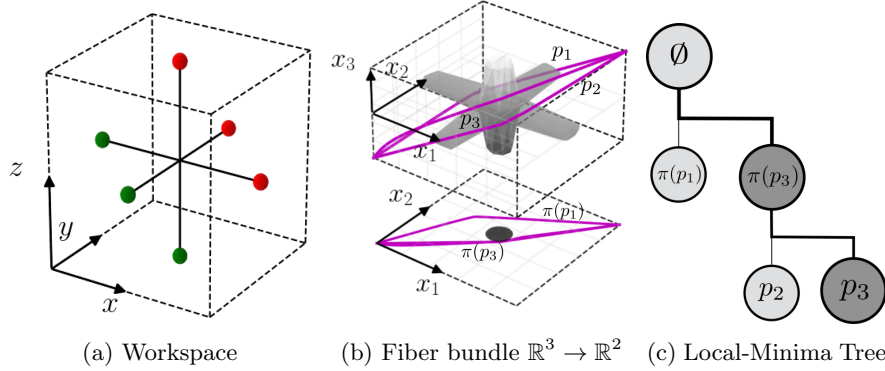(a) Workspace   (b) Fiber bundle $\mathbb{R}^3 \to \mathbb{R}^2$ (c) Local-Minima Tree

Fig. 2: Local Minima Tree for three spherical robots on line segments. See text for clarification.

A fiber bundle represents a single level of abstraction. However, we often like to represent configuration spaces on multiple levels of abstraction. In those cases, we use a *fiber bundle sequence*. A fiber bundle sequence $X_K \overset{\pi_{K-1}}{\longrightarrow} X_{K-1} \overset{\pi_{K-2}}{\longrightarrow} \cdots \overset{\pi_1}{\longrightarrow} X_1$ with $X_K = X$ consists of K bundle spaces and $K-1$ projection mappings, whereby each mapping adheres to the aforementioned requirements.

In a motion planning problem, we use fiber bundle sequences to describe simplifications, either by removing links from a robot, shrinking links, removing robots or by nesting simpler robots with less degrees-of-freedom.

3.3   Local-Minima Tree for Multilevel Morse theory

By combining Morse theory and fiber bundles, we can organize local minima into a local minima tree. Let $X_K \overset{\pi_{K-1}}{\longrightarrow} X_{K-1} \overset{\pi_{K-2}}{\longrightarrow} \cdots \overset{\pi_1}{\longrightarrow} X_1$ be a fiber bundle sequence and let Q_K be the space of local minima associated to $X_K = X$ (Sec. 3.1).

Using the projection mappings, we can reduce the space Q_K by introducing the notion of projection-equivalence. Two minima $q, q' \in Q_K$ are said to be projection-equivalent, written as $q \sim_{\{\Phi_J, \pi\}} q'$, if $\Phi_J(\pi_K(q)) = \Phi_J(\pi_K(q'))$. From this equivalence relation, we can create the quotient space $Q_{K-1} = Q_K / \sim_{\{\Phi_J, \pi\}}$. We can then iteratively apply projection-equivalence using $\pi_{K-1}, \pi_{K-2}, \ldots$ to obtain a sequence of local-minima spaces $Q_K, Q_{K-1}, \ldots, Q_1$ [19].

Finally, we can organize the local minima spaces into a local-minima tree. The local minima tree $T = (V, E)$ consists of all elements of the local-minima spaces as vertices V and of a set of directed edges E. An edge exists between two vertices v and v', if v' is an element of X_{k+1}, v an element of X_k and if v is equivalent to v' after projection and subsequent optimization, i.e. $v = \Phi_J(\pi_k(v'))$. We additionally add a root vertex which has a directed edge to every element of Q_1.

As an example, we visualize a local minima tree for a 3-dof multi-robot problem in Fig. 2. The workspace is depicted in Fig. 2a, where we show three

spherical robots at a start (green) and a goal configuration (red). The robots are allowed to move exclusively on the line segments between start and goal. The resulting configuration space is a 3d cube and we impose upon it a fiber bundle $\mathbb{R}^3 \xrightarrow{\pi} \mathbb{R}^2$, which corresponds to the removal of the third sphere (Fig. 2b). Inside the cube, we have infeasible regions where the spheres are in collision (grey areas). On the bundle space we visualize three local minima paths (magenta). The two paths p_2 and p_3 are projection-equivalent when projected onto $\mathbb{R}^2$. This gives a local minima tree, we depict in Fig. 2c. The tree shows a particular selection of path p_3 by a user (dark grey nodes), while we collapse the unselected node p_1 for visualization purposes.

4 Multi-Robot Motion Explorer

The multi-robot motion explorer is an algorithm we use to incrementally build a local minima tree for multi-robot planning problems. A user of the algorithm has to specify as input a planning problem, a fiber bundle and a path optimization algorithm. To represent the problem, we grow sequentially and simultaneously sparse roadmaps on each bundle space. At each iteration of the algorithm, a user can specify a local minimum to expand. We use this minimum to grow one of the sparse roadmaps, update the local minima tree and visualize the tree to the user. The algorithm differs from previous work [19] by using a component-based fiber bundle method which allows its usage for multi-robot planning problems.

We show the details of the multi-robot motion explorer in Alg. 1. As input, we are given a planning problem (X_f, x_I, x_G), a prespecified fiber bundle $X_K \xrightarrow{\pi_K} X_{K-1} \xrightarrow{\pi_{K-1}} \cdots \xrightarrow{\pi_1} X_0$ and an optimizer Φ_J which minimizes a cost functional J. Additional inputs are N, the maximum number of returned minima per iteration, t_b, the time budget for one iteration, δ_S, the visibility radius used to create a sparse graph, ϵ, a sampling perturbation parameter and ρ a trade-off between graph sampling and sampling towards a selected local minimum.

Before starting the algorithm, we initialize each bundle space X_k by associating with it a dense graph $\mathbf{G}_k$, a sparse graph $\mathbf{S}_k$ and a fiber space $F_k = X_{k+1}/X_k$. Extending previous work, we use a new component-based fiber space method computing $F_m^k \leftarrow Y_m^{k+1}/Y_m^k$ for each robot m in $[1, M]$ whereby Y_m^k is the m-th component space of bundle space X_k. This component-based formulation allows us to define empty fiber subspaces ($Y_m^k = Y_m^{k+1}$) and trivial fiber subspaces ($Y_m^k = \emptyset$), which correspond to an empty set projection and an identity projection, respectively.

Once all datastructures are initialized, we enter a while loop and ask the user to choose a local minimum (Line 3). In the first iteration, we let the user automatically choose the empty set local minimum (corresponding to the root node of the local minima tree). Once a local minimum q_k on the bundle space X_k has been selected, we then grow a roadmap on the next bundle space X_{k+1} while a planner terminate condition (PTC) based on a time budget t_b is false. The grow function is similar to one iteration of the sparse roadmap planner

---

Algorithm 1 MultiRobotMotionExplorer($x^I, x^G, X_{1:K}, \Phi_J, N, t_b, \delta_S, \epsilon, \rho$)

1: $T = \emptyset$ ▷ Local Minima Tree
2: **while** True **do**
3:     $q_k = $ USERSELECTLOCALMINIMUM(T)
4:     **while** ¬PTC(t_b) **do**
5:         GROWROADMAP(X_{k+1}, q_k)
6:     **end while**
7:     REMOVEREDUCIBLEFACES($\mathbf{S}_{k+1}$)
8:     UPDATELOCALMINIMATREE(T, Φ_J, N)
9: **end while**

Algorithm 2 GrowRoadmap(X_{k+1}, q_k)

1: $x_{\text{rand}} \leftarrow$ COMPONENTRESTRICTIONSAMPLING($X_{k+1}, \mathbf{G}_k, q_k, \epsilon$)
2: $\mathbf{G}_{k+1} \leftarrow$ ADDDENSEGRAPH($x_{\text{rand}}, \mathbf{G}_{k+1}$)
3: $\mathbf{S}_{k+1} \leftarrow$ ADDSPARSEGRAPH($x_{\text{rand}}, \mathbf{S}_{k+1}, \delta_S$)

Algorithm 3 ComponentRestrictionSampling($\mathbf{G}_k, q_k, X_{k+1}$)

1: **if** $k > 1$ **then**
2:     $x_{X_k}^{1:M} \leftarrow$ SAMPLEBASE($\mathbf{G}_k, q_k, \epsilon, \rho, X_k$)
3:     $x_{F_k}^{1:M} \leftarrow$ SAMPLE($x_{X_k}^{1:M}, F_k$)
4:     **for** each m in $[1, M]$ **do**
5:         $x_{k+1}^m \leftarrow$ LIFT($x_{X_k}^m, x_{F_k}^m, Y_{k+1}^m$)
6:     **end for**
7: **else**
8:     $x_{k+1}^{1:M} \leftarrow$ SAMPLE(X_{k+1})
9: **end if**
10: **return** $x_{k+1}^{1:M}$

---

[4], with the difference that we use a component restriction sampling procedure instead of uniform sampling over the configuration space.

To sample configurations we use the component restriction sampling method, as depicted in Alg. 3. We use this method to compute a biased sample on the bundle space X_{k+1}, such that if projected onto X_k, it will be close to the chosen local minimum q_k. If k is equal to 1, we do not have a chosen local minimum, and we sample uniformly on X_1 by uniformly sampling each component (Line 8). If k is larger than 1, we assume a local minimum q_k on X_k is given. We then sample the base space X_k by sampling from the graph $\mathbf{G}_k$ biased towards q_k with bias parameter ρ (Line 2). We then perturbate the sample in an ϵ-neighborhood of q_k, which helps overcoming narrow passages.

Once a sample on the base space has been computed, we uniformly sample each fiber component spaces (Line 3). We then use the fiber samples to lift each base space component sample into the bundle space X_{k+1} (Line 5). The lift method depends on the type of component mapping. Currently, we support component mappings of the form $SE(w) \rightarrow \mathbb{R}^w$, $X \times \mathbb{R}^N \rightarrow \emptyset$ and $X \times \mathbb{R}^N \rightarrow$

$X \times \mathbb{R}^M$ with $X = \{\emptyset, SE(w), SO(w)\}$, $w = \{2, 3\}$, $0 \leq M \leq N$, SE being the special euclidean and SO the special orthogonal group, respectively.

Once the grow method terminates, we update the local minima tree (Line 8). To update the tree, we enumerate N shortest paths on the sparse graph $\mathbf{S}_{k+1}$ using a depth-first search method and let those paths converge to a local minimum using the optimizer Φ_J. We then add the local minimum to the local minima tree, if the new minimum is not straight-line deformable [9] into an existing minimum.

Since optimizing paths is costly, in particular for multiple robots, we do a clean up operation before updating the tree (Line 7). In this method we iterate over all edges in the current sparse graph. For each edge with source vertex v_S and target vertex v_T, we compute common neighbors v_N of v_S and v_T and we check if the triangle v_S, v_N, v_T is feasible. This operation is done by checking if the two paths v_S to v_T and v_S to v_N to v_T are straight-line deformable [9], in which case we remove the vertex v_N from the sparse graph.

After cleaning up the sparse graph and updating the local minima tree, we return the tree and visualize it to the user.

We provide an implementation of the multi-robot motion explorer in C++ as an extension of the Open Motion Planning Library (OMPL) [29]. We additionally provide a graphical user interface (GUI) to visualize the local minima tree and to let users specify fiber bundle sequences. The code is freely available[4].

5   Demonstrations

To show the applicability of the multi-robot motion explorer, we demonstrate it on a variety of multi-robot systems. We execute all demonstrations on a laptop with a four-core 2.5GHz processor, 8GB Ram running Ubuntu 16.04. We use a minimal-length cost functional, a path optimizer provided by OMPL [29] and we define the parameters $N = 5, \delta_S = 0.15\mu, \rho = 0.05\mu$ and $\epsilon = 1 \times 10^{-3}\mu$ whereby μ is the measure of the corresponding composite configuration space.

Remark on Demonstrations For each demonstration, we visualize local minima which move the robots from an initial configuration in green to a goal configuration in red[5]. Using the time budget t_b, we run one iteration of the algorithm and report on the time and number of local minima found. Note that the times reported are rough estimates depending on the underlying sampling process and the chosen local minima. For each demonstration, we chose a fiber bundle sequence, which we visualize as a diagram in Fig. 3. Each diagram consists of the bundle space on the bottom, and the specification of the base and fiber spaces (Fig. 3a). We pick each fiber bundle based on faster computation time and leading to more meaningful local minima for users of the system.

---

[4] github.com/aorthey/MotionExplorer
[5] If printed in greyscale, initial configuration is in lightgrey, goal configuration in darkgrey and robot during execution in white.

(a) Template   (b) Two Drones   (c) Three Disks (Bhattacharya Square)

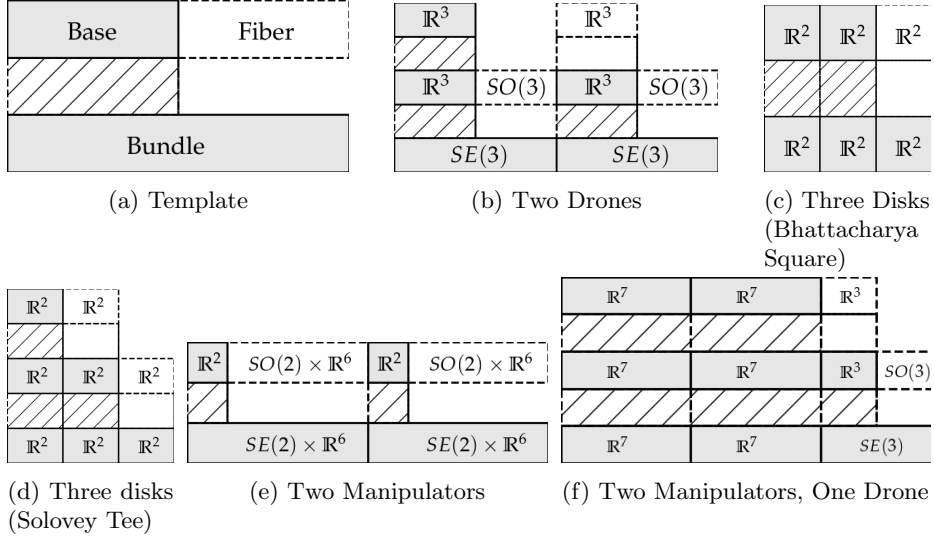(d) Three disks (Solovey Tee)   (e) Two Manipulators   (f) Two Manipulators, One Drone

Fig. 3: Fiber bundles reductions represented by fiber bundle diagrams. *Gray rectangles*: Base or Bundle spaces, *White and dashed rectangles*: Fiber spaces, *Hatch patterned rectangles*: Projections from Bundle to Base space.

Crossing Disks (2-dof) The crossing disk problem involves two disks with equivalent radius labeled 1 and 2, which can each move on a line segment orthogonal to each other (Fig. 4). The configuration space is a 2-d square with an infeasible circular region caused by configurations where both disks collide (Fig. 4a). Using our algorithm, we find two local minima after 0.47s ($t_b = 0.1$s), which we label p_1 and p_2, respectively. When choosing minimum p_1, disk 1 goes first and disk 2 follows (Fig. 4b). On minimum p_2, disk 2 goes first and disk 1 follows (Fig. 4c). The local minima tree is shown on the top left (see Sec. 3.3 for details).

Solovey Tee (6-dof) We next visualize local minima for three disks in a tee, a scenario proposed by Solovey et al. [26] (Fig. 5a). We can simplify the configuration space by removing disks. We note that the sequence of removal is important for planning time. To see this, we use two different fiber bundles, one where we first remove disk 3 and then remove disk 2 (321) and one where we first remove disk 1 and then remove disk 2 (123). For fiber bundle (123), we find one local minimum each in 0.21s, 0.30s and 3.07s ($t_b = 0.2$s), respectively (Fig. 5b). On the minimum, disk 3 goes straight towards the goal, while disk 1 and 2 clear the path by moving into the aisle. For fiber bundle (321), however, we find two local minima requiring 0.22s, 0.43s and 25.72s, respectively (Fig. 5c). Both local minima are similar in that disks 1 and 2 first move towards the goal, let disk 3 pass into the aisle, then move backwards to let disk 3 pass towards the goal.
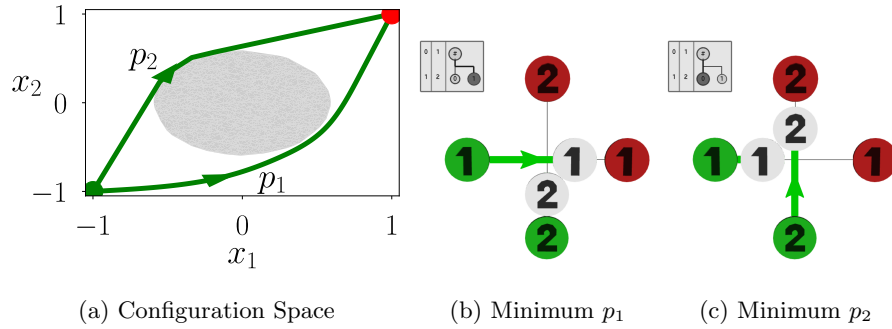
(a) Configuration Space   (b) Minimum p_1 (c) Minimum p_2

Fig. 4: Visualizing two local minima for the crossing of two disks.

(a) Initial and goal configu-
ration

(b) Local minimum using
fiber bundle (123)

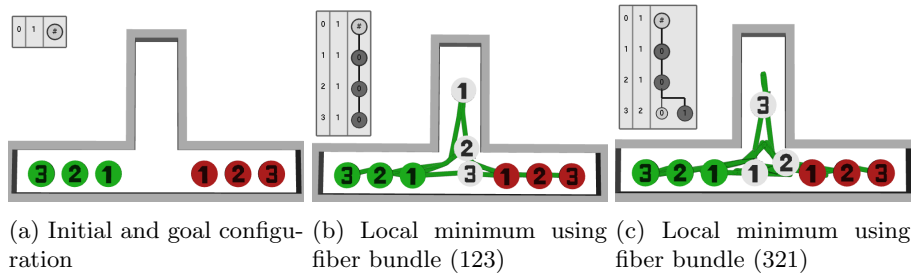(c) Local minimum using
fiber bundle (321)

Fig. 5: Three disks in a tee (Solovey et al. [26])

Drones on a Tree (12-dof) We next visualize local minima for two drones
flying around a tree (Fig. 6a). The fiber bundle reduction is given in Fig. 3b.
We find five local minima for the first drone reduction in 5.5s ($t_b = 0.1$s). We
then select the path going left around the tree, find three minima in 2.05s for
the second drone (Fig. 6b) and finally compute a valid local minimum in 0.19s
where both drones fly left around the tree (Fig. 6c).

Two-Arm Manipulator Baxter (14-dof) We next visualize local minima
for the two-arm Baxter robot. We consider each arm as a separate fixed-base
manipulator of 7-dofs. The composite configuration space has 14 dimensions. We
consider a problem where Baxter has both arms in front of its torso with the left
arm on top (Fig.7a). The goal is to change the position of the arms, such that
the right arm is on top. We find two local minima in 13.64s ($t_b = 10$s) planning
time. On the first local minimum, the left arm is moved backward and down
(Fig.7b), on the second local minimum, the left arm is moved forward and down
(Fig.7c).

Manipulators Crossing (18-dof) We next visualize local minima for two
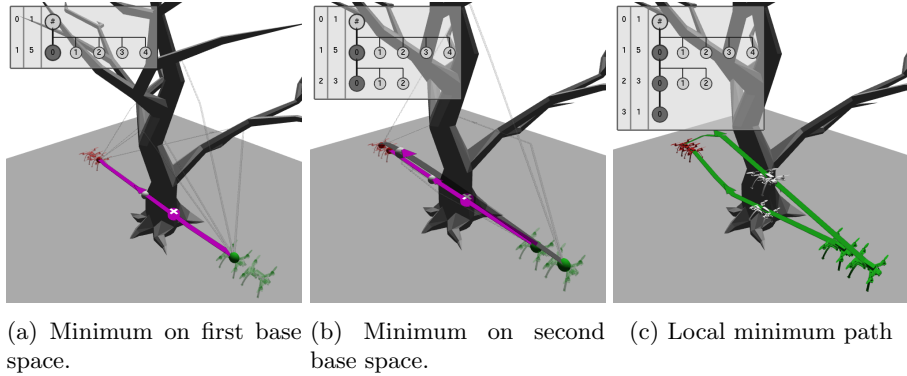mobile manipulators which need to cross each other to reach their goal (Fig.

(a) Minimum on first base space.
(b) Minimum on second base space.
(c) Local minimum path

Fig. 6: Two drones flying around a tree.

(a) Start configuration
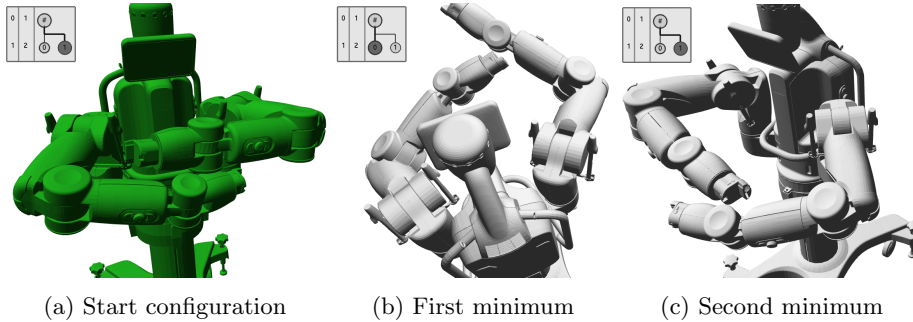(b) First minimum
(c) Second minimum

Fig. 7: Visualizing local minima for 14-dof baxter robot.

8). The composite configuration space is 18 dimensional. We use a reduction onto the base of the robots which is equivalent to two disks crossing. After planning for 0.57s ($t_b = 0.3$s) we find two local minima corresponding to the left manipulator going first or the right manipulator going first (Fig. 8a). Choosing the right manipulator to go first, we then compute three local minima in 2.38s on the composite configuration space, which correspond to different rotations of the arms (Fig. 8b and 8c).

Drone Crossing Manipulators (20-dof) We next visualize local minima for a drone crossing through two fixed-base manipulator arms which have to change places (Fig. 9a). The composite configuration space has 20 dimensions. The fiber bundle reduction is shown in Fig. 3f. On the lowest dimensional base space (14 dimensions), we compute 5 local minima in 8.96s ($t_b = 1$s), whereby two minima correspond to the forward/backward motions as in the Baxer demonstration. The other minima are variations of those but with additional rotations of the joints. We then use the local minimum where the right manipulator passes behind the left manipulator to compute in 2.15s two local minima for the inscribed sphere of
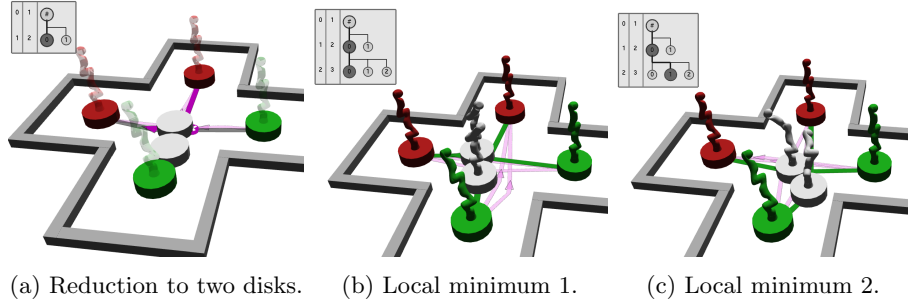
(a) Reduction to two disks.   (b) Local minimum 1.   (c) Local minimum 2.

Fig. 8: Two manipulators navigating a crossing.
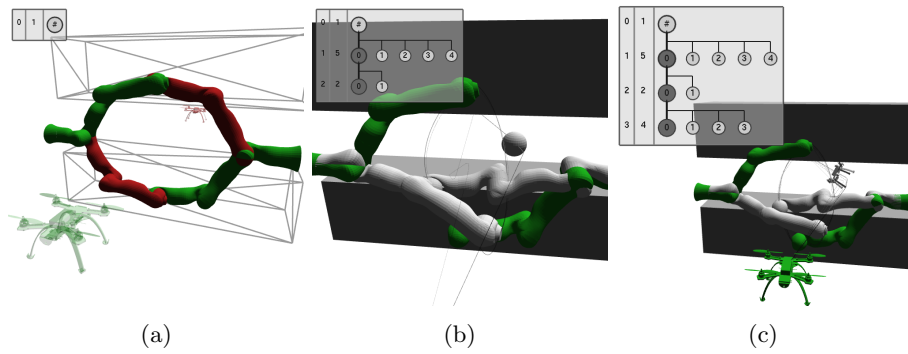

(a)   (b)   (c)

Fig. 9: Visualizing local minima for a drone flying through two fixed-base manipulators.

the drone, one going above (Fig. 9b), one going below the right manipulator. We use the local minima going above the right manipulator to obtain four minima in 26.36s on the bundle space (Fig. 9c). Those minima correspond to different rotations of the drone when flying above the manipulator.

Bhattacharya Square (6-dof) This scenario involves three labeled disks on a unit square [2] (Fig. 10). Following the discussion by Bhattacharya and Ghrist [2], this scenario involves at least eight non-winding homotopy classes. We can obtain a meaningful grouping by removing the third disk. In that case we have two local minima depending on if disk 1 goes first or disk 2 goes first. Our algorithm finds both minima in 2.65s ($t_b = 0.3$s). As can be seen in Fig. 10a, we find four local minima, two being slight variations of the desired local minima. This can be due to premature convergence of the optimizer or intricate geometric features in 4-d space. We then select the minimum where disk 2 goes first and find in 6.15s three local minima on the bundle space. By inspection, we know that there should be four minima depending on if disk 3 goes before or after disk 1 and before or after disk 2. However, we only find the minimum where disk 3

(a) Minimum on Base Space (b) First minimum on Bun- (c) Second minimum on
                          dle Space   Bundle Space
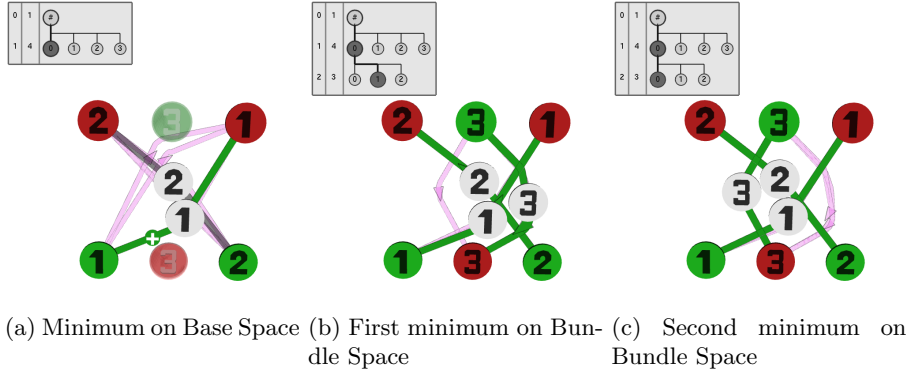
Fig. 10: Three labeled disks on a square (Bhattacharya and Ghrist [2]).

goes before disk 1 and before disk 2 (Fig. 10b) and the minimum where disk 3
goes before disk 2 and after disk 1 (Fig. 10c). We do not find the other local
minima, most likely because they belong to narrow passages in the configuration
space. We occasionally observe the algorithm to find local minima with cycling
behavior, where two robots meet, cycle around each other and then continue
onward. We discuss possible solutions to those problems in Sec. 6.

6   Conclusion

To visualize local minima, we developed the multi-robot motion explorer. Using
the explorer, we extended previous results on single-robot visualization [19] by
using a component-based Multilevel Morse theory framework. In demonstrations,
we showed the motion explorer to be applicable to several multi-robot scenarios
involving disks, drones, and manipulator arms.

While the algorithm works robustly on many robot platforms, we observed
three limitations. First, we often missed local minima when the configuration
space contained narrow passages. We could alleviate this problem by biasing
sampling towards narrow passages, by analyzing locally reachable sets [27] or by
targeted sampling of undiscovered braid pattern [3]. Second, the algorithm can
return minima with cycling behavior, where two robots cycle around each other
before continuing. We could alleviate this problem by detecting and removing
cycles or by penalizing cycle paths using additional cost functionals. Third, we
rely on manually specified fiber bundle reductions. To automate this, we could
specify a set of elementary planning problems and search for one which best
reduces the problem at hand.

Despite limitations, by visualizing local minima, we have contributed a useful
algorithm to the multi-robot planning toolbox. Using this algorithm, we can
increase our conceptual understanding to better debug, reduce and interact with
multi-robot motion planning problems.

Bibliography

[1] E. Artin, "Theory of braids," *Annals of Mathematics*, pp. 101–126, 1947.

[2] S. Bhattacharya and R. Ghrist, "Path homotopy invariants and their application to optimal trajectory planning," *Annals of Mathematics and Artificial Intelligence*, vol. 84, no. 3-4, pp. 139–160, 2018.

[3] Y. Diaz-Mercado and M. Egerstedt, "Multirobot mixing via braid groups," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1375–1385, 2017.

[4] A. Dobson and K. E. Bekris, "Sparse roadmap spanners for asymptotically near-optimal motion planning," *International Journal of Robotics Research*, vol. 33, no. 1, pp. 18–47, 2014.

[5] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1-4, p. 477, 1987.

[6] I. M. Gelfand, R. A. Silverman *et al.*, *Calculus of variations*. Courier Corporation, 2000.

[7] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[8] J. E. Hopcroft, J. T. Schwartz, and M. Sharir, "On the complexity of motion planning for multiple independent objects; pspace-hardness of the" warehouseman's problem"," *International Journal of Robotics Research*, vol. 3, no. 4, pp. 76–88, 1984.

[9] L. Jaillet and T. Siméon, "Path deformation roadmaps: Compact graphs with useful cycles for motion planning," *International Journal of Robotics Research*, 2008.

[10] D. Kornhauser, G. L. Miller, and P. Spirakis, "Coordinating pebble motion on graphs, the diameter of permutation groups, and applications," in *Symposium on the Foundations of Computer Science*. IEEE, October 1984, pp. 241–250.

[11] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[12] J. M. Lee, *Introduction to Smooth Manifolds*. New York, NY: Springer New York, 2003.

[13] R. Luna and K. E. Bekris, "Efficient and complete centralized multi-robot path planning," in *IEEE International Conference on Intelligent Robots and Systems*, 2011, pp. 3268–3275.

[14] C. I. Mavrogiannis and R. A. Knepper, "Decentralized multi-agent navigation planning with braids," in *Workshop on the Algorithmic Foundations of Robotics*, 2016.

[15] ——, "Multi-agent trajectory prediction and generation with topological invariants enforced by hamiltonian dynamics," in *Workshop on the Algorithmic Foundations of Robotics*, 2018.

[16] M. Morse, *The calculus of variations in the large*, ser. Colloquium Publications. American Mathematical Society, 1934, vol. 18.

[17] J. R. Munkres, *Topology: a first course*. Prentice-Hall, 1974.

[18] A. Orthey and M. Toussaint, "Rapidly-exploring quotient-space trees: Motion planning using sequential simplifications," *International Symposium of Robotics Research*, 2019.

[19] A. Orthey, B. Frész, and M. Toussaint, "Motion planning explorer: Visualizing local minima using a local-minima tree," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 346–353, April 2020.

[20] E. Páll, A. Sieverling, and O. Brock, "Contingent contact-based motion planning," in *IEEE International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 6615–6621.

[21] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy, "Topological trajectory classification with filtrations of simplicial complexes and persistent homology," *International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 204–223, 2016.

[22] M. Ryan, "Constraint-based multi-robot path planning," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 922–928.

[23] T. Siméon, S. Leroy, and J. P. Laumond, "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 42–49, 2002.

[24] S. Smale, "On the topology of algorithms, i," *Journal of Complexity*, vol. 3, pp. 81–89, 1987.

[25] K. Solovey and D. Halperin, "k-color multi-robot motion planning," *International Journal of Robotics Research*, vol. 33, no. 1, pp. 82–97, 2014.

[26] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," *International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, 2016.

[27] S. Söntges and M. Althoff, "Computing possible driving corridors for automated vehicles," *IEEE Intelligent Vehicles Symposium (IV)*, pp. 160–166, 2017.

[28] N. E. Steenrod, "The topology of fibre bundles," 1951.

[29] I. A. Şucan, M. Moll, and L. Kavraki, "The open motion planning library," *IEEE Robotics and Automation Magazine*, 2012.

[30] P. Svestka and M. H. Overmars, "Coordinated path planning for multiple robots," *IEEE Robotics and Autonomous Systems*, vol. 23, no. 3, pp. 125–152, 1998.

[31] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 4044–4051.

[32] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha, "Centralized path planning for multiple robots: Optimal decoupling into sequential plans," in *Robotics: Science and Systems*, Seattle, USA, June 2009.

[33] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.

[34] W. Wu, S. Bhattacharya, and A. Prorok, "Multi-Robot Path Deconfliction through Prioritization by Path Prospects," *arXiv preprint arXiv:1908.02361*, 2019.

[35] Y. Yang and O. Brock, "Elastic roadmaps—motion generation for autonomous mobile manipulation," *Autonomous Robots*, vol. 28, no. 1, p. 113, 2010.

[36] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[37] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, J. A. D. Bagnell, and S. Srinivasa, "CHOMP: Covariant Hamiltonian Optimization for Motion Planning," *International Journal of Robotics Research*, 2013.