Motion Planning for Multi-Link Robots by Implicit Configuration-Space Tiling

Oren Salzman  Kiril Solovey  Dan Halperin

Abstract—We study the problem of motion-planning for free-flying multi-link robots and develop a sampling-based algorithm that is specifically tailored for this type of robots. Our work is based on the simple observation that the set of configurations for which the robot is self-collision free is independent of the obstacles or of the exact placement of the robot. This allows us to eliminate the need to perform costly self-collision checks online when solving motion-planning problems, assuming some offline preprocessing. In particular, given a specific robot our algorithm precomputes a tiling roadmap, which efficiently and implicitly encodes the self-collision free (sub-)space over the entire configuration space, where the latter can be infinite for that matter. To answer any query, in any given scenario, we traverse the tiling roadmap while only testing for collisions with obstacles. Our algorithm suggests more flexibility than the prevailing paradigm in which a precomputed roadmap depends both on the robot and on the scenario at hand. We show through various simulations the effectiveness of this approach. Specifically, in certain settings our algorithm is ten times faster than the RRT algorithm.

I. INTRODUCTION

Motion planning is a fundamental problem in robotics. In its most basic form, it is concerned with moving a robot from start to goal while avoiding collisions with obstacles. Initial efforts in motion planning have focused on designing complete analytical algorithms (see, e.g., [34]), which aim to construct an explicit representation of the free space—the set of collision-free configurations. However, with the realization that such approaches are computationally intractable [32], even for relatively-simple settings, the interest of the robotics community has gradually shifted to sampling-based techniques for motion planning [3] [15]. Such techniques attempt to capture the connectivity of the free space by random sampling. Sampling-based algorithms are conceptually simple, easy to implement, and remarkably efficient in practical settings. As such, they are widely used in practice. Another key advantage of these techniques is the fact that they are typically described in general terms and can often be applied to a wide range of robots and scenarios. However, this also has its downsides. Due to the limited reliance of sampling-based algorithm on the specific structure of the problem at hand, they tend to overlook unique aspects of the problem, which might be exploited to increase the efficiency of such methods. For instance, a more careful analysis of the specific problem may result in a reduced reliance on collision detection, which is often considered to be the bottleneck component in sampling-based algorithms.

In this paper we study the problem of planning the motion of a multi-link robot that consist of multiple rigid links connected by a set of joints (see Fig. 1). We assume that the robot is free flying, i.e., none of its joints is anchored to a specific point in the workspace. We describe a novel algorithm which exploits the unique structure of the problem. Our work is based on the simple observation that the set of configurations for which the robot is not in self collision is independent of the obstacles or on the exact placement of the robot. This allows us to develop an algorithm which eliminates the need to perform self-collision checks in the query stage. In particular, self-collision checks are only performed in a preprocessing stage, while the actual planning solely requires obstacle-collision checks. The novelty comes from the fact that preprocessing needs to be made once for a given type of robot. This is in contrast with prevalent state-of-the-art techniques, such as PRM [9] or PRM* [8], where the preprocessed structure can only be applied to a particular scenario and robot type. In some situations, the cost of self-collision checks can be as high as the cost of obstacle-collision checks—particularly in cases where the robot consists of many links.

The novelty of our approach lies in an implicit representation of a roadmap, termed the tiling roadmap, which efficiently represents the space of configurations which are self-collision free and is completely independent of the scenarios in which it can be employed. Once a query is given in the form of a scenario—a description of the workspace obstacles, a source configuration and a target region, the tiling roadmap is traversed using the recently introduced dRRT pathfinding algorithm [33]. We do not make any particular assumption regarding the shape of the links or the type of joints. While our current work deals with free-flying multi-link robots, we hope that it will pave the way to the development of similar techniques for various types of robots. This will have immediate practical implications: when developing a robot for mass production, a preprocessed structure, similar to the tiling roadmap, would be embedded directly to the hardware of the robot. This would eliminate the need to consider self-collisions when dealing with complex robots such as the PR2 robot.

We demonstrate the effectiveness of our technique for the

1Our algorithm may be applied straightforwardly to anchored robots but the speed-up that may be obtained for such cases is less profound. We envision, however, more efficient implementations of our framework for such cases. See evaluation in Section VII and discussion in Section VIII.

2PR2 robot: http://www.willowgarage.com/pages/pr2/overview

This work has been supported in part by the Israel Science Foundation (grant no. 1102/11), by the German-Israeli Foundation (grant no. 1150-82.6/2011), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.
II. RELATED WORK

Besides the obvious relevance to the design and use of industrial robots, motion planning for multi-link robots has applications in diverse domains such as protein folding [39] and reconfigurable robots [11, 24]. For a general overview of the subject and additional applications the reader is referred to [15].

A common approach to plan the motion of multi-link robots is by sampling-based algorithms [3, 15]. While sampling-based planners such as PRM [9] and RRT [16] may be used for some types of multi-link robots, they are not suited for planning when the robot is constrained [43]. For example, when the links form a closed chain, the probability of uniformly sampling a valid configuration approaches zero [43]. Thus, the recent years have seen many works attempting to sample valid configurations and to compute local paths for a variety of multi-link robots differing in the dimension of the workspace, the type of joints and the constraints on the system [6, 7, 22, 40, 43, 44].

Particularly noteworthy is the notion of reachable volumes which was recently introduced [20, 21]. The reachable volume of a multi-link system is the set of points that the end effector of the system can reach. The authors show how to compute the reachable volume and present a method for generating self-collision free configurations using reachable volumes. This method is applicable to open and closed chain robots, tree-like robots, and robots that include both loops and branches. Pan et al. [26] introduced a motion-planning
algorithm for articulated models such as multi-link robots, which is integrated in an RRT-like framework.

Our work shares similarities with the work by Han and Amato [7], who studied closed chain systems. They proposed a two-stage kinematics-based probabilistic roadmap (KBPRM) planner. In the first stage, they build a PRM roadmap that ignores the obstacles and only considers the robot’s kinematics. In the second stage, copies of the roadmap are populated and connections are made between nodes with specific characteristics. The first stage of KBPRM relies on a partitioning of the closed chain into open chains. Thus, in [42] a fully automated method is proposed for partitioning an arbitrary linkage into open chains.

A key element in the implementation of sampling-based motion-planning algorithms is collision detection. As such there have been application-specific collision-detection algorithms for articulated robots in general [35], and for multi-link robots in particular [4, 19, 31].

Sampling-based algorithms are not the only tool used to study the problem at hand. There have been attempts to study the structure of the configuration space of multi-link robots (see, e.g., [25, 36] or to explicitly construct it (see, e.g., [18, 41, 37]). Space-decomposition techniques were used to approximate the structure of the configuration space [28, 29] and efficient graph-search algorithms were used to search in a configuration space that was discretized using a grid [17].

For protein chains, which are typically modelled as high-dimensional tree-shaped multi-link robots, sampling-based approaches have been used together with an energy function which guides the search in space (see, e.g., [1, 14, 30]).

III. ALGORITHM OVERVIEW AND PRELIMINARIES

We start with a brief overview of our algorithm and continue with a set of definitions required to formally describe the tiling roadmap.

The tiling roadmap $\mathcal{G}$ is a preprocessed graph that efficiently represents the space of configurations which are self-collision free. To generate $\mathcal{G}$, we first generate a collection of self-collision free configurations $\mathcal{C}$ called base configurations which in turn induce a collection of base roadmaps. These base roadmaps, which are explicitly constructed, describe a set of movements and configurations in which the robot is anchored at one of its links’ endpoints to the origin. We refer to all such endpoints as anchor points. They are then used to implicitly define $\mathcal{G}$. The approach is sufficiently general in terms of methods to sample the base configurations or to connect close-by configurations. Furthermore, the base roadmaps may be generated using any sampling-based planner that constructs a roadmap. Given a query, the tiling roadmap $\mathcal{G}$ is traversed using the dRRT [38] pathfinding algorithm (see Section V). Only when a configuration or an edge is considered by the pathfinding algorithm, is it checked for intersection with the obstacles.

We now proceed with a set of definitions that will be used in the rest of the paper. Let $R$ be a multi-link robot moving in some workspace $\mathcal{W} \subseteq \mathbb{R}^d$ (where $d \in \{2, 3\}$) cluttered with a set $\mathcal{O}$ of obstacles. The robot $R$ consists of several rigid links and $m$ anchor points. To simplify the discussion we consider robots which are “snake-like”, i.e., each anchor point connects at most two rotating links and there are no loops. We stress that the technique remains correct for any other type of a free-flying multi-link robot. While a configuration of such a robot is usually represented by the position $p \in \mathcal{W}$ of its reference point and the angles of the joints, it will be much simpler to describe our technique while representing a configuration by a collection of $m$ points in $\mathbb{R}^d$, which describe the coordinates of the anchor points. Thus, we define the configuration space of the robot to be

$$\mathcal{C} := \{(p_1, \ldots, p_m) \mid p_i \in \mathbb{R}^d\},$$

such that the lengths of the links are not altered. Given an index $1 \leq j \leq m$ and a point $q \in \mathbb{R}^d$ we denote by

$$\mathcal{C}_j(q) := \{(p_1, \ldots, p_m) \mid p_i \in \mathbb{R}^d, p_j = q\},$$

the set of configurations in which the $j$’th anchor point is anchored at the point $q$.

We denote the obstacle-collision free space by $\mathcal{F}^O \subset \mathcal{C}$, which is the set of configurations in which the robot does not collide with any obstacle. In addition, we denote the self-collision free space by $\mathcal{F}^S \subset \mathcal{C}$, which is the set of configurations in which no two links of the robot intersect. Finally, we set $\mathcal{F} := \mathcal{F}^O \cap \mathcal{F}^S$ and refer to this set as the free space. In a similar fashion we define these sets for the case where the $j$’th anchor point of the robot is fixed at $q \in \mathbb{R}^d$, i.e., $\mathcal{F}^O(q), \mathcal{F}^S(q), \mathcal{F}_j(q) \subset \mathcal{C}_j(q)$.

Given a configuration $C = (c_1, \ldots, c_m)$ and a point $p \in \mathbb{R}^d$, let $C + p$ denote the configuration $(c_1 + p, \ldots, c_m + p)$. Namely, $C + p$ is the configuration obtained by computing a vector sum of each anchor point with the vector $p$. We say that $C + p$ is the configuration $C$ translated by $p$. Additionally, for a configuration $C$, as defined above, and an index $1 \leq j \leq m$, let $j(C) := c_j$. Namely, $j(C)$ denotes the location of the $j$’th anchor point of configuration $C$. Finally, let $0$ denote the origin of $\mathbb{R}^d$.

IV. MOTION PLANNING BY TILING ROADMAPS

In this section we formally define the tiling roadmap $\mathcal{G}$. We first describe how, given a set $\mathcal{C}$ of self-collision free base configurations, we define a collection of base roadmaps. We then explain how the base roadmaps are used to construct $\mathcal{G}$. Finally, we describe how the tiling roadmap $\mathcal{G}$ is used to answer motion-planning queries.

A. Base roadmaps for the anchored robot

Let $\mathcal{C} := \{C_1, \ldots, C_n\} \subset \mathcal{F}^S$ be a collection of $n$ self-collision free configurations called base configurations. For each configuration $C_i \in \mathcal{C}$, and for each index $j$, we consider the configuration $C_{j,i} := C_i - j(C_i)$ which represents the configuration $C_i$ translated by $-j(C_i)$. Clearly, the $j$’th anchor point is not to be confused with the robot’s joints. A snake-like robot with $m$ links has $m$ joints but $m + 1$ anchor points.
anchor point of \( C_{j,i} \) coincides with the origin. Now, set \( C_j := \{ C_{j,1}, \ldots, C_{j,n} \} \) and note that \( C_j \subset \mathcal{F}^S(0) \).

For every index \( j \) we construct the base roadmap \( G_j(0) = (V_j(0), E_j(0)) \), where \( V_j(0) = \mathcal{C}_j \). We compute for each anchored configuration \( C_{j,i} \in \mathcal{C}_j \) a set of \( k \) nearest neighbors. We add an edge \( E_j(0) \) between \( C_{j,i} \) and each of its neighbors if the local path connecting the two is in \( \mathcal{F}^S_j(0) \). We denote this path by \( \pi_j(C_{i_1}, C_{i_2}) \) and by \( \text{nbr}(C_{j,i}, G_j(0)) \) the set of neighbors of the configuration \( C_{j,i} \) in the base roadmap \( G_j(0) \). Given \( p \in \mathbb{R}^d \), we use the notation \( G_j(p) = (V_j(p), E_j(p)) \) to represent the roadmap \( G_j(0) \) translated by \( p \). See Fig. 2.

**Observation 1:** For every base configuration \( C_i \in \mathcal{C} \) and anchor point \( p_j \), \( C_{j,i} = C_i - j(C_i) \in V_j(0) \). Thus, for every point \( p \in \mathbb{R}^d \), \( C_{j,i} + p \in V_j(p) \).

**Observation 2:** Let \( C \) be a vertex of \( G_j(0) \) for some index \( j \). Then \( C \) is also a vertex of \( G_j(j'(C)) \) (the roadmap fixed at the \( j \)'th anchor point of \( C \)) for any index \( j' \neq j \). Similarly, if \( C \) is a vertex of \( G_j(p) \) for some point \( p \in \mathbb{R}^d \), then it is also a vertex of \( G_j'(p + j'(C)) \).

**B. Implicitly-represented tiling roadmaps**

Given a point \( p \in \mathbb{R}^d \), the set \( \mathcal{C} \subset \mathcal{F}^S \) of \( n \) base configurations, and the explicitly constructed base roadmaps \( G_1(0), \ldots, G_n(0) \), we implicitly define the tiling roadmap \( \mathcal{G} = (V, \mathcal{E}) \). The set of vertices \( V \) is defined recursively as follows: We start by setting \( P = \{ p_0 \} \) and adding all configuration \( C_{j,i}(p) \) as vertices to \( V \) for each \( p \in P \), for each base configuration \( C_i \in \mathcal{C} \) and for each index \( j \). Define \( P_j = \{ j(C) \mid C \in \mathcal{V} \} \). Namely, \( P_j \) is the set of all locations of the \( j \)'th anchor point for each vertex \( C \in \mathcal{V} \). We now set \( P = \bigcup_{j=1}^{n} P_j \) and repeat the process. Finally, let \( C, C' \in \mathcal{V} \). Then \( (C, C') \in \mathcal{E} \) if there exist \( j, p \) such that \( C, C' \in V_j(p) \), and \((C', C) \in E_j(p) \).

**Lemma 3:** If \( C \in \mathcal{V} \) then for every index \( j \) there exists some point \( p_j \in \mathbb{R}^d \) such that \( C \in G_j(p_j) \).

The proof of this lemma follows from Observation 2.

To visualize the recursive definition of the tiling roadmap we examine the simple (self-collision free) case of a robot with one link that was preprocessed with \( n = 12 \) base configurations. Furthermore, assume that for all base configurations, the link’s endpoint is fixed at the origin and the angle of the link with the x-axis is chosen at fixed intervals of \( \frac{\pi}{6} \) (see Fig. 3b). To avoid cluttering the figure, we only visualize part of the recursive construction: We place a copy of this base roadmap on each of the endpoints of the link (Fig. 3b) and iteratively repeat this process for all endpoints around the origin (Fig. 3c, 3d). Even for this simple example with only twelve base configurations we obtain a highly dense tiling of \( \mathcal{F}^S \).

We describe the set of neighbors of a vertex \( C \in \mathcal{V} \). First, by the definition of the tiling roadmap there exists an index \( j \) and a point \( p_j \) for which \( C \) is a vertex of \( G_j(p_j) \). Now, by Observation 2 \( C_{j,i} + p_j = C \) for some base configuration \( C_j \). Recall that \( \text{nbr}(C_{j,i}, G_j(0)) \) is the set of neighbors of the configuration \( C_{j,i} \) in the base roadmap \( G_j(0) \). Hence,

\[
\text{nbr}(C, G_j(p_j)) = \{ C' \mid C' + p_j \in \text{nbr}(C_{j,i}, G_j(0)) \}
\]

By Lemma 3 this is true for every index \( j \). Thus, the set of neighbors of a vertex \( C \in \mathcal{V} \) in \( \mathcal{G} \) is

\[
\text{nbr}(C, \mathcal{G}) = \bigcup_j \text{nbr}(C, G_j(p_j)).
\]

In order to compute the neighbors of a vertex \( C \), one simply needs to identify the base configuration \( C_i \) that induces \( C \). Then, for each index \( j \), collect the set of neighbors of the configuration \( C_{j,i} \) in the base roadmap \( G_j(0) \) and appropriately translate each one.

**C. Path planning with \( \mathcal{G} \)**

We describe how \( \mathcal{G} \) is used to find a solution given a query that consists of a start configuration \( S \in \mathcal{F} \) and a target region \( T \subseteq \mathcal{F} \). The solution is found by (i) adding the start configuration \( S \) to each base roadmap (together with local paths in this roadmap) and (ii) attempting to find a collision-free path from \( S \) to \( T \) using \( \mathcal{G} \). To connect \( S \) to each base roadmap we do the following. Let \( S_j := S - j(S) \) for

\(^{3}\)We emphasize that in the query phase, one still needs to check if the path to each such neighbor is in the obstacle-collision free space \( \mathcal{F}^O \).

![Fig. 2. Visualization of base roadmaps for a two-link planar robot. (a) Two base configurations depicted in green and purple. (b) The configurations of the three base roadmaps induced by the two base configurations.](image-url)
V. EFFICIENT TRAVERSAL OF TILING ROADMAPS

To search for a path using the tiling roadmap, one may consider using standard pathfinding algorithms on graphs such as A* [27]. Indeed, our first implementation for traversal of \( \mathcal{G} \) employed this algorithm. However, when the graph is dense, and the problem lacks a good heuristic function to guide the search, A* (and its many variants) resort to a BFS-like search which is highly demanding, in terms of running time and memory consumption. As we demonstrate in Section VI, the tiling roadmap implicitly defines a dense covering of the self-collision free space. This is backed-up by our experimental work, which showed that A* was unable to make progress on \( \mathcal{G} \), regardless of the heuristic that was employed.

Our motion-planning algorithm integrates the implicitly-represented tiling roadmap \( \mathcal{G} \) with a highly-efficient pathfinding technique called discrete-RRT [38] (dRRT). We call our algorithm TR-dRRT, where “TR” stands for “tiling roadmap”. The dRRT algorithm is an adaptation of the RRT algorithm for the purpose of exploring discrete, geometrically-embedded graphs. Since the tiling roadmap \( \mathcal{G} \) serves as an approximation of some relevant portion of the self-collision free space, traversal of the graph can be viewed as a process of exploring the space. dRRT rapidly explores the graph by biasing the search towards vertices embedded in unexplored regions of the space. The algorithm only needs an oracle that retrieves information regarding neighbors of visited vertices. Roughly speaking, given a configuration \( p \) and a vertex \( u \) of the roadmap, the oracle returns the neighbor \( v \) of \( u \) such that the direction from \( u \) to \( v \) is closest to the direction from \( u \) to \( p \). See Fig. 4

VI. PROPERTIES OF THE TILING ROADMAP

We discuss some theoretical properties of the tiling roadmap. We begin by stating that the tiling roadmap covers the collision-free space and providing a proof sketch. Specifically, we show that for every self-collision free configuration \( C_S \in \mathcal{F}^S \) there exist another configuration \( C \in \mathcal{V} \) such that \( C_S \) and \( C \) are arbitrarily close as the number of samples tends to infinity. We then provide a discussion on the additional steps required to show that our method is probabilistic complete.

A. Coverage

Intuitively, as the number of base configurations grows, the tiling roadmap \( \mathcal{G} \) increases its coverage of \( \mathcal{F}^S \). For simplicity, we consider a planar snake-like robot whose configuration space is represented by \( \mathbb{R}^2 \times S^1 \times \ldots \times S^1 \). We use the term “head” to refer to the first anchor point, namely, the first endpoint of the first link, and denote by \( L \) the length of the first link. We will use the standard representation of configurations in such a configuration space. Namely, we will consider a configuration as a pair \( (p, \Theta) \), where \( p = (x, y) \in \mathbb{R}^2 \) is the location of the head of the robot and \( \Theta = (\theta_1, \ldots, \theta_{m-1}) \) is
a list of angles, where $\theta_i \in S^1$ is the angle between the $x$-axis and link $i$. We omit here details regarding the metric in order to simplify the presentation. Clearly, a rigorous proof will have to take the specific metric into consideration.

We are now ready to state our claims. First, we show that from the fact that base configurations are uniformly sampled $(c)$ exclusive the configurations $C, C'$ are self-collision free and there exists a self-collision free path between the two. Thus, we only assume that they are indeed nearest neighbors, which is true for sufficiently large value of $k$, which represents the maximal vertex degree in every base roadmap. See Fig. 5b.

We will show that using the assumption that $C, C'$ are neighbors in both base roadmaps $G_1(0)$ and $G_2(0)$, we can construct a series of points $\{q_i' \mid i \in \mathbb{N}\}$ which lie on a line intersecting the origin. These points represent locations where the head of the robot may lie (step S2) induced by $\ell'$ (dashed yellow line) and parallel copies of $\ell''$ (dashed blue line). Only every other copy of $\ell''$ is depicted to avoid cluttering the figure.

S1 Given a start configuration where the head of the robot is at the origin $0$, one can construct a straight line $\ell'$ in $\mathbb{R}^2$ such that (i) $\ell'$ intersects $0$ and (ii) for every point on $\ell'$, there exists a vertex in the tiling map whose head lies on $\ell'$ and is close to $p$ up to any desired resolution as the number of samples tends to infinity.

S2 The construction in step S1 can be used to construct two non-parallel lines $\ell'$, $\ell''$ that span $\mathbb{R}^2$. This in turn implies that $\ell', \ell''$ induce a grid such that any point in $\mathbb{R}^2$ as close as desired to a grid point.

**Sketch of step S1:** Set $\alpha_n = \pi/n$ and assume that the following configurations are in the set of $n$ base configurations: $C = (0, 0, \theta_2 \ldots \theta_{n-1})$ and $C' = (0, \alpha_n, \theta_2 \ldots \theta_{m-1})$. Namely, $C$ is the base configuration where the head of the robot is placed at the origin, the first link lies on the $x$-axis and the remaining angles $\theta_2 \ldots \theta_{m-1}$ are uniformly chosen. $C'$ is defined in a similar manner, except that the first link has a small angle of $\alpha_n$ with the $x$-axis. See Fig. 5c. Moreover, we assume that $C, C'$ are neighbors in both base roadmaps $G_1(0)$ and $G_2(0)$ (namely, the base roadmaps of the first two anchor points). Such an assumption is valid since both $C, C'$ are self-collision free and there exists a self-collision free path between the two. Thus, we only assume that they are indeed nearest neighbors, which is true for sufficiently large value of $k$, which represents the maximal vertex degree in every base roadmap. See Fig. 5b.

We will show that using the assumption that $C, C'$ are neighbors in both base roadmaps $G_1(0)$ and $G_2(0)$, we can construct a series of points $\{q'_i \mid i \in \mathbb{N}\}$ which lie on a line intersecting the origin. These points represent locations where the head of the robot may lie (step S2) induced by $\ell'$ (dashed yellow line) and parallel copies of $\ell''$ (dashed blue line). Only every other copy of $\ell''$ is depicted to avoid cluttering the figure.

Recall that $L$ represents the length of the first link. Using basic trigonometry we have that

$$q'_i = (i \cdot L \cos(\alpha_n - 1), i \cdot L \sin(\alpha_n)).$$

Moreover, all such points $q'_i$ lie on a line $\ell'$ intersecting the origin and every two consecutive points are of (Euclidean) distance

$$\Delta(n) = \|q'_{i+1} - q'_i\|_2 = 2L \sin \frac{\alpha_n}{2} = 2L \sin \frac{\pi}{2n}.$$ Clearly $\lim_{n \to \infty} \Delta(n) = 0$.

**Sketch of step S2:** The construction of the line $\ell'$ in step S1 was described for a specific pair of configurations $C, C'$. Now, if we consider the configuration $C'' = (0, -\alpha_n, \theta_2 \ldots \theta_{m-1})$, the same construction holds for the pair of configurations $C, C''$ to obtain the points $\{q''_i \mid i \in \mathbb{N}\}$ who all lie on the line $\ell''$ and for which

$$q''_i = (i \cdot L \cos(\alpha_n - 1), -i \cdot L \sin(\alpha_n)).$$

Fig. 5. Construction used head coverage. (a) Base configurations $C$ (purple) and $C'$ (green) which are identical except for the first angle $\alpha_n = \pi/n$. The first link (of length $L$) of the base configurations is depicted using solid lines and only it will be used in (b) and (c). (b) The two base roadmaps $G_1(0), G_2(r)$ for $C, C'$. (c) Iteratively constructing the points $\{q'_i\}$ which densely lie on the line $\ell'$ passing through the origin. (d) The points $\{q''_i\}$ constructed using $C, C''$ ($C''$ is identical to $C$ except for the first angle $-\alpha_n$). (d) The grid of points (black crosses) where the head of the robot may lie (step S2) induced by $\ell'$ (dashed yellow line) and parallel copies of $\ell''$ (dashed blue line). Only every other copy of $\ell''$ is depicted to avoid cluttering the figure.
neighbor search is performed using the FLANN library \[23\].

represents the coordinates of the coverage \[13\] rely on two properties of the examined algorithm. First, typical proofs of probabilistic completeness (see, e.g., \[10, 12\]) needs to navigate in tight quarters among obstacles. The Coiled Scenario (Fig. 1b) depicts a robot with ten links which needs to uncoil itself. This represents a scenario where the majority of the collisions that will occur will be due to self collision. In the Bricks Scenario (Fig. 1c) a small 13-linked robot needs to move from a tight start configuration to reach the goal.

Finally, in the Gripper scenario (Fig. 1d), we demonstrate how one can employ the same concepts presented in this paper to anchored robots. The 10-link robot is anchored at its middle joint and both of the robot’s endpoints need to reach the goal region. This simulates two robotic arms that need to grasp an object. To solve this problem, we constructed one base roadmap representing configurations where the middle joint is anchored. Currently, it is not clear how to extend the tiling-roadmap concept to the case of anchored robots (see discussion in Section VIII). Thus, we resort to dense sampling in the preprocessing stage. In the query stage, this roadmap was traversed using the dRRT algorithm. We note that there are more suitable algorithms to solve this problem than our TR-dRRT algorithm. However, this simple approach, which outperforms the RRT algorithm, serves as a proof of concept of the applicability of our technique to anchored-robot settings as well.

Both algorithms are not designed to return high-quality paths. Thus, we only measured the time to answer a query successfully. We report the results in Fig. 6. The preprocessing times for constructing the tiling roadmaps (Fig 1-3) are fairly low (between two to three minutes). For the Gripper scenario, we had to apply much longer preprocessing times (roughly ten minutes) in order to construct a dense roadmap. For both the Tight and the Coiled Scenario, TR-dRRT is roughly ten times faster than the RRT algorithm, while in the Bricks Scenario, TR-dRRT is roughly five times faster than the RRT algorithm. For the Gripper Scenario, TR-dRRT is roughly twice as fast as the RRT algorithm.

VIII. Discussion and future work

This paper introduces a new paradigm in sampling-based motion-planning in which the specific structure of the robot is taken into consideration to reduce the amount of self-collision checks one has to perform online. We demonstrate this paradigm using the TR-dRRT algorithm which is designed...
for free-flying multi-link robots. TR-dRRT performs a preprocessing stage, which results in an implicit tiling roadmap that represents an infinite set of configurations and transitions which are entirely self-collision free. Given a query, the search of the configuration space is restricted to the tiling roadmap. As a result, no self-collision checks need to be performed, and this stage is dedicated exclusively to finding an obstacle-collision free path.

While the experimental results are promising, TR-dRRT has its limitations. The explicitly-constructed base roadmaps should accurately capture the self-collision free spaces for which one of the links of the robot is anchored. For a robot with \( \delta \) degrees of freedom moving in \( \mathbb{R}^d \), this space is \( (d-\delta) \)-dimensional. Clearly, the favourable characteristics of our approach diminish as \( \delta \) increases.

To overcome the so-called “curse of dimensionality” for this specific type of robots, we believe that one can apply our technique in a recursive manner. For instance, assume that the self-collision free space of a “snake-like” robot with \( m \) links can be captured by a roadmap accurately and efficiently. Now, given a robot with \( 2m \) links, concatenate two configurations from this roadmap in the pathfinding phase. There are many details that should be addressed: Clearly, one still needs to check for self-collisions between the two concatenated configurations. Moreover, it is not straightforward how to use the precomputed local paths. However, using existing methods which recursively use copies of precomputed subspaces in motion-planning algorithms \([33]\), we believe that the approach can be applied to robots with more degrees of freedom than we presented in this paper. Moreover, we believe that this approach is also applicable to anchored multi-link robots which are prevalent in industry and academia.

## References

[1] Nancy M. Amato and Guang Song. Using Motion Planning to Study Protein Folding Pathways. *Journal of Computational Biology*, 9(2):149–168, 2002.

[2] Matthew H. Austern. *Generic programming and the STL: using and extending the C++ Standard Template Library*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[3] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005.

[4] Vicente Ruiz de Angulo, Juan Cortés, and Thierry Simeón. BioCD : An Efficient Algorithm for Self-collision and Distance Computation between Highly Articulated Molecular Models. In *RSS*, pages 241–248, 2005.

[5] GAMMA group. PQP - a proximity query package, 1999. URL http://gamma.cs.unc.edu/SSV.

[6] Li Han. Hybrid probabilistic roadmap Monte Carlo motion planning for closed chain systems with spherical joints. In *ICRA*, pages 920–926, 2004.

[7] Li Han and Nancy Amato. A Kinematics-Based Probabilistic Roadmap Method for Closed Chain Systems. In *WAFR*, pages 233–246, 2000.

[8] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *I. J. Rob. Res.*, 30(7):846–894, 2011.

[9] Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *T. Rob. and Aut.*, 12(4):566–580, 1996.

[10] Lydia E. Kavraki, Mihail N. Kolountzakis, and Jean-Claude Latombe. Analysis of probabilistic roadmaps for path planning. *T. Rob. and Aut.*, 14(1):166–171, 1998.

[11] Keith Kotay, Daniela Rus, Marsette Vona, and Craig McGray. The self-reconfiguring robotic molecule: Design and control algorithms. In *WAFR*, pages 375–386, 1998.

[12] J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *ICRA*, volume 2, pages 995–1001 vol.2, 2000.

[13] Andrew M. Ladd and Lydia E. Kavraki. Generalizing the Analysis of PRM. In *ICRA*, pages 2120–2125, 2002.

[14] Andrew M. Ladd and Lydia E. Kavraki. Using Motion Planning for Knot Untangling. *I. J. Rob. Res.*, 23(7-8):797–808, 2004.

[15] Steven M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[16] Steven M. LaValle and James J. Kuffner Jr. Randomized Kinodynamic Planning. *I. J. Rob. Res.*, 20(5):378–400, 2001.

[17] Maxim Likhachev, Geoffrey J. Gordon, and Sebastian Thrun. ARA*: Anytime A* with Provable Bounds on Sub-Optimality. In *Neural Information Processing Systems*, pages 767–774, 2003.

[18] Guanfeng Liu and Jeffrey C. Trinkle. Complete Path Planning for Planar Closed Chains Among Point Obstacles. In *RSS*, pages 33–40, 2005.

[19] Itay Lotan, Fabian Schwarzer, Dan Halperin, and Jean-Claude Latombe. Efficient maintenance and self-collision testing for Kinematic Chains. In *SoCG*, pages 43–52, 2002.

[20] Troy McMahon, Shawna L. Thomas, and Nancy M. Amato. Sampling-based motion planning with reachable volumes: Theoretical foundations. In *ICRA*, pages 6514–6521, 2014.

[21] Troy McMahon, Shawna L. Thomas, and Nancy M. Amato. Sampling based motion planning with reachable volumes: Application to manipulators and closed chain systems. In *IROS*, pages 3705–3712, 2014.

[22] Jean-Pierre Merlet. A local motion planner for closed-loop robots. In *IROS*, pages 3088–3093, 2007.

[23] M. Muja and D. G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *VISSAPP*, pages 331–340. INSTICC Press, 2009.

[24] An Nguyen, Leonidas J. Guibas, and Mark Yim. Controlled module density helps reconfiguration planning.
[25] Jun O’Hara. The configuration space of a spider. *Knots and Everything Book*, 40:245–252, 2007.

[26] Jia Pan, Liangjun Zhang, and Dinesh Manocha. Retraction-based RRT planner for articulated models. In *ICRA*, pages 2529 – 2536, 2010.

[27] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[28] Josep M. Porta, Juan Cortés, Lluís Ros, and Federico Thomas. A space decomposition method for path planning of loop linkages. In *IROS*, pages 1882–1888, 2007.

[29] Josep M. Porta, Lluís Ros, and Federico Thomas. A Linear Relaxation Technique for the Position Analysis of Multiloop Linkages. *Trans. on Rob.*, 25(2):225–239, 2009.

[30] Barak Raveh, Angela Enosh, Ora Schueler-Furman, and Dan Halperin. Rapid Sampling of Molecular Motions with Prior Information Constraints. *PLoS Computational Biology*, 5(2), 2009.

[31] Stephane Redon, Young J. Kim, Ming C. Lin, and Dinesh Manocha. Fast Continuous Collision Detection for Articulated Models. In *Symposium on Solid Modeling and Applications*, pages 145–156, 2004.

[32] John H. Reif. Complexity of the Mover’s Problem and Generalizations (Extended Abstract). In *FOCS*, pages 421–427, 1979.

[33] Oren Salzman, Michael Hemmer, and Dan Halperin. On the Power of Manifold Samples in Exploring Configuration Spaces and the Dimensionality of Narrow Passages. In *WAFR*, pages 313–329, 2012.

[34] Jacob T. Schwartz and Micha Sharir. On the “piano movers” problem: II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4(3):298 – 351, 1983.

[35] Fabian Schwarzer, Mitul Saha, and Jean-Claude Latombe. Adaptive Dynamic Collision Checking for Single and Multiple Articulated Robots in Complex Environments. *Robotics, Trans. on*, 21(3):338–353, 2005.

[36] Nir Shvalb, Moshe Shoham, and David Blanc. Motion Planning for a Class of Planar Closed-chain Manipulators. *Forum Mathematicum*, 17(6):1033–1042, 2005.

[37] Nir Shvalb, Moshe Shoham, Guanfeng Liu, and Jeffrey C. Trinkle. Motion Planning for a Class of Planar Closed-chain Manipulators. *I. J. Rob. Res.*, 26(5):457–473, 2007.

[38] Kiril Solovey, Oren Salzman, and Dan Halperin. Finding a Needle in an Exponential Haystack: Discrete RRT for Exploration of Implicit Roadmaps in Multi-Robot Motion Planning. In *WAFR*, to appear, 2014.

[39] Guang Song and Nancy M. Amato. A motion-planning approach to folding: from paper craft to protein folding. *T. Rob. and Aut.*, 20(1):60–71, 2004.

[40] Xinyu Tang, Shawa L. Thomas, Phillip Coleman, and Nancy M. Amato. Reachable Distance Space: Efficient Sampling-Based Planning for Spatially Constrained Sys-

[41] Jeffrey C. Trinkle and R. James Milgram. Complete Path Planning for Closed Kinematic Chains with Spherical Joints. *I. J. Rob. Res.*, 21(9):773–790, 2002.

[42] Dawen Xie and Nancy M. Amato. A Kinematics-based Probabilistic Roadmap Method for High DOF Closed Chain Systems. In *ICRA*, pages 473–478, 2004.

[43] Jeffery H. Yakey, Steven M. LaValle, and Lydia E. Kavraki. Randomized path planning for linkages with closed kinematic chains. *T. Rob. and Aut.*, 17(6):951–958, 2001.

[44] Y. Zhang, K. Hauser, and J. Luo. Unbiased, scalable sampling of closed kinematic chains. In *ICRA*, pages 2459–2464, 2013.