Sparse Multilevel Roadmaps for High-Dimensional Robotic Motion Planning

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

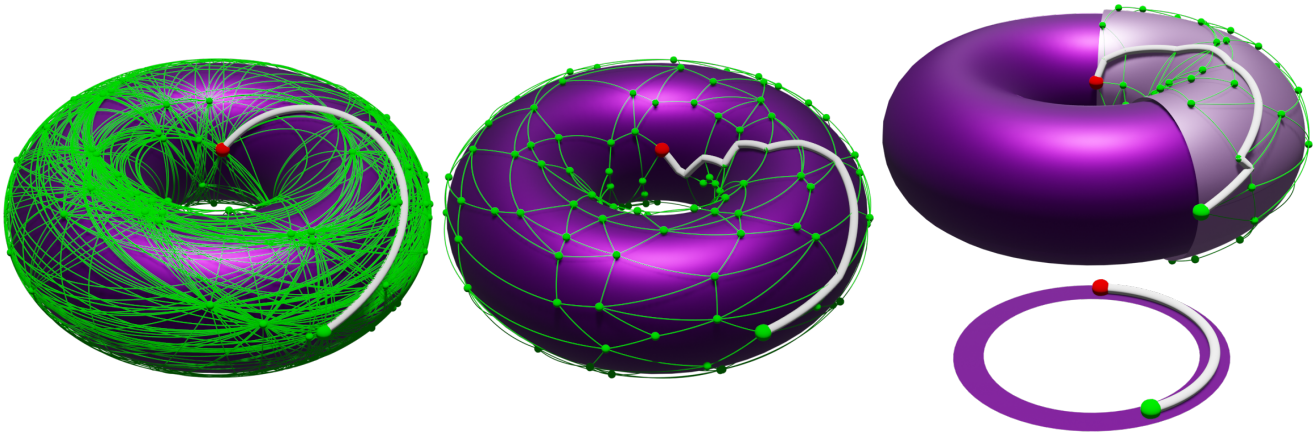


Fig. 1: We generalize sparse roadmaps to fiber bundles. Here, we demonstrate this idea on the Torus $T^2 = S^1 \times S^1$ with S^1 being the circle. **Left**: Dense roadmap using probabilistic roadmap planner [12]. **Middle**: Sparse roadmap using sparse roadmap spanner [3]. **Right**: Sparse multilevel roadmap on fiber bundle $T^2 \to S^1$ using our algorithm (SMLR), which restricts sampling based on information from the lower-dimensional space S^1.

***Abstract*— Sparse roadmaps are important to compactly represent state spaces, to determine problems to be infeasible and to terminate in finite time. However, sparse roadmaps do not scale well to high-dimensional planning problems. In prior work, we showed improved planning performance on high-dimensional planning problems by using multilevel abstractions to simplify state spaces. In this work, we generalize sparse roadmaps to multilevel abstractions by developing a novel algorithm, the sparse multilevel roadmap planner (SMLR). To this end, we represent multilevel abstractions using the language of fiber bundles, and generalize sparse roadmap planners by using the concept of restriction sampling with visibility regions. We argue SMLR to be probabilistically complete and asymptotically near-optimal by inheritance from sparse roadmap planners. In evaluations, we outperform sparse roadmap planners on challenging planning problems, in particular problems which are high-dimensional, contain narrow passages or are infeasible. We thereby demonstrate sparse multilevel roadmaps as an efficient tool for feasible and infeasible high-dimensional planning problems.**

I. Introduction

Sparse roadmaps [3] are essential in motion planning tasks to reduce model complexity and terminate motion planning in finite time, thereby providing (probabilistic) infeasibility proofs. Such infeasibility proofs are essential if we like to use a motion planner as building block for larger action skeletons [11] or symbolic planning systems [37]. However, sparse roadmaps often operate on the full state space of the robot(s), thereby taking too much time to converge—making them often inapplicable for higher-dimensional systems.

To address this problem, we propose to use sparse roadmaps [3] in conjunction with multilevel abstractions of the state space [22]. By exploiting multilevel abstractions—which we model using fiber bundles [35]—we can often terminate the algorithm significantly faster than state-of-the-art sparse roadmap planners operating on the full state space.

While multi-resolution roadmaps exists [7], [30], we are not aware of any algorithm to compute sparse roadmaps over multilevel abstractions. We therefore believe to be the first to combine both concepts into one concise algorithm. Let us summarize our contributions as follows.

1) We present the Sparse MultiLevel Roadmap planner (SMLR), which generalizes sparse roadmaps [3] to efficiently exploit fiber bundle structures [22]
2) We evaluate SMLR on eight challenging feasible and infeasible motion planning problems involving high-dimensional state spaces up to 34-degrees of freedom (dof)

II. Related Work

We review two aspects of (sampling-based) motion planning [15]. First, we discuss multilevel motion planning, where we plan over multiple levels of abstraction. Second, we discuss sparse roadmaps on general state spaces. We will investigate both topics in detail in Sec. III.

[1]Max Planck Institute for Intelligent Systems, Stuttgart, Germany. Marc Toussaint thanks the MPI-IS for the Max Planck Fellowship.

[2]Technical University of Berlin, Berlin, Germany {aorthey}@is.mpg.de, {toussaint}@tu-berlin.de

A. Multilevel Motion Planning

To efficiently solve high-dimensional motion planning problems, we can use the framework of multilevel motion planning [5], [32], [27], [38], [22], where (admissible) lower-dimensional projections are used to simplify the state space of a robot. We can construct multilevel abstractions either manually [26], [24] or learn them from data [8], [2]. Our approach is complementary, in that we assume a multilevel abstraction to be given and we concentrate on computing sparse roadmaps over those abstractions.

Once we fix a multilevel abstraction, we can utilize classical motion planning algorithms to exploit them. A popular choice is the rapidly-exploring random tree algorithm [14], which we can generalize to selectively grow samples towards regions informed by lower-dimensional abstractions [8], [24] or workspace information [28]. While algorithms often show speed-ups of two to three orders of magnitude [28], [36], they usually lack guarantees on asymptotic optimality [12]. There are, however, two planner which provide those guarantees. First, the quotient-space roadmap planner (QMP*) [22], [23], which generalizes the probabilistic roadmap planner (PRM*) [12]. Second, the hierarchical bi-directional fast marching tree (HBFMT*) [26], [27], which generalizes the fast marching trees algorithm (FMT*) [10]. While both guarantee asymptotic optimality [22], [27], they support, however, either only euclidean spaces [27] or rely on dense roadmaps [22]. Our approach differs significantly, in that we are the first to compute sparse roadmaps over general multilevel abstractions—while providing guarantees on asymptotic near-optimality.

B. Sparse Roadmaps

The history of sparse roadmaps essentially begins with the pioneering work by Siméon et al. [33], who were the first to prune states based on visibility regions. With visibility regions, we try to find a minimal set of states from which the full state space is visible, similar to the concept of guards in the art gallery problem [21]. However, visibility roadmaps often sacrifice on path quality. As remedies, we could introduce cycles [31], [20] or use edge visibility [9] to improve path quality.

While cycles and edge visibility can improve path quality, there are no guarantees on optimality. This changed with the advent of near-optimal sparse roadmaps [17]. Using dense asymptotic optimal roadmaps [12], we can use graph spanners to sparsify a dense roadmaps while providing guarantees on path quality. We can achieve this by either removing edges [17], [39] or edges and vertices [29]. Computing dense roadmaps before sparsification is, however, computationally expensive. Later work introduces incremental sparse graph spanners, with which we can remove dependence on dense roadmaps altogether [3]. Our work is complementary to sparse graph spanners, in that we also use incremental sparse graph spanners [3]. We differ, however, in building not one, but multiple sparse roadmaps on different abstraction levels.

When using sparse roadmaps, we often face the problem of explicitly defining a visibility or connection radius to define the sparseness of the graph. To handle this trade-off between optimality and efficiency, we can often create multi-resolution roadmaps [4]. Multi-resolution roadmaps are sets of roadmaps which differ in how sparse they are. To vary roadmap sparsity, we could change the connection radius [30] or we can selectively remove edges, either evenly distributed [7] or based on a reliability criterion [19]. To exploit those multi-resolution roadmaps, we could plan on the highest resolution roadmap and selectively refine the roadmap whenever we hit an obstacle [30]. Such a strategy is efficient, because solutions on sparser roadmaps act as admissible heuristics for planning [1], [4]. While multi-resolution roadmaps exist on the same state space, our approach is complementary, in that we create sparse multilevel roadmaps on different state spaces, whereby each state space represents a relaxed planning problem.

III. Background

We develop an algorithm which grows sparse roadmaps over fiber bundles to efficiently exploit high-dimensional planning problems. As background for this task, we review the topics of optimal motion planning, multilevel abstractions (modelled using fiber bundles) and sparse roadmaps.

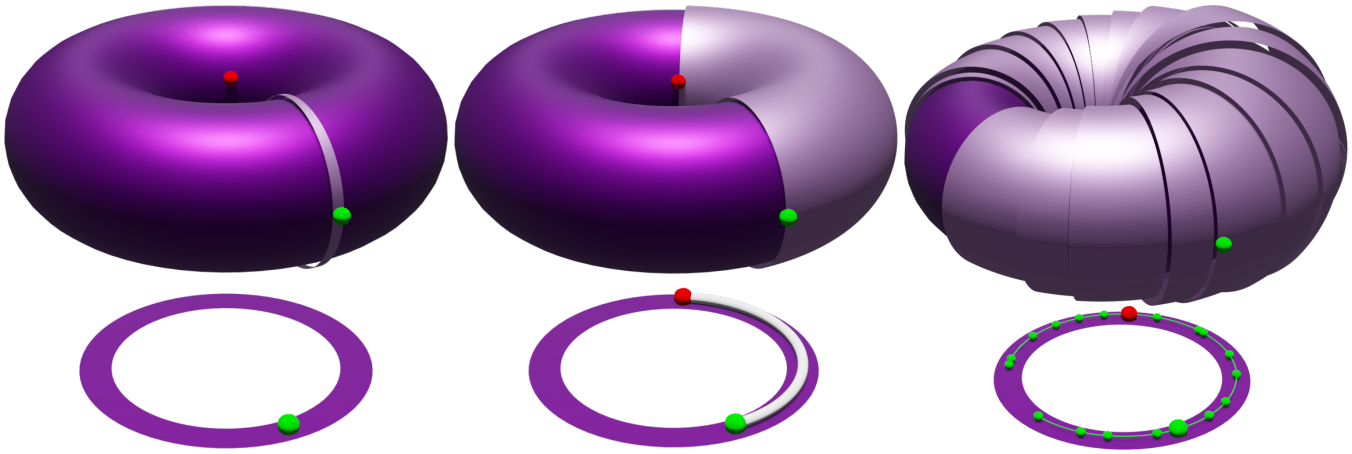
A. Optimal Motion Planning

Let X be an n-dimensional state space and let x_I and x_G be two states in X which we call the initial and the goal state. To each state space, we associate a metric function $d : X \times X \to \mathbb{R}$ and a constraint function $\phi : X \to \{0, 1\}$ which evaluates to zero if a state is feasible and to one otherwise. The state space thus splits into two components, the constraint-free subspace $X_{\text{free}} = \{x \in X \mid \phi(x) = 0\}$ and its complement. We define the optimal motion planning problem as the tuple $A = (X_{\text{free}}, x_I, x_G, J)$, which requires us to design an algorithm to find a continuous path from x_I to x_G while (1) staying exclusively inside X_{free} and (2) minimizing the cost functional J which maps paths in X_{free} to real numbers.

We define a motion planning algorithm (a planner) as a mapping from A to a path through X_{free}. A planner can have different desirable properties. First, we like a planner to be *probabilistically complete*, meaning the probability of finding a solution path if one exists approaches one as time goes to infinity. Second, we like a planner to be *asymptotically near-optimal*, meaning the probability of finding a path is at least ϵ worse than the optimal solution path (under cost functional J). Third, we like a planner to be *asymptotically sparse*, meaning the probability of adding new nodes and edges converges to zero if time goes to infinity [3].

B. Multilevel Motion Planning

Because state spaces are often too high-dimensional to plan in, we use multilevel abstractions which we model using fiber bundles [35], [16]. A fiber bundle is a tuple (X, B, F, π), consisting of a bundle space X, a base space B, a fiber space F and a projection mapping π from X to B. We assume that both state space and base space



(a) Fiber over base point. (b) Path restriction over base path. (c) Graph restriction over sparse base graph.

Fig. 2: Fiber bundle restrictions on the fiber bundle $T^2 \to S^1$ with T^2 being the torus and S^1 being the circle. See text for clarification.

have associated constraint functions ϕ and ϕ_B and that the projection mapping π is admissible w.r.t. the constraint functions, i.e. $\phi_B(\pi(x)) \leq \phi(x)$ for any x in X [24]. The admissibility condition ensures that we preserve feasible solution paths under projection. While we exclusively use product spaces in this work, we model them using fiber bundles since they provide a useful vocabulary (restrictions and sections) and since they are required for extensions to task-space projections.

Our approach uses the following three concepts. First, we define fibers over a base element b in B as $F(b) = \{x \in X \mid \pi(x) = b\}$, which is the set of points in X projecting onto b. Please see Fig. 2a for an example of a fiber on the torus $T^2 = S^1 \times S^1$ with base space S^1. We additionally define the method $\textsc{Lift} : B \times F \to X$, which takes a base element b and a fiber element f in $F(b)$ to the bundle space. In the case of product spaces, we can define $\textsc{Lift}(b, f) = (b, f)$. Second, we define path restrictions over a base path $p : I \to B$ as $r(p) = \{x \in X \mid \pi(x) \in p[I]\}$, whereby I is the unit interval and $p[I]$ is the image of the base path in B. Please see Fig. 2b. Third, we define graph restrictions over a graph $G_B = (V_B, E_B)$ on B as $r(G_B) = \{x \in X \mid \pi(x) \in e[I], e \in E_B\}$ whereby V_B are vertices in B, E_B is the set of edges in B and $e[I]$ is the image of an edge on the base space. Fig. 2c provides a visualization of a graph restriction (individual edge restrictions have different distances from torus for better visualization). For more details, please see [22] or [35].

C. Sparse Roadmaps

To grow a sparse roadmap, we use the algorithm by Dobson and Bekris [3]. The sparse roadmap planner is similar to probabilistic roadmaps [13], [12], but uses a visibility region δ, which consists of all feasible states in the hypersphere of radius δ around a state, to prune samples. To implement the pruning step, we add a new feasible sample if and only if it fulfills a sparseness condition.

The sparseness condition consists of four elementary tests [3]. First, we test for coverage, meaning we add the sample if it does not lie in the visibility region of any sample in the graph. Second, we test for connectivity, meaning we add the sample, if it lies in multiple visibility regions, which belong to disconnected components of the sparse graph. Third, we test for interfaces, meaning we add the sample, if it lies in multiple visibility regions, which are not yet connected by an edge. Fourth and finally, we test for shortcuts, meaning we add the sample, if it provides proof of a shorter path through the free state space. We terminate the algorithm, if we either find a feasible path or if we fail M consecutive times to add a sample to the sparse roadmap. For more details please see [3].

The sparse roadmap planner is probabilistically complete and asymptotically near-optimality [3] and depends on the following parameters. First, the visibility region δ, which is usually a fraction of the measure of the state space. Second, the maximum number of consecutive failures M. M is important in the analysis of the algorithm, because it provides a probabilistic estimation of the free state space covered, which is defined as the percentage $1 - \frac{1}{M}$ [34]. As an example, if we stop with $M = 100$, our probabilistic estimate of the free state space covered is 99%. Finally, we have an additional parameter for testing for shortcuts, which provides a trade-off between optimality and efficiency [3].

IV. Sparse Multilevel Roadmaps

Let $(x_I, x_G, X_1, \ldots, X_K)$ be a fiber bundle sequence with x_I and x_G being start and goal state. Our task is to generalize the sparse roadmap planner [3] to fiber bundle sequences by growing K graphs $(G_1, \ldots, G_K)$ on the bundle spaces $(X_1, \ldots, X_K)$, whereby we grow the k-th graph using restriction sampling [22] of the $(k-1)$-th graph. We call our algorithm the sparse multilevel roadmap planner (SMLR). SMLR depends on three parameters, the two parameters δ and M from sparse roadmaps, and the additional parameter η, which we detail later.

We show the algorithm in Alg. 1. We start to create a priority queue (Line 1.1), which orders bundle spaces

Algorithm 1 SMLR($x_I, x_G, X_1, \ldots, X_K$)

1: Let $\mathbf{X}$ be a PRIORITY QUEUE ▷ Top is Max Value
2: **for** X_{cur} in $X_1, \ldots, X_K$ **do**
3: $\mathbf{X}$.PUSH($X_{\text{cur}}, 1$)
4: SECTIONTEST(X_{cur}) ▷ See [25]
5: **while** $\neg$PTC(X_{cur}) **do**
6: $X_{\text{top}} = \mathbf{X}$.POP
7: $x_{\text{rand}} \leftarrow$ RESTRICTIONSAMPLING(X_{top})
8: ADDCONDITIONAL($x_{\text{rand}}, G_{\text{top}}$)
9: $i \leftarrow$ COMPUTEIMPORTANCE(X_{top}) ▷ In $[0, 1]$
10: $\mathbf{X}$.PUSH(X_{top}, i)
11: **end while**
12: **end for**

Algorithm 2 RestrictionSampling(X_k)

1: **if** EXISTS(X_{k-1}) **then**
2: $e \leftarrow$ SAMPLEEDGE(G_{k-1})
3: $x_{\text{base}} \leftarrow$ SAMPLEUNIFORM(e) ▷ State on Edge
4: $\delta_{\text{bias}} \leftarrow$ SMOOTHPARAMETER$(0, \delta, \eta)$
5: **if** RANDOM$(0, 1) < \delta_{\text{bias}}/\delta$ **then**
6: $x_{\text{base}} \leftarrow$ UNIFORMNEAR($x_{\text{base}}, \delta_{\text{bias}}$)
7: **end if**
8: $x_{\text{fiber}} \leftarrow$ SAMPLE(x_{base}, F_k) ▷ Element of F_k
9: $x_{\text{rand}} \leftarrow$ LIFT($x_{\text{base}}, x_{\text{fiber}}$) ▷ Element of X_k
10: **else**
11: $x_{\text{rand}} \leftarrow$ SAMPLE(X_k)
12: **end if**
13: **return** x_{rand}

depending on an importance criterion i, which we detail later. We sort the queue such that the space with the maximum value is on top. We then iterate over the bundle spaces from X_1 to X_K (Line 1.2) and push the current space onto the priority queue with an importance of 1 (Line 1.3). We then execute a section test (Line 1.4), where we search for a feasible solution over the path restriction of the solution path (if any) on the previous bundle space $X_{\text{cur}-1}$. The SECTIONTEST method helps to overcome narrow passages, but is not essential for the understanding of this paper – we use it as a black box within SMLR. Please see our previous publication [25] for more information.

We then grow the roadmaps $(G_1, \ldots, G_{\text{cur}})$ as long as the planner terminate condition (PTC) of the current bundle space X_{cur} is not fulfilled (Line 1.5). In our case, we terminate if a solution is found or if we reach either the infeasibility criterion or a time limit. Inside the while loop, we take the top bundle space X_{top} with the largest importance value (Line 1.6) and sample a random point using RESTRICTIONSAMPLING (Line 1.7). We then add the point to the graph with ADDCONDITIONAL (Line 1.8), if it fulfills the sparseness condition [3], which we detail in Sec. III. Finally, we recompute the importance of the bundle space (Line 1.9) and push the space back onto the queue (Line 1.10).

The two methods RESTRICTIONSAMPLING and COMPUTEIMPORTANCE are further detailed in the next two subsections. To facilitate understanding, we give first a brief overview of each. First, in RESTRICTIONSAMPLING, we restrict sampling on the bundle space by using information from the graph on its base space. We differ from dense roadmaps by using the visibility region of the sparse graph which depends on the visibility range δ. Second, in COMPUTEIMPORTANCE, we use the sampling density of the sparse graph together with the number of consecutive failures to estimate the importance of the bundle space and thereby its position in the priority queue. Next, we discuss each method in more detail and provide an analysis of the algorithm.

A. Restriction Sampling with Visibility Regions

Let X_k be a bundle space with graph G_k, and let X_{k-1} be its base space with graph G_{k-1}. To grow the graph G_k, we use the framework of restriction sampling [22]. In restriction sampling, we sample states on X_k by uniformly sampling from the graph restriction of G_{k-1} (see Sec. III). To give guarantees on asymptotic optimality, we would need the vertices of G_{k-1} to become dense in the free state space.

To avoid using a dense graph for sampling [3] while giving guarantees on asymptotic near-optimality, we opt to exploit the graph visibility region. The visibility region of a graph G is the set $V(G, \delta) = \{x \in X \mid d(x, e[I]) \leq \delta \text{ for some } e \text{ in } G\}$, whereby d is the metric on X, e is an edge from G and $e[I]$ is the image of the edge in X.

To sample the graph visibility region, we use the restriction sampling algorithm depicted in Alg. 2. The algorithm requires an existing base graph G_{k-1} (Line 2.1), then samples a random state on a random edge (Line 2.2). Sampling the visibility region directly would be too uninformative. We thus use a smoothly varying parameter $\delta_{\text{bias}} \in [0, \delta]$, which first restricts sampling to the sparse graph ($\delta_{\text{bias}} = 0$), then smoothly increase in each iteration until the whole visibility region δ. This situation is visualized in Fig. 3. To control the rate of change of δ_{bias}, we use the parameter η.

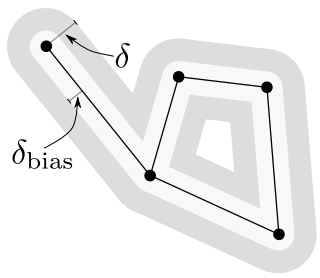


Fig. 3: Visibility region $V(G, \delta)$ of a graph G.

In particular for narrow passages, it is often crucial to sample directly on the graph restriction. We thus sample the visibility region (Line 2.6) only in a certain percentage of cases, depending on δ_{bias}. Once a base element is chosen, we sample a corresponding fiber space element (Line 2.8), lift the states (Line 2.9) and return the state (Line 2.13). If no base graph exists, we revert to a uniform sampling of the space (Line 2.11).

B. Importance and Ordering of Bundle Spaces

To grow sparse multilevel roadmaps, we need to decide which roadmap on which level we should grow next, i.e. we

need an ordering of bundle spaces. In prior work [22], we advocated the use of an exponential importance criterion $i(X_k) = 1/(|V_k|^{1/n_k} + 1)$, with $|V_k|$ being the vertices on the graph G_k on X_k and n_k being the dimensionality of X_k, which was motivated by the sampling density of the graph which is proportional to $|V_k|^{1/n_k}$ [6].

However, sampling density is not good criterion for sparse roadmaps, because we care more about the coverage of the free space. To account for the coverage of the free space, we advocate an importance criterion using M_k, the number of consecutive sample failures. The number M_k provides an estimate of the free space coverage, namely as the percentage $1-\frac{1}{M_k}$ [33]. The higher M_k, the less often we should sample X_k. We formulate the importance criterion thus as

$$i(X_k) = \frac{1}{M_k + 1}. \tag{1}$$

Note that we stop the algorithm only if $M_k > M$ *and* X_k is the current bundle space X_{cur}. Since $i(X_k)$ will eventually converge to zero, we ensure that every bundle space up until k would be chosen infinitely many times. This is an important requirement to provide asymptotic guarantees of the algorithm.

C. Analysis of Algorithm

To prove SMLR to be asymptotically near-optimal and asymptotic sparse, we need to prove that restriction sampling with visibility regions is dense in the free state space of the last bundle space X. Since the importance criterion in Eq. (1) eventually converges to zero, we can thus ensure that we produce an infinite sampling sequence on the free state space X_{free}. Therefore, when using sparse roadmap spanner [3] to grow the roadmap on X, we retain all their properties, which include asymptotic near-optimality and asymptotic sparseness. However, we might reduce the number of vertices considerably.

Let us prove that restriction sampling with visibility regions is dense in the *free* state space X_{free} on the fiber bundle (X, B, F, π). This argument can be applied recursively to prove the same for fiber bundle sequences [22]. Note that we use the set-theoretic definition of dense, which states that a set A is dense in a space X if the intersection of A with any non-empty open subset U of X is non-empty [18].

Theorem 1: Restriction sampling with visibility regions on X produces a sampling sequence $A = \{x_m\}$, which is dense in X_{free}.

Proof: Let U be an arbitrary open set in X_{free}. Since π is admissible, the projection $\pi(U)$ of U onto B is an open subset of the free base space [24]. Since uniform sampling on B with visibility regions will eventually cover the free base space [33], $\pi(U)$ will be a subset of the visibility region of the graph on B. When the number of samples goes to infinity, we revert to uniform sampling of the graph restriction and will thus sample $\pi(U)$ infinitely many times. By sampling the fiber over $\pi(U)$, we thus eventually obtain a sample x in U. Since U was arbitrary, the sequence is dense in X_{free}. ■

	06D Bugtrap [feasible]	**06D Bugtrap [infeasible]**	**06D drone [feasible]**	**06D drone [infeasible]**	**07D kuka [feasible]**	**07D kuka [infeasible]**	**34D PR2 [feasible]**	**34D PR2 [infeasible]**
SMLR (ours)	**4.37** 10|0|0	**2.47** 0|10|0	0.23 10|0|0	**0.72** 0|10|0	**1.42** 10|0|0	**5.34** 0|10|0	**9.25** 9|0|1	**0.32** 0|10|0
SPARS	60.00 0|0|10	60.00 0|0|10	0.37 10|0|0	60.00 0|0|10	33.66 6|0|4	60.00 0|0|10	60.00 0|0|10	60.00 0|0|10
SPARStwo	60.00 0|0|10	60.00 0|0|10	**0.16** 10|0|0	60.00 0|0|10	34.86 7|0|3	60.00 0|0|10	60.00 0|0|10	60.00 0|0|10

TABLE I: Runtime (in seconds) of motion planner averaged over 10 runs with 60s time limit. We additionally show how often a planner terminated with a status of feasible|infeasible|timeout on each scenario.

V. Evaluation

To evaluate SMLR, we compare its performance on eight scenarios against the algorithms SPARS and SPARS2 from the open motion planning library (OMPL). Both SPARS and SPARS2 are the only algorithms in OMPL we know of which can return on infeasible scenarios while not timing out. To ensure a fair comparison, we set the parameters of SMLR, SPARS and SPARS2 all to $M = 1000$, $\delta = 0.25\mu$ with μ being the measure of the state space (removing effects stemming from different parameter values). For SMLR, we use the parameter $\eta = 1000$ which designates how fast we expand the graph visibility region for restriction sampling.

While we like our algorithm to correctly declare an infeasible problem as infeasible, we also like to make sure that the algorithm does not show false negatives, i.e. declaring a feasible problem to be infeasible. To ensure correctness, we always use two similar scenarios, one which is feasible and one which is infeasible. For all scenarios, we run each algorithm 10 times with a time limit of 60s. Our setup is a 8GB RAM 4-core 2.5GHz laptop running Ubuntu 16.04.

A. 6-dimensional Bugtrap

Our first scenario is the classical narrow-passage Bugtrap scenario, where a cylindrical robot (the bug with 6 degrees of freedom (dof)) has to escape a spherical object with a narrow exit (the trap), as shown in Fig. 4. We use two versions, a feasible one with a bug which barely fits through the exit, and an infeasible one where the bug does not fit. As a simplification, we use an inscribed sphere which we describe using the fiber bundle $SE(3) \to \mathbb{R}^3$. We show the results in Table I. While SMLR can solve (on average) both scenarios in 4.37 and 2.47s, respectively, both SPARS and SPARS2 time out after 60s.

B. 6-dimensional Drone

In the second scenario, we use a free-floating drone with 6-dof. The drone has to traverse a room which is separated by a

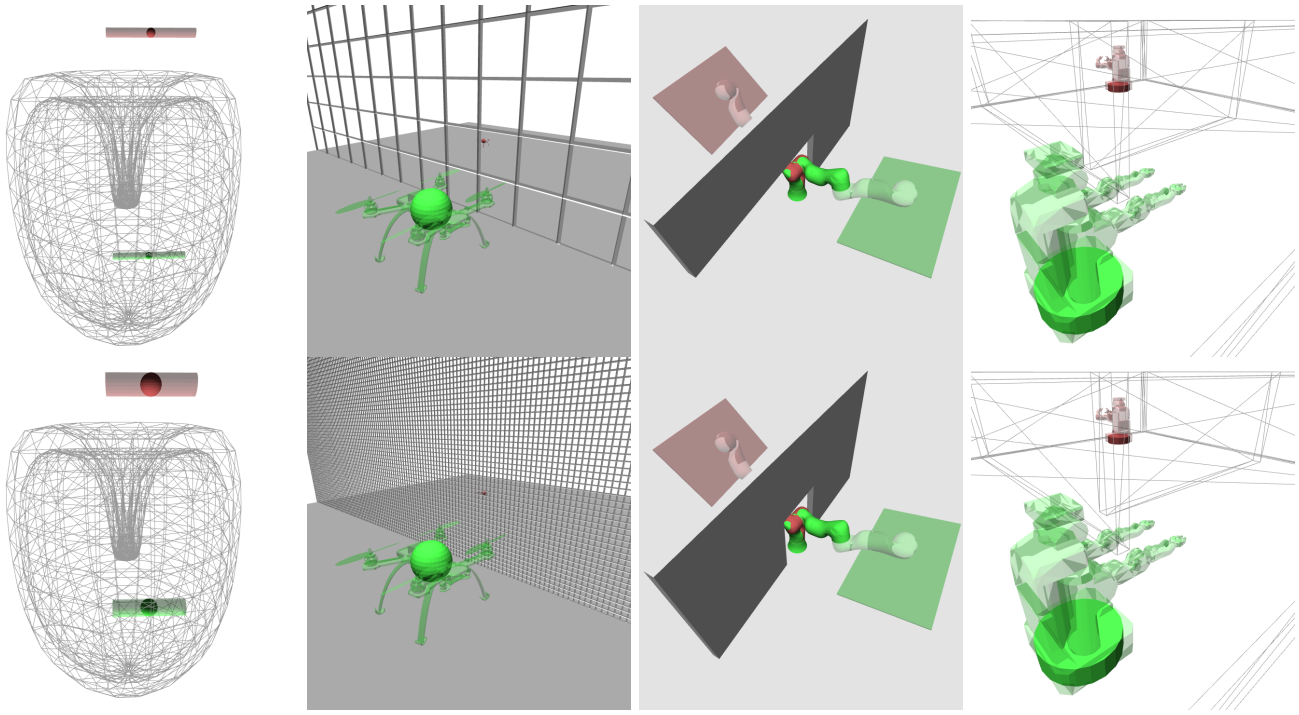


Fig. 4: The eight scenarios used for evaluating our algorithm. Task for each scenario is to move robot from initial state (green) to goal state (red). Full robot geometry is shown as transparent color, the simplified version as non-transparent (Note that we use two simplifications for PR2, but only show one). **Top Row**: Feasible scenarios, where a solution exists. **Bottom Row**: Infeasible scenarios, where no solution exists. **Left to Right**: 6-dimensional Bugtrap, 6-dimensional drone, 7-dimensional KUKA LWR, 34-dimensional PR2.

net. In the first version of the problem, we make the net large enough to let the drone fly trough (the feasible problem). In the second version, we make the net finely woven to prevent the drone from passing (the infeasible problem). As a simplification, we use a sphere at the center of the drone. We model this situation with the fiber bundle $SE(3) \to \mathbb{R}^3$. For the feasible scenario, all three planners solve the problem with SPARS2 taking 0.16s, SMLR taking 0.23s and SPARS taking 0.37s. In the infeasible scenario, only SMLR solves the problem in 0.72s, while SPARS and SPARS2 both time out.

C. 7-dimensional KUKA LWR

In the third scenario, we use a fixed-base KUKA LWR robot with 7-dof, which has to transport a windshield through a gap in a wall (Fig. 4). We create two versions, a feasible one with the gap in the wall and an infeasible one where we close the gap. As a simplification, we use a projection onto the first two links of the manipulator arm, which we describe using the fiber bundle $\mathbb{R}^7 \to \mathbb{R}^3$. With our algorithm SMLR, we can solve both scenarios in 1.42s and 5.34s. For the feasible scenario, SPARS requires 33.66s (but times out in 4 cases) and SPARS2 requires 34.86s (but times out in 3 cases). Both SPARS algorithms time out for the infeasible scenario in all runs.

D. 34-dimensional PR2

In the fourth scenario, we use the mobile-base PR2 robot with 34-dof, which has to enter a room with a small opening as shown in Fig. 4. We use again two scenarios, the feasible one with the opening and an infeasible one where we close the opening. As a simplification, we use two projections, first we remove the arms of the robot and second we project onto the mobile base. We model this situation by the fiber bundle sequence $\mathbb{R}^{34} \to \mathbb{R}^7 \to \mathbb{R}^2$. Our algorithm SMLR requires 9.25s to solve the feasible scenario (but times out in 1 case) and it requires 0.32s to terminate on the infeasible scenario. Both SPARS and SPARS2 cannot solve any of the runs in the time limit given.

VI. Conclusion

We presented the sparse multilevel roadmap planner (SMLR), which we believe to be the first algorithm to generalize sparse roadmap spanners [3] to fiber bundles [22], which are models of multilevel abstractions. Our algorithm exploits multilevel abstraction using the notion of restriction sampling with visibility regions. We have shown SMLR to be asymptotically near-optimal and asymptotically sparse by showing restriction sampling to produces a dense sampling sequence. In evaluations, we showed SMLR to efficiently and correctly terminate on feasible and infeasible problems, even when those problems have narrow passages, intricate geometries or state spaces with dimensions of up to 34-dof.

REFERENCES

[1] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A*," *International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[2] M. Brandao and I. Havoutis, "Learning sequences of approximations for hierarchical motion planning," in *International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 508–516.

[3] 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.

[4] W. Du, F. Islam, and M. Likhachev, "Multi-resolution A*," 2020, arXiv:2004.06684 [cs.RO].

[5] P. Ferbach and J. Barraquand, "A method of progressive constraints for manipulation planning," *IEEE Transactions on Robotics*, vol. 13, no. 4, pp. 473–485, 1997.

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

[7] J. Ichnowski and R. Alterovitz, "Multilevel incremental roadmap spanners for reactive motion planning," in *IEEE International Conference on Intelligent Robots and Systems*. IEEE, 2019, pp. 1504–1509.

[8] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.

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

[10] 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," *International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[11] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1470–1477.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[13] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics*, vol. 12, no. 4, pp. 566–580, 1996.

[14] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2000, pp. 995–1001.

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

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

[17] J. D. Marble and K. E. Bekris, "Asymptotically near-optimal planning with probabilistic roadmap spanners," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 432–444, 2013.

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

[19] S. Murray, G. D. Konidaris, and D. J. Sorin, "Roadmap subsampling for changing environments," in *IEEE International Conference on Intelligent Robots and Systems*. IEEE, 2020.

[20] D. Nieuwenhuisen and M. H. Overmars, "Useful cycles in probabilistic roadmap graphs," in *IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 2004, pp. 446–452.

[21] J. O'Rourke, *Art gallery theorems and algorithms*. Oxford University Press Oxford, 1987, vol. 57.

[22] A. Orthey, S. Akbar, and M. Toussaint, "Multilevel motion planning: A fiber bundle formulation," 2020, arXiv:2007.09435 [cs.RO].

[23] A. Orthey, A. Escande, and E. Yoshida, "Quotient-space motion planning," in *IEEE International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 8089–8096.

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

[25] ——, "Section patterns: Efficiently solving narrow passage problems using multilevel motion planning," 2020, arXiv:2010.14524 [cs.RO].

[26] W. Reid, R. Fitch, A. H. Göktoğan, and S. Sukkarieh, "Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover," *Journal of Field Robotics*, 2019.

[27] W. Reid, R. Fitch, A. H. Göktoğgan, and S. Sukkarieh, "Motion planning for reconfigurable mobile robots using hierarchical fast marching trees," in *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 656–671.

[28] M. Rickert, A. Sieverling, and O. Brock, "Balancing exploration and exploitation in sampling-based motion planning," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1305–1317, 2014.

[29] O. Salzman, D. Shaharabani, P. K. Agarwal, and D. Halperin, "Sparsification of motion-planning roadmaps by edge contraction," *International Journal of Robotics Research*, vol. 33, no. 14, pp. 1711–1725, 2014.

[30] B. Saund and D. Berenson, "Fast planning over roadmaps via selective densification," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2873–2880, 2020.

[31] E. Schmitzberger, J.-L. Bouchet, M. Dufaut, D. Wolf, and R. Husson, "Capture of homotopy classes with probabilistic road map," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2002, pp. 2317–2322.

[32] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars, "Multilevel path planning for nonholonomic robots using semiholonomic subsystems," *International Journal of Robotics Research*, vol. 17, no. 8, pp. 840–857, 1998.

[33] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.

[34] 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.

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

[36] S. Tonneau, A. D. Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An Efficient Acyclic Contact Planner for Multiped Robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, June 2018.

[37] M. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Robotics: Science and Systems*, 2018.

[38] E. Vidal, M. Moll, N. Palomeras, J. D. Hernández, M. Carreras, and L. E. Kavraki, "Online multilayered motion planning with dynamic constraints for autonomous underwater vehicles," in *IEEE International Conference on Robotics and Automation*. IEEE, 2019, pp. 8936–8942.

[39] W. Wang, D. Balkcom, and A. Chakrabarti, "A fast online spanner for roadmap construction," *International Journal of Robotics Research*, vol. 34, no. 11, pp. 1418–1432, 2015.