MIT-CSAIL-TR-2013-027 November 18, 2013

On Randomized Path Coverage of Configuration Spaces

Alejandro Perez

On Randomized Path Coverage of Configuration Spaces

Alejandro Perez

Abstract We present a sampling-based algorithm that generates a set of locally-optimal paths that differ in visibility.

1 Introduction

Sampling-based algorithms have been successfully applied to motion planning, a problem known to be PSPACE-hard [35], [20], in robotics [18], [20] and other fields [11], [24], [2]. However, although methods like the Probabilistic RoadMap [16], the RRT [21], and their optimal variants [14], are able to compute solutions to high-dimensional instances of this problem in a timely manner, these return a single feasible path, be it a suboptimal one or an approximation of the optimal solution.

There is an increased interest in applying these approaches to swiftly compute minimal sets of high-quality paths. Indeed, several applications have been proposed in the literature [1], [6]. For example, a small set of paths that are distinct and locally-optimal could allow a robot to efficiently reason about multiple options for a given task [33], compactly represent configuration spaces that are computed offline [23], speedup replanning in dynamic environments [13], serve as local paths used by multiple robots to coordinate distributed motion plans [26], aid the pruning and searching of paths that meet certain constraints [10], and guide the search of protein conformations for multiple energy minima and transition paths [11].

In this paper we present a sampling-based algorithm that computes a set of paths that are locally-optimal in terms of their visibility. The algorithm iteratively constructs and updates a set of *guards* that cover the space with their *visibility domains*. Each guard maintains a trajectory that asymptotically approximates the *locally-optimal path* within its visibility domain. The algorithm avoids explicitly computing the redundancy or deformability between paths and instead removes guards found to be approximating the same locally-optimal path once their current solutions overlap. Because all visibility computations are based on a *local method* that represents the local ‘reachability’ of a given system, the approach is applicable to instances of the motion planning of arbitrary dimensions and is able to generate paths that differ in ways other than their relationship to obstacles. For example, this approach can reason about path difference induced by constraints imposed on the system, or the physical limitations of the platform. The resulting algorithm has a running time comparable to that of standard optimal sampling-based planners, as it requires a minimal set of additional procedures at each iteration and lazily constructs multiple paths simultaneously.

Alejandro Perez, e-mail: atp@csail.mit.edu

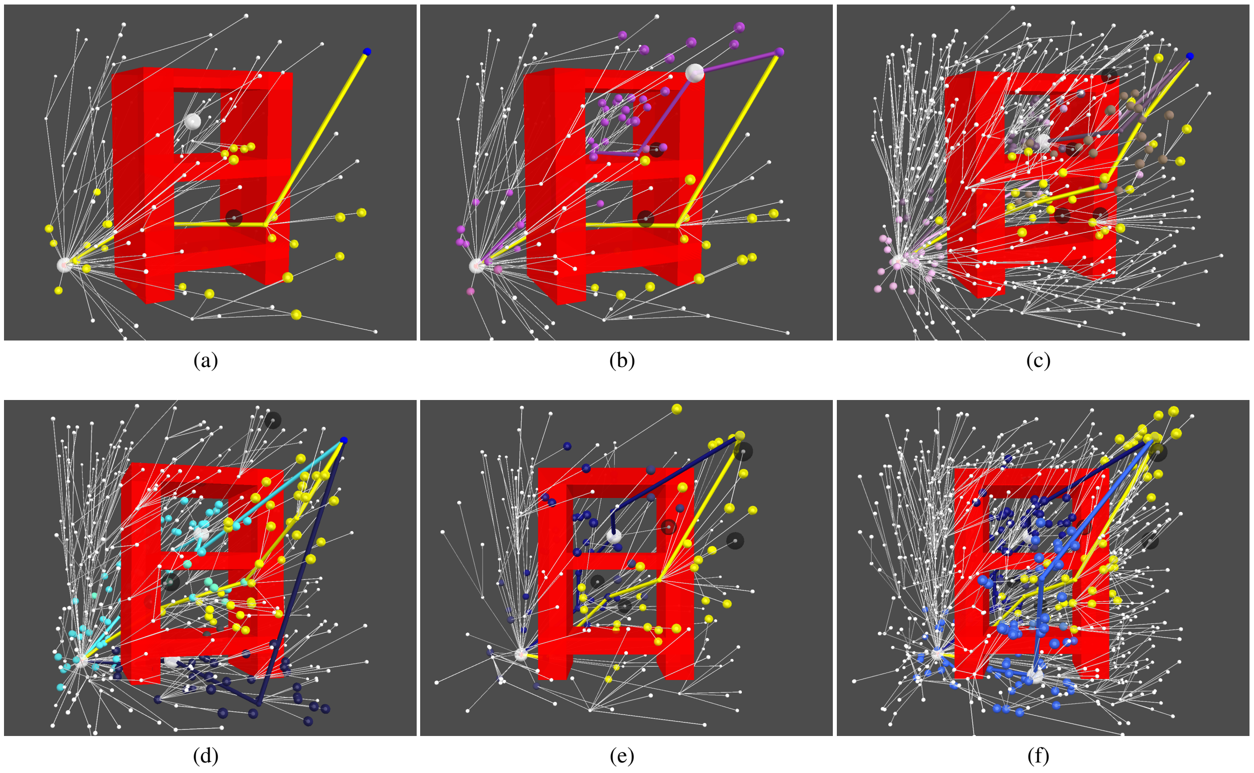

(a) (b) (c) (d) (e) (f)

Fig. 1 The Drawer environment. The proposed algorithm finds three types of paths: one through each hole, and one that avoids the obstacle entirely. Guard nodes are white spheres, dominated guard nodes are black spheres, and locally-optimal approximation for each visibility-domain are shown with a different color. In (a) one path provides a solution through the bottom part of the object (yellow path); a tree (not rendered) rooted at the top opening is being constructed. (b) shows the first guarded paths (purple and yellow lines), dominated guards (black spheres) from dominated paths can be seen. (c) Three locally-optimal solutions are shown; yellow for the bottom opening, gray for the top, and light purple for the path that goes around and behind the object. (d,e) progress during another run in the same environment. (f) Three paths.

2 Related Work

The problem of finding a set of feasible paths in a configuration space has been explored over the last two decades. Brock and Khatib [5] presented an approach to generate sets of paths by computing approximations of free space, or bubbles, and using this information to verify if there exist continuous mappings between paths. However, the different solution classes are implicitly represented as regions of the configuration space, and a path planner is required to find a solution within each one. Lamound and Nissoux presented the Visibility-PRM, a variant of the PRM algorithm that reasons about the visibility between configurations in order to cover the configuration space with a sparse roadmap [19]. This approach was extended by Jaillet and Simeon [12] to allow multiple class-equivalent paths to be constructed if their deformation meets certain criteria. These algorithms probabilistically cover the space with guards and construct compact roadmaps that represents the different classes of paths. However, the resulting coverage is suboptimal as it does not attempt to find optimal locations for the guards or to minimize the cardinality of the set. Moreover, there are no guarantees regarding the quality of the paths.

Another approach that has received a considerable amount of attention is to classify or represent the different homotopy classes in a given space. Bhattacharya et al. consider a representation of the homotopy classes while performing search-based planning on a graph to generate non-homotopic paths [3], [4]. This approach has been successful in generating optimal paths for different instances of the motion planning problem. However, the algorithm is only able to find path homotopies induced by obstacles, is limited to three-dimensional problems, and is complete and optimal only with respect to its discretization of the space.

Schmitzberger et al. [34] define Homotopy Preserving Probabilistic Roadmaps, or roadmaps that include the list of paths that cannot be deformed into one another without colliding with obstacles, and propose an algorithm that constructs them. The resulting approach incorporates guarded visibility domains [19] and considers a family of balls that form a topological base of the configuration space to compute a set with every feasible, non-redundant path such that all paths are non-homotopic. The algorithm was theoretically characterized and successfully applied to a six-dimensional instance of the motion planning problem. However, although the approach attempts to minimize the cardinality of the set of paths by considering redundancy, the paths themselves are of suboptimal quality. Furthermore, the redundancy operation is computationally expensive, as it requires numerous calls to collision-checking procedures.

In this paper we present a sampling-based algorithm that covers the space with the minimum set of *locally-optimal* paths that differ in *visibility*.

3 Problem Formulation

In this section we formalize the *minimum guarded path cover* problem. We begin by presenting the path planning problem in terms of *configurations* and *visibility*.

Let $X \subseteq \mathbb{R}^d$, referred to as the *configuration space*, be a compact set. The elements of X are called *configurations*. Let X_{inv} be an open set, called the *invisible region* and the set defined as $X_{\text{vis}} := X \setminus X_{\text{inv}}$ be the *visible-space*

It can be noted that the term *visibility* represents feasibility and local 'reachability'. In general terms, the configuration x' is *visible* from configuration x if a *local method* is able to compute a feasible path that connects them. We use this notion in this way in order to maintain consistency with the visibility-based algorithms literature [30]. However, this terminology is equivalent to that used in the motion planning literature [20], i.e., visible-space is equivalent to free-space, invisible region to obstacle region, feasible to collision-free, and so on.

Now we consider the problem of finding feasible *paths* composed of configurations in this space.

Definition 1. (Path) A *path* in X is a continuous function $\sigma : [0,1] \to X$. The path σ is considered to be *feasible*, if $\sigma(\tau) \in X_{\text{vis}}$ for all $\tau \in [0,1]$. The set of all feasible paths is denoted by Σ_{vis}.

Problem 1. (Path Planning) Given an initial configuration x_{init}, an invisible region X_{inv}, and a *goal configuration* x_{goal}, find a feasible path $\sigma : [0,1] \to X_{\text{vis}}$ that starts from the initial configuration $\sigma(0) = x_{\text{init}}$ and reaches the goal configuration $\sigma(1) = x_{\text{goal}}$.

Problem 2. (Optimal Path Planning [14]) Given a *cost functional* $c : \Sigma_{\text{vis}} \to \mathbb{R}_{\geq 0}$ that maps each feasible path to a non-negative cost, find a feasible path $\sigma^* : [0,1] \to X_{\text{vis}}$ that solves the *path planning problem*, and minimizes the cost functional $c(\cdot)$.

The incremental approximation of optimal paths will play a key role in the proposed algorithm. We will discuss this in more detail in the next section.

Now let us explore the relationship between the solutions found in a given instance of the *path planning* problem. This relationship will be based on the visibility of the *guards* that 'own' each path. Let us first define how we compute this visibility.

Definition 2. (Local method) Let $\mathbb{L}$ denote a *local method* that computes a local path $\mathbb{L}(x,x')$ where x and x' are two configurations to be connected and furthermore, returns whether the resulting path is feasible. For kinematic systems, local paths can be represented as straight line paths in the configuration space.

Definition 3. (Visibility Domain [19]) Given a local method $\mathbb{L}$, the *visibility domain* of a configuration x is defined as $Vis_{\mathbb{L}}(x) = \{x' \in X_{\text{vis}} \text{ s.t. } \mathbb{L}(x,x') \subset X_{\text{vis}}\}$. [1]

[1] In this paper we assume local displacements in manifolds can be modeled by computing visibility between configurations via a local method. Extending this approach to cover more complex topologies is left for future work.

The visibility domain is rooted in configuration x, which is the *guard* configuration of $Vis_{\mathbb{L}}(x)$ [30]. A *guard* is not visible by any of the other guards in X_{vis}.

Guards and their *visibility domains* are essential components in the algorithm presented in the next section. We will use a *local method* to determine the visibility among guards and to move them around in the space. Consider Figure 2. Let the yellow and blue squares represent the start and goal states respectively. The gray object is an obstacle or hole, i.e., an *invisible region*, in the space.

Let us consider the three guards in this space. These are shown as white circles; each one 'owns' a path of a different color. In this paper, an 'owned' path is said to be the path of best cost that solves the path planning problem and contains the configuration of a particular guard.

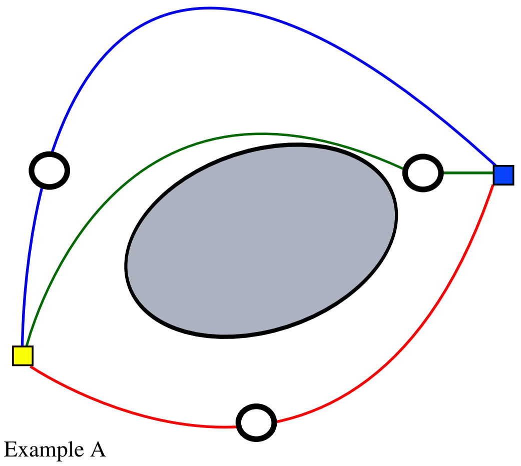


Fig. 2 Example A

Now consider the guards that own the blue and green paths. You will notice they are not able to 'see' each other. However, their current paths are similar, i.e., they both move above the hole to reach the goal. Intuitively speaking, these two guards can be thought to be 'guarding' the upper edge of the hole or obstacle in the center. Moreover, imagine you were to stretch or tighten these two paths into their shortest possible length. These would clearly merge or overlap above the top of the hole or obstacle. However, this is not the case for the guard on the red path. Like the other guards, it is not visible. However, the path it owns takes another route entirely.

Our goal will be to minimize the number of paths in a space while ensuring the ones we keep are not redundant (e.g., blue and green paths in Figure 2). Let us explore how to 'cover' the space with guards and the paths they own. First, we define a *guarded path*.

Definition 4. (Guarded path) Given a guard configuration x_{guard} in the configuration space X, a *guarded path* $g.\sigma : [0,1] \rightarrow X_{\text{vis}}$ is a feasible path in X, that starts from the initial configuration $g.\sigma(0) = x_{\text{init}}$ and reaches the goal configuration $g.\sigma(1) = x_{\text{goal}}$, such that $g.\sigma(\cdot) \in X_{\text{vis}}$, and contains the x_{g} configuration, i.e., $\exists \alpha \in [0,1]$ such that $\sigma(\alpha) = x_{\text{guard}}$.

Figure 2 shows three guarded paths. There is no location in this space that cannot be 'seen' by the guards in it. When this is the case, we consider the set of guards a solution to the *visible-space coverage* problem.

Problem 3. (Visible-space Coverage [19]) Given the configuration space X and a local method $\mathbb{L}$, find a *guard set* G such that the union of their visibility domains covers X_{vis}.

In this paper we will consider the optimal version of this problem. Imagine you wanted to reduce the number of guards in Figure 2. If a single guard centered above the object replaces the guards with the green and blue paths the space could be covered by two guards instead of three. Let us now consider this version of this problem.

Problem 4. (Minimal visible-space Coverage) Given the configuration space X and a local method $\mathbb{L}$, find the minimum guard set G^*, that is, a set of minimal cardinality, such that the union of their visibility domains covers X_{vis}.

The approach we present in this paper constructs approximate visibility domains using sampling-based algorithms. Therefore, the resulting problem could be thought of as a probabilistic version of the art gallery problem (more specifically minimal guard coverage) [30] or a special case of the minimum set cover [17] problem. Yet, in this paper we will focus more on the paths these guards are able to generate from their locations and less on how the guards themselves cover the space with their visibility. We can think of this component of the problem as a version of the minimum path cover problem [29].

From the literature we know that asymptotically optimal path planning algorithms are able to turn any path in a given space into the optimal one [14]. In the next section we will explain how to use this property to remove guards and paths that are unnecessary. However, we first need to describe what it means for paths to be optimal within their own visibility domains.

Definition 5. (Locally-optimal path) Given a guard configuration x_{guard} in the space X and its visibility-domain $Vis_{\mathbb{L}}(x_{\text{guard}})$, the *locally-optimal path* is the *guarded path* for the configuration in $Vis_{\mathbb{L}}(x_{\text{guard}})$ that provides the minimal cost $c(g.\sigma)$.

Problem 5. (Path Cover Planning) Given an initial configuration x_{init}, an invisible region X_{inv}, and a goal configuration x_{goal}, find a set of guard configurations G that solves the *visible-space coverage problem* and a *guarded path* for each guard.

Problem 6. (Minimum Path Cover Planning) Given the configuration space X, an invisible region X_{inv}, and the initial and goal configurations x_{init} and x_{goal}, find (i) the set of guard configurations G^* that solves the *minimal visible-space coverage problem* and (ii) a *locally-optimal path* for each guard such that the number of *guarded paths* is minimized.

4 Probabilistic Path Cover

In this section, we present an incremental sampling-based algorithm for path coverage of configuration spaces. The minimum path cover planning problem has an input of $(X, x_{\text{init}}, x_{\text{goal}}, c(\cdot), \mathbb{L}(\cdot))$ and returns an output of $F = (G, \Sigma_{\text{vis}})$, where $G = \{(x_{\text{root}}, T_{\text{vis}}, \sigma_{\text{best}})\}$, i.e., each guard in the set G is composed of a rooting configuration x_{root}, a tree T_{vis}, and the path that reaches x_{goal} with the lowest cost σ_{best}. The set Σ_{vis} contains all paths in F that reach x_{goal}. Recall the scenario shown in Figure 2. The location of each guard (white circle) represent x_{root}, the colored paths represent σ_{best}, and finally, T_{vis} (not rendered) represents an optimal tree that is constructed from this location and which is responsible for generating σ_{best}.

In the previous section we saw how simply using a standard sampling-based algorithm to cover the space with guards can result in suboptimal coverage of the space. We will now consider an algorithm that is able to merge similar paths into each other and uses this property to minimize the cardinality of the guard set.

4.1 Algorithmic Procedures

Let us now describe the primitives of the proposed algorithm.

SampleForest: The `SampleForest` procedure returns a guard g_{rand} randomly sampled from the set of guards in forest F where $g_{\text{rand}} = \{(x_{\text{root}}, T_{\text{vis}}, \sigma_{\text{best}})\}$.

Extend: Given a tree T_{rand} the `Extend` procedure performs one iteration of an asymptotically-optimal sampling-based algorithm, e.g., RRT*, k-RRT*, on $g_{\text{rand}}.T_{\text{vis}}$ and returns the latest vertex added to the tree x_{new} along with the updated tree, i.e., $T_{\text{rand}} = (V, E, \Sigma_{\text{vis}}, x_{\text{new}})$. A path is added to Σ_{vis} if $\mathbb{L}(x_{\text{new}}, x_{\text{goal}})$.

CoveringBalls [14]: Given a path $\sigma_n : [0,1] \to X$ and the real number r_n the `CoveringBalls` procedure returns the set $B_n = \{B_{n,1}, B_{n,2}, \ldots, B_{n,M_n}\}$ of M_n balls of connection radius r_n such that $B_{n,m}$ is centered at $\sigma(\tau_m)$ and the set collectively covers the path σ_n. The radius of the balls along the path is the connection radius of the RRT* algorithm, i.e., $r_n = \gamma F(\frac{\log|V|}{|V|})^{1/d}$ where $|V|$ is the number of vertices in the forest F. For details see Lemma 71 by Karaman and Frazzoli [14] and the Borel-Cantelli lemma [7].

RandomVertex: Given a forest F, the `RandomVertex` procedure randomly selects and returns a vertex from the set of trees.

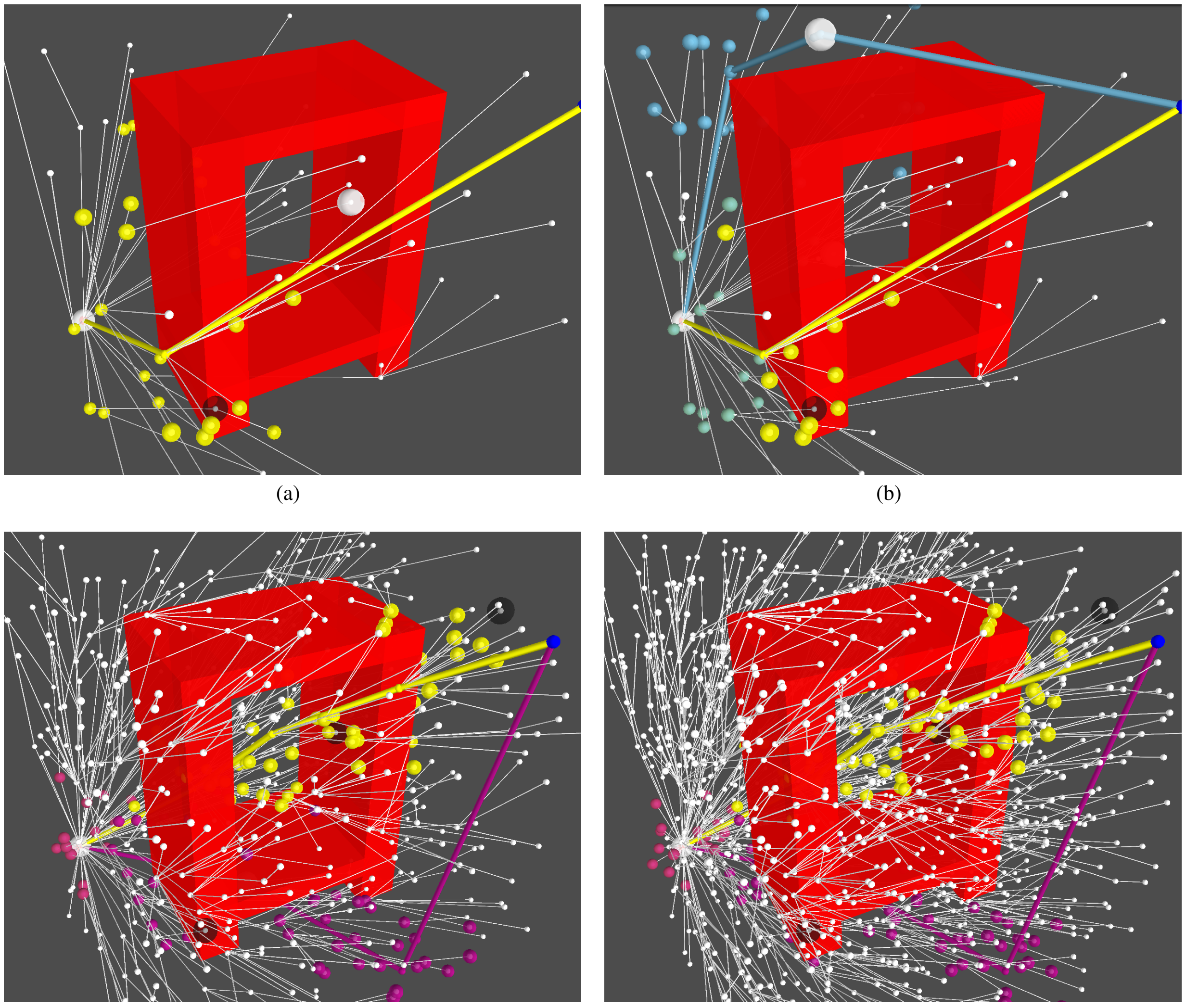

(a) (b) (c) (d)

Fig. 3 The progress in the Window scenario is shown. (a) Only one path has been found while a tree (not rendered) rooted at a guard (center white sphere) is constructed from inside the object. (b-d) solutions for both trajectories improve, reroot, and eventually overlap leaving the dominant paths. Intuitively speaking, imagine the paths in (a) and (b) as rubber bands that are being deformed to be as short as possible. The light-blue and yellow paths eventually overlap, creating a dominated guard node in the process. Finally, the dominating path becomes the purple solution and another tree constructed through the opening creates the yellow path (c,d).

Cost-to-go: The cost-to-go function serves as an equivalent to the admissible heuristic $h(x)$ employed by A* planning algorithms [9]. Given a vertex x, the `CostToGo` procedure returns the admissible global cost from that vertex, i.e., $\texttt{Cost}(x) + h(x)$ where $h(x)$ is the admissible cost incurred by the straight line path from x to the goal configuration. More specifically, $c(\sigma_a)$, where, $\sigma_a \leftarrow \mathbb{L}(x, x_{\text{goal}})$, and the feasibility of the path is not considered.

Dominates: Given two guard vertices g and g' the `Dominates` procedure determines whether g dominates g'. Let $g.\sigma_{best}$ and $g'.\sigma_{best}$ be the paths of lowest cost owned by guards g and g' respectively, then g dominates g' if $c(g.\sigma_{best}) < c(g'.\sigma_{best})$.

Guard: The `Guard` procedure verifies the visibility of guard candidate x_{new} from all guards in G and G', i.e., $\mathbb{L}(x_{\text{new}}, g)$ for all $g \in G \cup G'$. The x_{new} configuration is considered a guard if it is not visible from any of the guards. The guard procedure considers new configurations only when each existing guard in G owns a solution, i.e., $g.\sigma_{\text{best}} \neq \emptyset$ for all $g \in G$.

4.2 Proposed Algorithm

The proposed algorithm incrementally constructs a forest of trees rooted at guard configurations. Vertices are added to the set of guards if they are unreachable by any of the guards in the current set. Each tree asymptotically approximates the optimal path from its root to x_{goal}. These are rerooted and reconstructed when configurations that provide better path cost with the same visibility are discovered. Moreover, redundant guards, i.e, those that are found to be in the covering balls of a guarded path, are removed from the set.

The procedure is presented in Algorithm 1. The algorithm is initialized with a forest F that contains a single initial tree rooted at the guard configuration of a vertex representing the initial state. The procedure iterates by sampling a random guards g_{rand} from F (Line 4), and performing one step of optimal tree construction on its tree (Line 5). At this point, the visibility of x_{new}, the latest vertex added to F, is tested against the guards in G and G' (Line 6). If x_{new} is not visible by any of the guards, it is added to G and a tree is initialized at its vertex (Lines 7-8).

In the second phase, the algorithm attempts to move g_{rand} with the `Reroot` procedure (Line 9, Algorithm 2). The `RandomVertex` procedure returns a random configuration x_{rand} from F (Line 2). The guard is rerooted at x_{rand} if the following two conditions are met: The admissible cost of x_{rand} is lower than that of g_{rand} (Line 3) and the sampled configuration is visible only by the current guard configuration (Lines 5, 7). In this case, the guard starts constructing a new tree at this new configuration. Note that the best solution found by the previous tree is kept (Line 8) and is only replaced if one of lower cost is found.

In the final phase, the algorithm attempts to remove redundant guards with the `GuardDominance` procedure (Line 10, Algorithm 3). If g_{rand} owns a path, the algorithm samples a random guard g'. Then, it computes the set of `CoveringBalls` for the path owned by g_{rand} (Line 3) and verifies if g' is contained in any of the balls (Lines 4-5). In this event, the guard of best solution cost is removed from G, added to the set of dominated guards G', and its tree is removed from F (Lines 6-9).

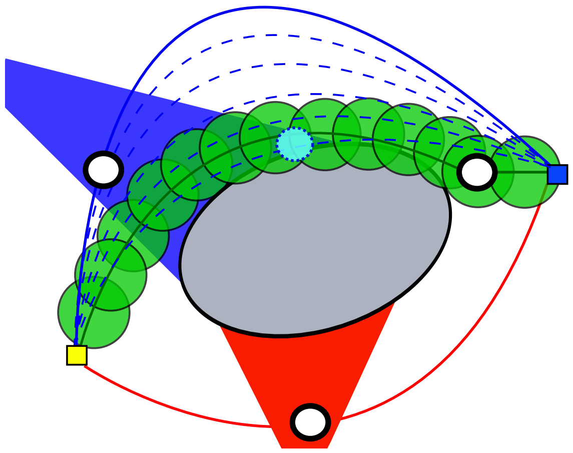


Fig. 4 Example B

Consider again the example in the previous section. Let us now focus on the visibility-domains owned by the guards and the *covering balls* for the guarded paths. The space of possible rerooting locations within the visibility-domains of the blue and red guards are depicted as regions of that color in Figure 4. Note that if the blue guard moves outside this region, it will be seen by either the red or green guards.

As the algorithm iterates, the blue guard will move to configurations that provide guarded paths of lower cost. An optimal configuration for this guard is rendered as the blue circle with the dashed edge. Additionally, the guarded path that becomes possible at this configuration along with paths of increasingly less cost are shown as dashed blue lines. Now, notice how the resulting locally-optimal path for the blue guard lies inside the covering balls of the green guard. When this event takes place, the guard with the path of highest cost loses its path (and tree) and is added to the dominated guard set. This prevents another guard from appearing in the same region.

Algorithm 1: $(X, x_{\text{init}}, x_{\text{goal}}, c(\cdot), \mathbb{L}(\cdot))$

```
1  $V \leftarrow \{x_{\text{init}}\}; E \leftarrow \emptyset; G' \leftarrow \emptyset; \Sigma_{\text{vis}} \leftarrow \emptyset;$ ;
2  $T_{\text{main}} \leftarrow (V, E, \Sigma_{\text{vis}}); F \leftarrow \{T_{\text{main}}\}; G \leftarrow \{(x_{\text{init}}, T_{\text{main}}, \emptyset)\};$
3  for $i = 1 \dots n$ do
4      $g_{\text{rand}} \leftarrow$ SampleForest($F$);
5      $(V, E, \Sigma_{\text{vis}}, x_{\text{new}}) \leftarrow$ Extend($g_{\text{rand}}.T_{\text{vis}}$);
6      if Guard($G, G', x_{\text{new}}$) then
7          $T_{\text{new}} \leftarrow \{V = \{x_{\text{new}}\}, E = \emptyset, \Sigma = \emptyset\}; F \leftarrow F \cup \{T_{\text{new}}\};$
8          $G \leftarrow G \cup \{(x_{\text{new}}, T_{\text{new}}, \sigma_{\text{best}} = \emptyset)\};$
9      $(F, G) \leftarrow$ Reroot($F, G, g_{\text{rand}}$);
10     $(F, G, G') \leftarrow$ GuardDominance($F, G, G', g_{\text{rand}}$);
11 return $F = (G, \Sigma_{\text{vis}})$.
```

Algorithm 2: Reroot (F, G, g_{rand})

```
1  if $g_{\text{rand}}.\sigma_{\text{best}} \neq \emptyset$ then
2      $x_{\text{rand}} \leftarrow$ RandomVertex($F$);
3      if CostToGo($x_{\text{rand}}$) $<$ CostToGo($g_{\text{rand}}$) then
4          foreach $g \in G \backslash \{g_{\text{rand}}\}$ do
5              if $\mathbb{L}(g, x_{\text{rand}})$ then
6                  return $(F, G, \Sigma_{\text{vis}})$
7          if $\mathbb{L}(x_{\text{rand}}, g_{\text{rand}})$ then
8              $g.T_{\text{vis}} \leftarrow (V = \{x_{\text{rand}}\}, E = \emptyset, \Sigma_{\text{new}} = \{g_{\text{rand}}.\sigma_{\text{best}}\});$
9              $F \leftarrow F \backslash \{g_{\text{rand}}.T\}; F \leftarrow F \cup \{T_{\text{new}}\};$
10 return $(F, G)$
```

Algorithm 3: GuardDominance$(F, G, G', g_{\text{rand}})$

```
1  if $g_{\text{rand}}.\sigma_{\text{best}} \neq \emptyset$ then
2      $g' \leftarrow$ SampleForest($F$);
3      $B_n \leftarrow$ CoveringBalls($g_{\text{rand}}.\sigma_{best}, \gamma F(\frac{\log|V|}{|V|})^{1/d}$);
4      foreach $b \in B_n$ do
5          if $g'$ in $b$ then
6              if Dominates($g_{\text{rand}}, g'$) then
7                  $F \leftarrow F \backslash \{g'.T\}; G' \leftarrow G' \cup \{g'\}; G \leftarrow G \backslash \{g'\};$
8              else
9                  $F \leftarrow F \backslash \{g.T\}; G' \leftarrow G' \cup \{g_{\text{rand}}\}; G \leftarrow G \backslash \{g_{\text{rand}}\};$
10 return $(F, G, G')$
```

Finally, consider the red guard. There is no possible rerooting location which will put its guard inside the covering balls of another guarded path in the space. The algorithm continues to iterate in this manner and eventually covers the space with guards (both dominated and dominating) and approximates the locally-optimal path for each remaining guard.

4.3 Computational Complexity

In this section, we consider the computational complexity of the proposed algorithm.

There are two main procedures invoked during graph construction. These are, *collision checking*, and *near* or range search. It is known from the literature that the expected number of calls to a collision-checking procedure during the construction of an optimal tree using an algorithm such as RRT*, or k-RRT*, requires time $k \log n$ for all n with $n/\log n > k$ where n is the number of iterations [14]. Moreover, the *near* computation, or the calculation of the points

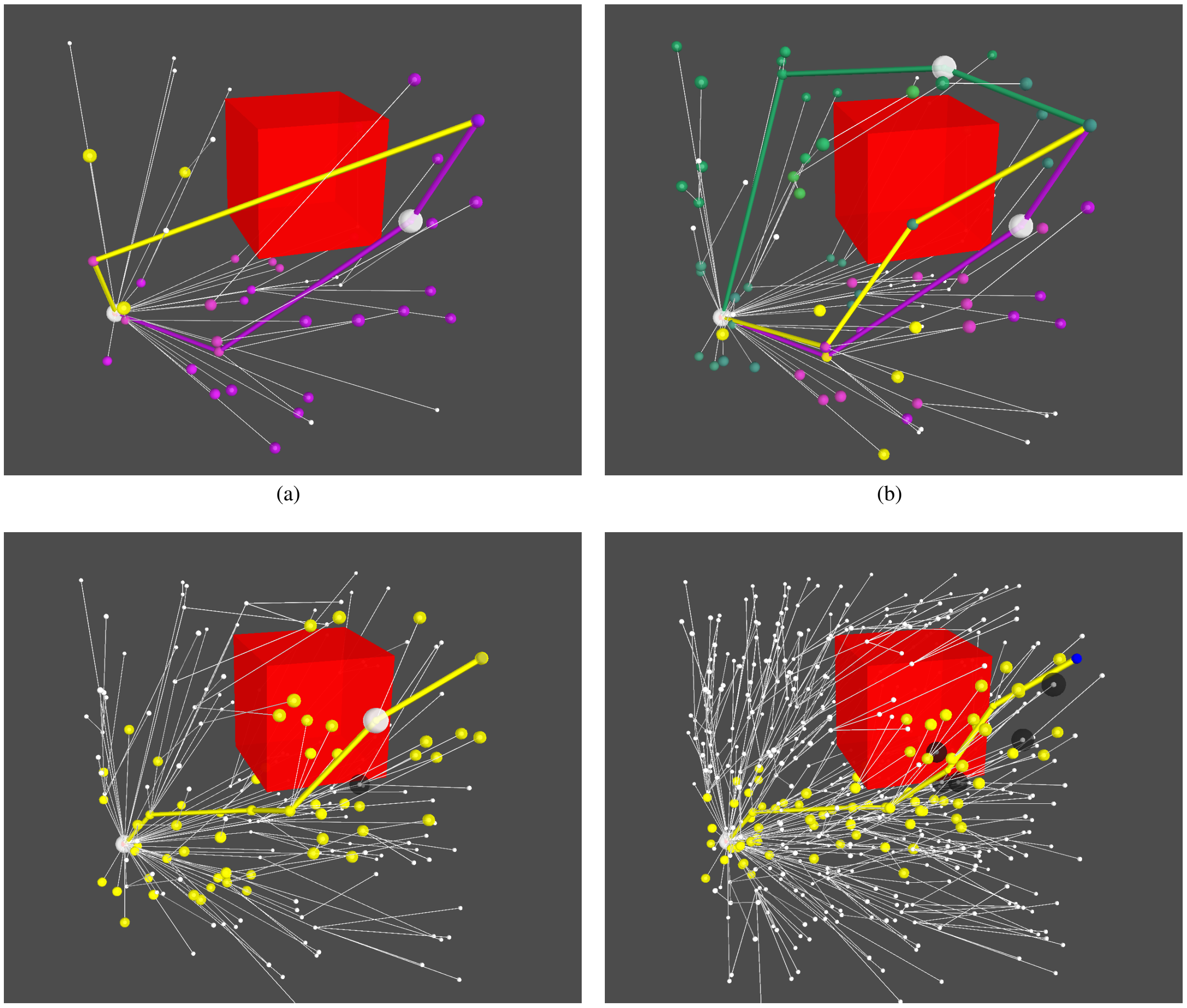
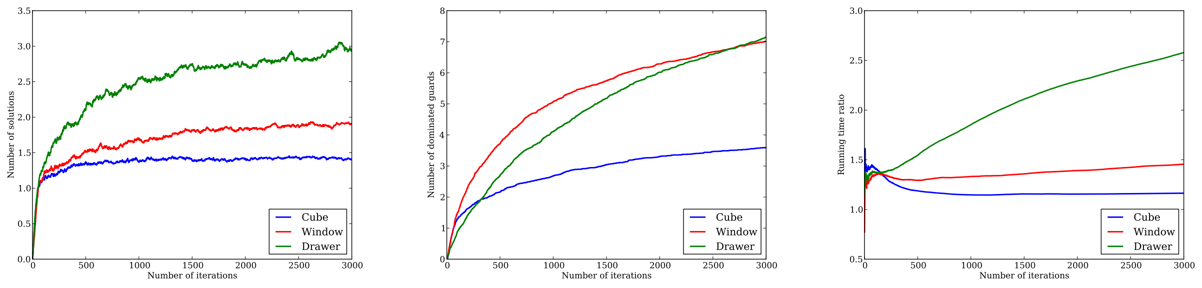

(a) (b) (c) (d)

Fig. 5 The Cube scenario only has one local optimum. However, because the algorithm iterates in an anytime manner, several potential guards are considered initially (a-b). Guards are shown as white spheres, dominated guards as black spheres, and locally-optimal paths as colored paths. In (a) a guard candidate is found behind the cube. (b) three guards with a solution each are found. (c) These paths eventually overlap and are dominated by the one of lowest cost (yellow). (d) The remaining path is optimized, dominated guards from previous paths maintain other trees from being created.

in some range, e.g., inside a ball of a specific radius, has a worst-case time of complexity $O(n^{1-1/d}+m)$ where m is the number of returned points [22]. More formally, at the algorithmic level, the time complexity of computing each tree is $O(n\log n)$ and its space complexity is $O(n)$.

The algorithm presented in this paper constructs multiple trees simultaneously. However, because most operations are done lazily, and only a single tree is considered at each iteration, the computational burden is comparable to that of single-tree optimal planners. In fact, the algorithm incurs additional computation at each iteration solely to calculate the overlap of covering balls and the visibility between guards. In the next section we empirically compare the performance of the proposed to algorithm to a single-tree optimal planner. We leave the theoretical characterization of the increased expected time and space complexity for future work.

5 Experimental Results



(a) (b) (c)

Fig. 6 The proposed algorithm is run 200 times for 3000 iterations in all three environments.

In this section we present and discuss some experimental results. The proposed algorithm was implemented in Python and tested on a computer with 4GB RAM and a 1.80 GHz x 4 processor. The focus of the experiments is to illustrate the ability of the algorithm to find a minimal set of guards with locally-optimal paths. We consider three problem instances: the *cube* scenario has a single cube shaped obstacle (Figure 5), the *window* scenario has a hollow obstacle (Figure 3), and the *drawer* scenario has a hollow obstacle with two separate openings (Figure 1). The number of expected paths for each scenario are one, two, and three respectively.

Figure 6 depicts the average performance of the algorithm in terms of number of solutions, number of dominated guards, and computational ratio over a single-tree RRT*, averaged over 200 runs. The three scenarios are shown as blue, red, and green lines for the Cube, Window, and Drawer environments respectively. Figure 6 (a) shows the average number of solutions/guards at every iteration. These values converge to the expected one, two, and three for the Cube, Window, and Drawer scenarios respectively. In (b), the average number of dominated guards is plotted against iterations for all three cases. (c) Shows the ratio of the running time of the proposed algorithm over a standard single-tree RRT*. It can be seen to converge to less than 1.5 in simple scenarios. In the Drawer scenario, it was observed that the algorithm took longer to fill the space with dominated guards and therefore continues to incur extra computation.

Cube

Alg.	It. t (s)	Fn. t (s)	NNs	V	CCs
RRT*	0.014	43.01	5408.34	2546.12	24202.26
Proposed	0.017	50.06	7853.94	2140.55	80946.47

Window

Alg.	It. t (s)	Fn. t (s)	NNs	V	CCs
RRT*	0.013	37.76	5198.13	2436.99	27247.50
Proposed	0.018	54.94	11087.64	1544.84	88405.68

Drawer

Alg.	It. t (s)	Fn. t (s)	NNs	V	CCs
RRT*	0.013	37.95	5141.36	2405.12	27584.65
Proposed	0.033	97.85	22264.46	1253.95	88288.97

Table 1

Table 1 summarizes the performance of the proposed algorithm and a standard single-tree RRT* in the three environments. The columns depict iteration time (It. t), completion time (Fn. t), number of nearest neighbor queries (NNs), number of vertices in the graph at completion (V), and number of collision-checking queries (CCs) averaged over 200 runs of 3000 iterations.

6 Discussion and Future Work

It is essential to theoretically characterize the complexity, correctness, and optimality of the approach. In general terms, the algorithm leverages theoretical properties of asymptotically-optimal sampling-based algorithms, i.e., RRT*, k-RRT*, RRG, to deform paths into each other and to remove redundant guards (see Karaman and Frazzoli [14] and the Borel-Cantelli lemma [7]). However, a detailed theoretical analysis of the overall approach is left for future work. Similarly, although the computational complexity of several related problems are in the literature, i.e., set cover [17], art gallery [30], path cover [29], and counting problems [36], the complexity of the problem presented in this paper is not discussed.

The properties of the paths computed by the approach also require theoretical analysis. The paths clearly differ in visibility, as this is directly computed and considered by the algorithm. However, it is unclear what topological relationship the resulting path classes might have (e.g., are all paths non-homotopic?) [27]. Indeed, the analysis of more complicated manifolds along with the resulting solutions generated by the approach are left for future work.

Moreover, given that the algorithm uses a simple *local method* to generate and remove paths, the approach can be applied to high-dimensional motion planning problems, e.g., computing multiple, distinct paths for arm manipulation with a PR2 or Baxter robot [31], and to instances of the motion planning problem where the platform has complicated dynamics that might result in different solution classes for reasons other than collisions with obstacles in the environment, e.g., underactuated systems [28], nonholonomic vehicles [15], belief space planning [32], kinodynamic planning for specific time horizons [8], and chance-constrained path planning [25]. Application to other domains should be straightforward.

References

[1] P. Agarwal. “Compact Representations for Shortest-Path Queries”. In: *IROS Workshop on Progress and Open Problems in Motion Planning*. 2011.

[2] A. Bhatia and E. Frazzoli. “Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems”. In: *Hybrid Systems: Computation and Control*. Ed. by R. Alur and G. Pappas. Lecture Notes in Computer Science 2993. 2004, pp. 451–471.

[3] S. Bhattacharya, V. Kumar, and M. Likhachev. “Search-based path planning with homotopy class constraints”. In: *In Proc. National Conference on Artificial Intelligence*.

[4] S. Bhattacharya, M. Likhachev, and V. Kumar. *Identification and Representation of Homotopy Classes of Trajectories for Search-based Path Planning in 3D*.

[5] O. Brock and O. Khatib. “Elastic Strips: A Framework for Motion Generation in Human Environments”. In: *The International Journal of Robotics Research* 21.12 (2002), pp. 1031–1052.

[6] A. Dobson, A. Krontiris, and K. E. Bekris. “Sparse Roadmap Spanners”. In: *Workshop on the Algorithmic Foundations of Robotics (WAFR)*. 2012.

[7] D. S. G. Grimmett. *Probability and Random Processes*. Third. Oxford University Press, 2001.

[8] G. Goretkin et al. “Optimal Sampling-Based Planning for Linear-Quadratic Kinodynamic Systems”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013.

[9] P. E. Hart, N. J. Nilsson, and B. Raphael. “A formal basis for the heuristic determination of minimum cost paths”. In: *IEEE Transactions on Systems, Science, and Cybernetics* SSC-4.2 (1968), pp. 100–107.

[10] K. Hauser. “The Minimum Constraint Removal Problem with Three Robotics Applications”. In: *WAFR*. 2012, pp. 1–17.

[11] L. Jaillet et al. “Randomized tree construction algorithm to explore energy landscapes”. In: *Journal of Computational Chemistry* (2011), pp. 3464–3474.

[12] L. Jaillet and T. Simon. “Path deformation roadmaps”. In: *in Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*. 2006.

[13] M. Kallmann and M. J. Matarić. “Motion Planning Using Dynamic Roadmaps”. In: *International Conference on Robotics and Automation*. 2004, pp. 4399–4404.

[14] S. Karaman and E. Frazzoli. “Sampling-based Algorithms for Optimal Motion Planning”. In: *International Journal of Robotics Research* (2011).

[15] S. Karaman et al. “Anytime Motion Planning using the RRT*”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011.

[16] L. Kavraki et al. “Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 1996, pp. 566–580.

[17] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. 4th. Springer Publishing Company, Incorporated, 2007. ISBN: 3540718435, 9783540718437.

[18] J. Latombe. “Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts”. In: *Int’l J. of Robotics Research* 18.11 (1999), pp. 1119–1128.

[19] J p. Laumond and C. Nissoux. “Visibility-based probabilistic roadmaps for motion planning”. In: *Journal of Advanced Robotics* 14 (2000), p. 2000.

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

[21] S. M. LaValle and J. J. Kuffner. “Randomized Kinodynamic Planning”. In: *International Journal of Robotics Research* 20.5 (2001), pp. 378–400.

[22] D. Lee and C. Wong. “Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees”. English. In: *Acta Informatica* 9.1 (1977), pp. 23–29. ISSN: 0001-5903.

[23] P. Leven and S. Hutchinson. “A Framework for Real-time Path Planning in Changing Environments.” In: *I. J. Robotic Res.* 21.12 (2002), pp. 999–1030.

[24] Y. Liu and N. Badler. “Real-time reach planning for animated characters using hardware acceleration”. In: *IEEE Int’l Conf. on Computer Animation and Social Characters*. 2003, pp. 86–93.

[25] B. D. Luders, S. Karaman, and J. P. How. “Robust Sampling-based Motion Planning with Asymptotic Optimality Guarantees”. In: *AIAA Guidance, Navigation, and Control Conference (GNC)*. Boston, MA, 2013.

[26] R. Luna and K. E. Bekris. “Network-guided multi-robot path planning in discrete representations”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan*. IEEE, 2010, pp. 4596–4602. ISBN: 978-1-4244-6674-0.

[27] J. Munkres. *Topology*. Topology. Prentice Hall, Incorporated, 2000. ISBN: 9780131784499.

[28] R. Murray and J. Hauser. *A Case Study in Approximate Linearization: The Acrobot Example*. Tech. rep. UCB/ERL M91/46. EECS Department, University of California, Berkeley, 1991.

[29] S. C. Ntafos and S. L. Hakimi. “On Path Cover Problems in Digraphs and Applications to Program Testing”. In: *IEEE Trans. Software Eng.* 5.5 (1979), pp. 520–529.

[30] J. O’Rourke. *Art gallery theorems and algorithms*. New York, NY, USA: Oxford University Press, Inc., 1987. ISBN: 0-19-503965-3.

[31] A. Perez et al. “Asymptotically-optimal Manipulation Planning using Incremental Sampling-based Algorithms”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2011.

[32] A. Perez et al. “LQR-RRT*: Optimal Sampling-Based Motion Planning with Automatically Derived Extension Heuristics”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2012.

[33] E. Plaku. “Planning in Discrete and Continuous Spaces: From LTL Tasks to Robot Motions”. In: *Advances in Autonomous Robotics*. Ed. by G. Herrmann et al. Vol. 7429. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 331–342. ISBN: 978-3-642-32526-7.

[34] E. Schmitzberger et al. “Capture of homotopy classes with probabilistic road map”. In: *IROS*. 2002, pp. 2317–2322.

[35] J. T. Schwartz and M. Sharir. “On the ‘piano movers’ problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds”. In: *Advances in Applied Mathematics* 4 (1983), pp. 298–351.

[36] L. G. Valiant. “The Complexity of Computing the Permanent”. In: *Theoretical Computer Science* 8 (1979), pp. 189–201.