Collision detection or nearest-neighbor search? On the computational bottleneck in sampling-based motion planning*

Michal Kleinbort[1]**, Oren Salzman[2]**, and Dan Halperin[1]

[1] Blavatnik School of Computer Science, Tel-Aviv University, Israel
[2] Carnegie Mellon University, Pittsburgh PA 15213, USA

Abstract. The complexity of nearest-neighbor search dominates the asymptotic running time of many sampling-based motion-planning algorithms. However, collision detection is often considered to be the computational bottleneck in practice. Examining various asymptotically optimal planning algorithms, we characterize settings, which we call *NN-sensitive*, in which the *practical* computational role of nearest-neighbor search is far from being negligible, i.e., the portion of running time taken up by nearest-neighbor search is comparable, or sometimes even greater than the portion of time taken up by collision detection. This reinforces and substantiates the claim that motion-planning algorithms could significantly benefit from efficient and possibly specifically-tailored nearest-neighbor data structures. The asymptotic (near) optimality of these algorithms relies on a prescribed connection radius, defining a ball around a configuration q, such that q needs to be connected to all other configurations in that ball. To facilitate our study, we show how to adapt this radius to non-Euclidean spaces, which are prevalent in motion planning. This technical result is of independent interest, as it enables to compare the radial-connection approach with the common alternative, namely, connecting each configuration to its k nearest neighbors (k-NN). Indeed, as we demonstrate, there are scenarios where using the radial connection scheme, a solution path of a specific cost is produced ten-fold (and more) faster than with k-NN.

1 Introduction

Given a robot $\mathcal{R}$ moving in a workspace $\mathcal{W}$ cluttered with obstacles, motion-planning (MP) algorithms are used to efficiently plan a path for $\mathcal{R}$, while avoiding collision with obstacles [9, 25]. Prevalent algorithms abstract $\mathcal{R}$ as a point in

* This work has been supported in part by the Israel Science Foundation (grant no. 825/15), by the Blavatnik Computer Science Research Fund, and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University. O. Salzman has been also supported by the German-Israeli Foundation (grant no. 1150-82.6/2011), by the National Science Foundation IIS (#1409003), Toyota Motor Engineering & Manufacturing (TEMA), and the Office of Naval Research. Parts of this work were done while O. Salzman was a student at Tel Aviv University.

** M. Kleinbort and O. Salzman contributed equally to this paper.

a high-dimensional space called the *configuration space* (C-space) $\mathcal{X}$ and plan a path (curve) in this space. A point, or a configuration, in $\mathcal{X}$ represents a placement of $\mathcal{R}$ that is either collision-free or not, subdividing $\mathcal{X}$ into the sets $\mathcal{X}_{\text{free}}$ and $\mathcal{X}_{\text{forb}}$, respectively. *Sampling-based* algorithms study the structure of $\mathcal{X}$ by constructing a graph, called a *roadmap*, which approximates the connectivity of $\mathcal{X}_{\text{free}}$. The nodes of the graph are collision-free configurations sampled at random. Two (nearby) nodes are connected by an edge if the straight line segment connecting their configurations is collision-free as well.

Sampling-based MP algorithms are typically implemented using two primitive operations: *Collision detection* (CD) [26], which is primarily used to determine whether a configuration is collision-free or not, and *Nearest-neighbor* (NN) search, which is used to efficiently return the nearest neighbor (or neighbors) of a given configuration. CD is also used to test if the straight line segment connecting two configurations lies in $\mathcal{X}_{\text{free}}$—a procedure referred to as *local planning* (LP). In this paper we consider both CD and LP calls when measuring the time spent on collision-detection operations.

Contribution The complexity of NN search dominates the asymptotic running time of many sampling-based MP algorithms However, the main computational bottleneck in practical settings is typically considered to be LP [9, 25]. In this paper we argue that this may not always be the case. We describe settings, which we call *NN-sensitive*, where the (computational) role of NN search after finite running-time is far from negligible and merits the use of advanced and specially-tailored data structures; see Fig. 1 for a plot demonstrating this behavior. NN-sensitive settings may be due to (i) planners that *algorithmically* shift the computational weight to NN search; (ii) scenarios in which certain planners perform mostly NN search; or (iii) parameters' values for which certain planners spend the same order of running time on NN and CD.

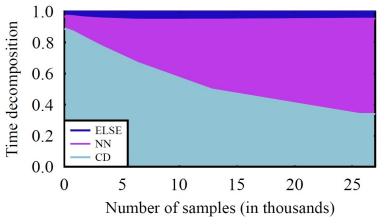


Fig. 1: Running-time breakdown of the main primitive operations used in MPLB [31] applied to the 3D-Grid scenario (Fig. 2a). For additional data, see Sec. 4. Best viewed in color.

Specifically, we focus on asymptotically (near) optimal MP algorithms. We study the ratio between the overall time spent on NN search and CD after N configurations were sampled. We observe situations where NN takes up to 100% more time than CD in scenarios based on the Open Motion Planning Library [11]; on synthetic high-dimensional C-spaces we even observe a ratio of 4500%.

We mostly concentrate on the *radial* version of MP algorithms, where the set of neighbors in the roadmap of a given configuration q includes all configurations of maximal distance r from q. To do so in *non-Euclidean* C-spaces, we derive closed-form expressions for the volume of a unit ball in several common C-spaces. This technical result is of independent interest, as the lack of such expressions seems to have thus far prevented the exploration and understanding of these types of algorithms in non-Euclidean settings—most experimental evaluation reported in the literature on the radial version of asymptotically-

optimal planners is limited to Euclidean settings only. We show empirically that in certain scenarios, the radial version of an MP algorithm produces a solution of specific cost more than *ten times faster* than the non-radial version, namely, where each node is connected to its k nearest neighbors.

We emphasize that we are not the first to claim that in certain cases NN may dominate the running time of MP algorithms, see, e.g., [5]. However, we take a systematic approach to characterize and analyze when this phenomenon occurs.

Throughout the paper we use the following notation: For an algorithm ALG, let $\chi_{\text{ALG}}(S)$ be the ratio between the overall time spent on NN search and CD for a specific motion-planning problem after a set S of configurations was sampled, where we assume that all other parameters of the problem, the workspace and the robot, are fixed—see details below. Let $\chi_{\text{ALG}}(N)$ be the expected value of $\chi_{\text{ALG}}(S)$ over all sample sets S of size N.

Organization We start with an overview of related work in Sec. 2 and continue in Sec. 3 to summarize, for several algorithms, the computational complexity in terms of NN search and CD. We show that asymptotically, as N tends to infinity, $\chi_{\text{ALG}}(N)$ tends to infinity as well. In Sec. 4 we point out several NN-sensitive settings together with simulations demonstrating how $\chi_{\text{ALG}}(N)$ behaves in such settings. These simulations make use of the closed-form expressions of the volume of unit balls, which are detailed in Sec. 5.

2 Background and related work

We start by giving an overview of asymptotically (near) optimal MP algorithms and continue with a description of CD and NN algorithms.

2.1 Asymptotically optimal sampling-based motion planning

A random geometric graph (RGG) $\mathcal{G}$ is a graph whose vertices are sampled at random from some space $\mathcal{X}$. Every two configurations are connected if their distance is less than a connection radius r_n (which is typically a function of the number of nodes n in the graph). We are interested in a connection radius such that, asymptotically, for any two vertices x, y, the cost of a path in the graph connecting x and y converges to the minimal-cost path connecting them in $\mathcal{X}$. A sufficient condition to ensure this property is that [20]

$$r_n \geq 2\eta \left(\frac{\mu(\mathcal{X}_{\text{free}})}{\zeta_d}\right)^{1/d} \left(\frac{1}{d}\right)^{1/d} \left(\frac{\log n}{n}\right)^{1/d}. \tag{1}$$

Here d is the dimension of $\mathcal{X}$, $\mu(\cdot)$ and ζ_d denote the Lebesgue measure (volume) of a set and of the d-dimensional unit ball, respectively, and $\eta \geq 1$ is a tuning parameter that allows to balance between exploring unvisited regions of the C-space and connecting visited regions. Alternatively, an RGG where every vertex is connected to its $k_n \geq e(1 + 1/d) \log n$ nearest neighbors will ensure similar convergence properties [21]. Unless stated otherwise, we focus on RGGs of the

former type. For a survey on additional models of RGGs, their properties and their connection to sampling-based MP algorithms, see [33].

Most asymptotically-optimal planners sample a set of collision-free configurations (either incrementally or in batches). This set of configurations induces an RGG $\mathcal{G}$ or a sequence of increasingly dense RGGs $\{\mathcal{G}_n\}$ whose vertices are the sampled configurations. Set $\mathcal{G}' \subseteq \mathcal{G}$ to be the subgraph of $\mathcal{G}$ whose edges represent collision-free motions. These algorithms construct a roadmap $\mathcal{H} \subseteq \mathcal{G}'$.

PRM* and RRG [21] call the local planner for *all* the edges of $\mathcal{G}$. To increase the convergence rate to high-quality solutions, algorithms such as RRT* [21], RRT$^{\#}$ [1], LBT-RRT [32], FMT* [20], MPLB [31], Lazy-PRM* [14], and BIT* [13] call the local planner for a *subset* of the edges of $\mathcal{G}$.

Reducing the number of LP calls is typically done by constructing $\mathcal{G}$ (using nearest-neighbor operations only) and deciding for which edges to call the local planner. Many of the algorithms mentioned do so by using graph operations such as shortest-path computation. These operations often take a tiny fraction of the time required for LP computation. However, in more recent algorithms such as FMT* and BIT* this may not be true.

2.2 Collision detection

Most CD algorithms are bound to certain types of models, where rigid polyhedral models are the most common. They often allow answering proximity queries as well (i.e., separation-distance computation or penetration-depth estimation). Several software libraries for collision detection are publicly available [10, 24]. The most general of which is the Flexible Collision Library (FCL) [29] that integrates several techniques for fast and accurate collision checking and proximity computation. For polyhedral models, which are prevalent in MP settings, most commonly-used techniques are based on *bounding volume hierarchies* (BVH).

A collision query using BVHs may take $O(m^2)$ time in the worst case, where m is the complexity of the obstacle polyhedra (recall that we assume that the robot system has constant-description complexity). However, tighter bounds may be obtained using methods tailored for large environments [10, 16]. Specifically, the time complexity is $O(m \log^{\delta-1} m + s)$, where $\delta \in \{2, 3\}$ is the dimension of the workspace $\mathcal{W}$ and s is the number of intersections between the bounding volumes. Other methods relevant to MP are mentioned in [23]. For a survey on the topic, see [26].

2.3 Nearest-neighbor methods: exact and approximate

Nearest-neighbor (NN) algorithms are frequently used in various domains. In the most basic form of the problem we are given a set P of n points in a metric space $M = (X, \rho)$, where X is a set and $\rho : X \times X \to \mathbb{R}$ is a distance metric. Given a query point $q \in X$, we wish to efficiently report the nearest point $p \in P$ to q. Immediate extensions include the k-nearest-neighbors (K-NN) and the r-near-neighbors (R-NN) problems. The former reports the k nearest points of P to q, whereas the latter reports all points of P within a distance r from q.

In the plane, the NN search problem can be efficiently solved by constructing a Voronoi diagram of P in $O(n \log n)$ time and preprocessing it to a linear-size point-location data structure in $O(n \log n)$ time. Queries are then answered in $O(\log n)$ time [4, 15]. However, for high-dimensional point sets this approach becomes infeasible, as it is exponential in the dimension d. This phenomenon is often termed "the curse of dimensionality" [19].

An efficient data structure for low dimensional spaces[3] is the kd-tree [3, 12], whose expected query complexity is logarithmic in n under certain assumptions. However, the constant factors hidden in the asymptotic query time depend exponentially on the dimension d [2]. Another structure suitable for low-dimensional spaces is the geometric near-neighbor access tree (GNAT); as claimed in [7], typically the construction time is $O(dn \log n)$ and only linear space is required. In the extended version [23] we also discuss methods that adapt to the intrinsic dimension of the subspace where the points lie.

All the aforementioned structures give an exact solution to the problem. However, many approximate algorithms exist, and often perform significantly faster than the exact ones, especially when d is high. Among the prominent approximate algorithms are *Balanaced box-decomposition trees* (BBD-trees) [2], and Locality-sensitive hashing (LSH) [19]. See [18] for a survey on approximate NN methods in high-dimensional spaces.

Finally, we note that in the context of MP, several specifically-tailored exact [17, 34] and approximate [22, 30] techniques were previously described. A theoretical justification for using approximate NN methods rather than exact ones is proven in [33] for PRM*.

3 The asymptotic behavior of common MP algorithms

In this section we provide more background on the asymptotic complexity analysis of various sampling-based MP algorithms. We then show that for both PRM-type algorithms and RRT-type algorithms, the expected ratio between the time spent on NN search and the time spent on CD goes to infinity as $n \to \infty$.

We denote by N the total number of configurations sampled by the algorithm, and by n the number of collision-free configurations in the roadmap. Let m denote the complexity of the workspace obstacles and assume that the robot is of constant-description complexity[4].

3.1 Complexity of common motion-planning algorithms

We start by summarizing the computational complexity of the primitive operations and continue to detail the computational complexity of a selected set of algorithms. We assume familiarity with the planners that are discussed.

[3] We refer to a space as low dimensional when its dimension is at most a few dozens.

[4] The assumption that the robot is of constant-description complexity implies that testing for *self-collision* can be done in constant time.

Complexity of primitive operations The main primitive operations that we consider are (i) nearest-neigbhor operations (NN and R-NN) and (ii) collision-detection operations (CD and LP). Additionally, MP algorithms make use of priority queues and graph operations. We assume, as is typically the case, that the running time of these operations is negligible when compared to NN and CD.

Since many NN data structures require a preprocessing phase, the complexity of a single query should consider the amortized cost of preprocessing. However, since usually at least n NN or R-NN queries are performed, where n is the number of points stored in the NN data structure, this amortized preprocessing cost is asymptotically subsumed by the cost of a query.

A list of the common complexity bounds for the different types of NN queries can be found in the extended version of this paper [23]. As mentioned in Sec. 2.2, the complexity of a single CD operation for a robot of a constant-description complexity can be bounded by $O(m \log^{\delta-1} m + s)$, where $\delta \in \{2, 3\}$ is the dimension of the workspace and s is the number of intersections between the bounding volumes, which is $O(m^2)$ in the worst case. On the other hand, for a system with ℓ such robots, a CD operation is composed of ℓ single robot CD queries as well as $O(\ell^2)$ robot-robot collision checks. Local planning (LP) is often implemented using multiple CD operations along a densely-sampled C-space line-segment between two configurations. Specifically, we assume that the planner is endowed with a fixed parameter called STEP specifying the sampling density along edges. During LP, edges of maximal length r_n will be subdivided into $\lceil r_n/\text{STEP} \rceil$ collision-checked configurations (see also [25, p. 214]). Therefore, the complexity of a single LP query can be bounded by $O(r_n \cdot \mathrm{Q}_{\mathrm{CD}})$, where Q_{CD} is the complexity of a single CD query (here STEP is assumed to be constant).

Complexity of algorithms In order to choose which edges of $\mathcal{G}$ to explicitly check for being free, all algorithms need to determine (i) which of the N nodes are collision free and (ii) what are the neighbors of each node. Thus, these algorithms typically require N CD calls and n R-NN calls.

To quantify the number of LP calls performed by each algorithm, note that the expected number of neighbors of a node in $\mathcal{G}$ is $\Theta(\eta^d 2^d \log n)$ [33]. Therefore, if an algorithm calls the local planner for all (or for a constant fraction of) the edges of $\mathcal{G}$, then the expected number of LP calls will be $\Theta(\eta^d 2^d n \log n)$.

3.2 The asymptotic behavior of the ratio $\chi_{\mathrm{ALG}}(N)$

Let $\mathrm{T}_{\mathrm{CD}}(S)$ be the overall time spent on CD for a specific motion-planning problem after a set S of configurations was sampled, where we assume, as before, that all other parameters of the problem are fixed. Let $\mathrm{T}_{\mathrm{CD}}(N)$ be the expected value of $\mathrm{T}_{\mathrm{CD}}(S)$ over all sample sets S of size N. We show here that the expected value $\chi_{\mathrm{ALG}}(N)$ of the ratio over all sample sets of size N goes to infinty as $N \to \infty$ for both sPRM* and RRT*. Recall that we are interested in the expected value of the ratio. We do that by looking at the ratio between a lower bound on the time of NN and $\mathrm{T}_{\mathrm{CD}}(N)$, defined above.

To obtain a lower bound on the time of NN, we assume that the NN structure being used is a j-ary tree for a constant j, in which the data points are kept

in the leaves. This is a reasonable assumption, as many standard NN structures are based on trees [2, 3, 7, 12]. Performing n queries of NN (or R-NN) using this structure, one for every data point, costs $\Omega(n \log n)$, as each query involves locating the leaf in which the query point lies. It is easy to show this both for sPRM*, in which the NN structure is constructed given a batch of all data points, and for RRT*, where the structure is constructed incrementally.

Additionally, we have the following lemma, whose proof is in [23]:

Lemma 1 *If an algorithm uses a uniform set of samples and the C-space obstacles occupy a constant fraction of the C-space, then $n = \Theta(N)$ almost surely.*

For sPRM* it holds that $\mathrm{T}_{\mathrm{CD}}(N) = \#_{\mathrm{CD}} \cdot \mathrm{Q}_{\mathrm{CD}} + \#_{\mathrm{LP}} \cdot \mathrm{Q}_{\mathrm{LP}}$. Clearly, $\#_{\mathrm{CD}} = N$ and $\mathrm{Q}_{\mathrm{CD}} = O(m^2)$ (see Sec. 3.1). In expectation we have that $\#_{\mathrm{LP}} = O(n^2 \cdot r_n^d)$ and in addition $\mathrm{Q}_{\mathrm{LP}} = O(r_n \cdot \mathrm{Q}_{\mathrm{CD}})$. Finally, recall that $r_n = \Theta\left(2\eta \left(\log n/n\right)^{1/d}\right) = \Theta\left(\left(\log n/n\right)^{1/d}\right)$. Therefore,

$$\begin{aligned}
\mathrm{T}_{\mathrm{CD}}(N) &= N \cdot \mathrm{Q}_{\mathrm{CD}} + O(\eta^d 2^d n \log n) \cdot \mathrm{Q}_{\mathrm{LP}} \\
&= N \cdot \mathrm{Q}_{\mathrm{CD}} + O(\eta^d 2^d n \log n) \cdot O(r_n \cdot \mathrm{Q}_{\mathrm{CD}}) \\
&= O(m^2 N) + O(m^2 N^{1-1/d} \log^{1+1/d} N) = O(m^2 N). \qquad (2)
\end{aligned}$$

As $\Omega(n \log n)$ is a valid lower bound on the overall complexity of NN, there exists a constant $c_2 > 0$ s.t. the time for NN for a roadmap with n nodes is at least $c_2 n \log n$. Moreover, since $\mathrm{T}_{\mathrm{CD}}(N) = O(m^2 N)$ then there exists a constant $c_3 > 0$ s.t. the overall time for CD is at most $c_3 m^2 N$.

Thus, using Lemma 1, $\chi_{\mathrm{sPRM}^*}(N) \geq \frac{c_2 n \log n}{c_3 m^2 N} \geq \frac{c' \log N}{m^2}$, where $c' > 0$ is a constant. Observing that the above fraction goes to infinity as N goes to infinity, we obtain that $\lim_{N \to \infty} \chi_{\mathrm{sPRM}^*}(N) = \infty$, as anticipated.

We note that although m is assumed to be constant, we leave it in our analysis to emphasize its effect on $\chi_{\mathrm{ALG}}(N)$. In summary,

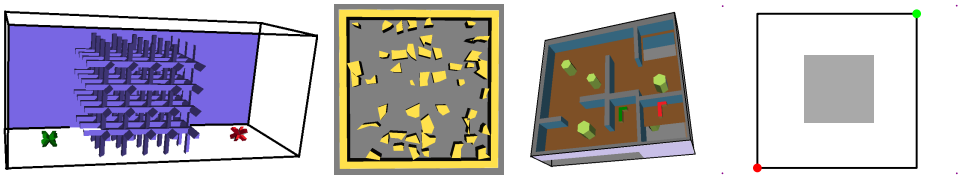
Proposition 2 *The values $\chi_{sPRM^*}(N)$ and $\chi_{RRT^*}(N)$ tend to infinity as $N \to \infty$.*

The proof for $\chi_{\mathrm{RRT}^*}(N)$ can be found in the extended version [23].

From a theoretical standpoint, NN search determines the asymptotic running time of typical sampling-based MP algorithms. In contrast, the common experience is that CD dominates the running time in practice. However, we show in the remainder of the paper that in a variety of special situations NN search is a non-negligible factor in the running-time in practice.

4 Nearest-neighbor sensitive settings

In this section we describe settings where the computational role of NN search in practice is far from negligible, even for a relatively small number of samples. We call these settings *NN-sensitive*. For each such setting we empirically demonstrate this behavior. In Sec. 4.1 we describe our experimental methodology and outline properties common to all our experiments. Each of the subsequent sections is devoted to a specific type of NN-sensitivity.



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

Fig. 2: Scenarios used in experiments. (a) 3D Grid, (b) 2D Random polygons, (c) 3D Cubicles and (d) dD Hypercube with a centered obstacle. Start and target configurations for a robot are depicted in green and red, respectively. Scenarios (b) and (c) are provided with the OMPL distribution. More details are provided in the body of the paper.

4.1 Experimental methodology

In our experiments, we ran the Open Motion Planning Library (OMPL 1.1) [11] on a 2.5GHz×4 Intel Core i5 processor with 8GB of memory. Each reported result is averaged over fifty (50) runs and includes error bars which denote the 20'th and 80'th percentiles. The scenarios used are depicted in Fig. 2.

Several of our experiments are in non-Euclidean C-spaces, which in turn require a closed-form expression for ζ_d, the measure (volume) of the d-dimensional unit ball (see Eq. 1). In Sec. 5 we describe a general approach to compute this value together with a heuristic that makes the computed radius effective in practice. This heuristic is used in all the experiments presented in this section.

What do we measure? Recall that our main thesis is that while the folklore in MP is that the running time of sampling-based algorithms in practice is strongly dominated by CD, we (and others) observe that quite often the time taken up by NN-search is significant, and not rarely larger than the time taken up by CD. Therefore, our primary measure is *wall time*, namely the running time spent on the different primitives as gauged in standard clock time (to distinguish from CPU-time or other more system-specific measurements like number of floating point operations). The principal reason for doing that is that wall time is what matters most in practice. This, for example, will affect the response time of a planner used by a robot system. One may argue that this measurement may only be meaningful for a very limited suite of software tools used by motion planners. However, we use state-of-the-art tools that are used by many. There is not such an abundance of efficient stable software tools for this purpose, and most researchers in the field seem to use a fairly small set of tools for CD and NN. This said, we still provide additional measurements for each experiment—the average number of basic operations, which should allow people who come up with their own (or specialized) MP primitives to assess what will be the effect of their special primitives on the overall running time of the algorithms in practice.

4.2 NN-sensitive algorithms

In recent years, several planners were introduced, which *algorithmically* shift some of the computational cost from CD to NN search. Two such examples are

Lazy-PRM* [14] and MPLB [31], though lazy planners were described before (e.g., [6]). Both algorithms delay local planning by building an RGG $\mathcal{G}$ over a set of samples *without* checking if the edges are collision free. Then, they employ graph-search algorithms to find a solution. To construct $\mathcal{G}$ only NN queries are required. Moreover, using these graph-search algorithms dramatically reduces the number of LP calls. Thus, in many cases (especially as the number of samples grows) the weight of CD is almost negligible with respect to that of NN.

Specifically, Lazy-PRM* iteratively computes the shortest path in $\mathcal{G}$ between the start and target configurations. LP is called only for the edges of the path. If some are found to be in collision, they are removed from the graph. This process is repeated until a solution is found or until the source and target do not lie in the same connected component. We use a batch variant of Hauser's Lazy-PRM* algorithm [14], which we denote by Lazy-sPRM*. This variant constructs the roadmap in the same fashion as sPRM* does but delays LP to the query phase.

MPLB uses $\mathcal{G}$ to compute lower bounds on the cost between configurations to tightly estimate the cost-to-go [31]. These bounds are then used as a heuristic to guide the search of an anytime version of FMT* [20]. The bounds are computed by running a shortest-path algorithm over $\mathcal{G}$ from the target to the source. Fig. 1 (on page 2) presents the amount of NN, CD and other operations used by MPLB running on the 3D Grid scenario for two robots translating and rotating in space that need to exchange their positions (Fig. 2a). With several thousands of iterations, which are required for obtaining a high-quality solution, NN dominates the running time of the algorithm. For additional details see [23].

Additional experiments demonstrating the behavior of NN-sensitive algorithms can be found in [5, 14, 31].

4.3 NN-sensitive scenarios

A scenario $\mathcal{S} = (\mathcal{W}, \mathcal{R})$ is defined by a workspace $\mathcal{W}$ and a robot system $\mathcal{R}$. The robot system $\mathcal{R}$ may, in turn, be a set of ℓ single constant-description complexity robots operating simultaneously in $\mathcal{W}$. Let the dimension d of $\mathcal{S}$ be the dimension of the C-space induced by $\mathcal{R}$, and, hence, $d = \Theta(\ell)$. Let the complexity of $\mathcal{S}$ be the complexity m of the workspace obstacles. Note that CD is affected by ℓ, as both robot-obstacle and robot-robot collisions should be considered. Therefore, the bound on the complexity of a CD operation is: $O(\ell \cdot m^2 + \ell^2)$, see Sec. 3.1.

We next show how the role of NN may increase when (i) the dimension of $\mathcal{S}$ increases or (ii) the complexity of $\mathcal{S}$ decreases.

The effect of the dimension *d* Proposition 2 states that as the number of samples tends to infinity, NN dominates the running time of the algorithm. A natural question to ask is "what happens when we fix the number of samples and increase the dimension?" The different structure of RRT* and sPRM* merits a different answer for each algorithm.

*RRT** Here, we show that the NN sensitivity grows with the number of *unsuccessful iterations*[5]. This implies that if the number of unsuccessful iterations

[5] Here, an iteration is said to be unsuccessful when the RRT* tree is not extended.

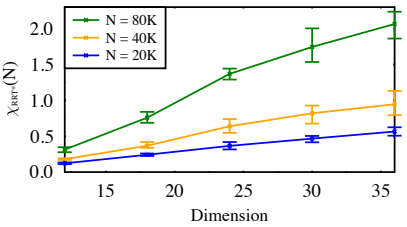


Fig. 3: $\chi_{\text{RRT*}}$(N) as a function of d in the 3D Cubicles scenario (Fig. 2c), when fixing the number N of iterations.

d	$\#_{\text{NN}}$	$\#_{\text{R-NN}}$	$\#_{\text{CD}}$	$\#_{\text{LP-A}}$	$\#_{\text{LP-B}}$	$\#_{\text{CD in LP}}$
12	80K	10.2K	80K	80K	325K	4,235K
24	80K	7.5K	80K	80K	724K	4,135K
36	80K	5.7K	80K	80K	886K	3,812K

Table 1: Average number of calls for the main primitive operations for different values of d, for $N = 80K$ iterations.

grows with the dimension, so will $\chi_{\text{RRT*}}(N)$. Indeed, we demonstrate this phenomenon in the 3D cubicles scenario (Fig. 2c). Note that in this situation the effect of d is indirect.

To better discuss our results we define two types of LP operations: the first is called when the algorithm attempts to grow the tree towards a random sample while the second is called during the rewiring step. We denote the former type of LP calls by LP-A and the latter by LP-B and note that LP-A will occur every iteration while LP-B will occur only in successful ones.

We use ℓ translating and rotating L-shaped robots. We gradually increase ℓ from two to six, resulting in a C-space of dimension $d = 6\ell$. Robots are placed in different sections of the workspace and can reach their target with little robot-robot interaction. We fix N and measure $\chi_{\text{RRT*}}(N)$ as a function of d. The results for several values of N are depicted in Fig. 3. Additionally, Table 1 shows the average number of operation calls for various values of d.

As d grows, the number of unsuccessful iterations grows (see $\#_{\text{LP-A}}$ in Table 1). This growth, which is roughly linear with respect to d induces a linear increase in $\chi_{\text{RRT*}}(N)$ for a given N (see Fig. 3). Furthermore, the slope of this line increases with N which further demonstrates the fact that for a fixed d, $\lim_{N\to\infty} \chi_{\text{RRT*}}(N) = \infty$.

Finally, Fig. 4 depicts the time decomposition of the main primitives as a function of d, for $N = 80K$.

*sPRM** Here, the NN sensitivity of the algorithm is more complex. The reason, roughly speaking, is that for a *fixed* n, the expected value of the number κ of

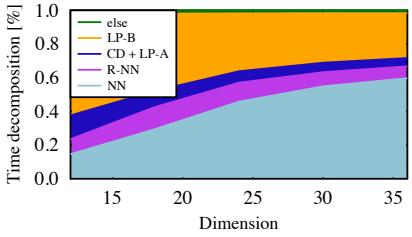


Fig. 4: Time breakdown of the main primitive operations in RRT* running on the 3D Cubicles scenario (Fig. 2c) as a function of d, for $N = 80$K iterations. Best viewed in color.

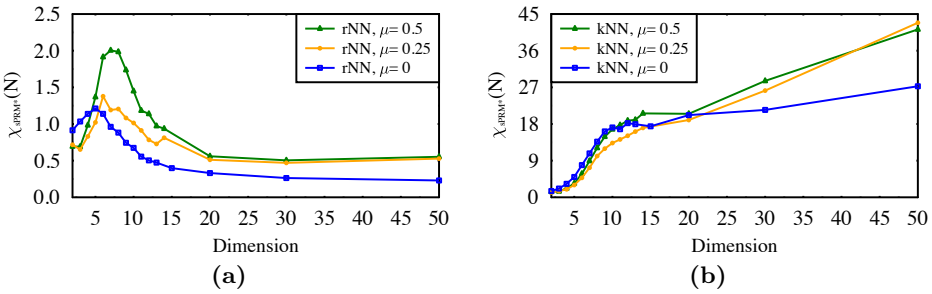


Fig. 5: $\chi_{\text{sPRM}^*}(N)$ as a function of d in the dD Hypercube scenario (Fig. 2d) for a roadmap with $n = 5000$ vertices. Experiments with obstacle measures μ of $0, 0.25$ and 0.5 are displayed. The connection strategies used in (a) and (b) are R-NN and K-NN, respectively.

reported neighbors is $\Theta(2^d n \log n)$. Thus, in expectation, κ grows exponentially in d. However, for large enough values of d, we have $\kappa = \Theta(n^2)$. Interestingly, this means that the computational cost of the overall NN time shifts from *finding* the set of nearest neighbors to *reporting* it.

According to our analysis, presented in [23], we expect to see an initial increase of $\chi_{\text{sPRM}^*}(N)$ followed by a convergence to a constant value. The increase in $\chi_{\text{sPRM}^*}(N)$ is due to the increasing complexity of finding the set of nearest neighbors, which grows with the dimension[6]. A possible decrease will occur as the computational weight "shifts" to reporting the set of neighbors followed by an asymptotic convergence to a constant value, for large values of d.

Aiming to test this conjecture, we solved a planning problem for a point robot moving in the d-dimensional unit hypercube containing a hyper-cubicle obstacle of measure μ (Fig. 2d). Indeed, this trend can be seen in Fig. 5a. For additional details see [23]. We repeated the experiment while using K-NN instead of R-NN queries. The results, depicted in Fig. 5b, show a growth in the ratio as a function of d by 2000%. This is not surprising, as the standard value of k that is commonly used is proportional to $\log n$, and is smaller by a factor of 2^d than the expected number of neighbors returned by an R-NN query.

The effect of the geometric complexity m of the obstacles Recall that a collision query may take $O(m^2)$ in the worst case. For small values of m, this becomes negligible with respect to other primitive operations, such as NN. In order to demonstrate this effect we ran the following experiment which is based on the 2D Random polygons scenario (Fig. 2b). We created two sequences of increasing geometric-complexity (growing m) environments. Each sequence was constructed as follows: we start with the empty environment and incrementally add random polygons from Fig. 2b until all the polygons in Fig. 2b have been added. We then placed *eight* robots that need to change their positions, and ran LBT-RRT (with approximation factor $\varepsilon = 0.4$) for a fixed N. Fig. 6 plots $\chi_{\text{LBT-RRT}}(N)$ as a

[6] Here, we assume, as is common in the literature, that the cost of *finding* the set of NN grows with the dimension.

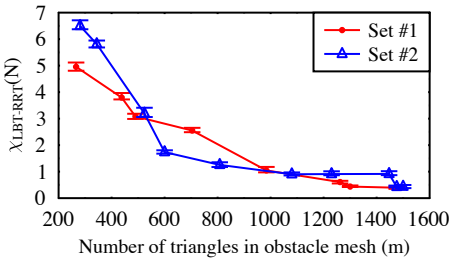


Fig. 6: $\chi_{\text{LBT-RRT}}(\mathsf{N})$ as a function of m in workspaces of increasing obstacle complexity based on the 2D Random polygons scenario (Fig. 2b). The presented plots are for $N = 4\mathsf{K}$ iterations in two different randomly-generated experiment sets.

m	$\#_{\text{NN}}$	$\#_{\text{R-NN}}$	$\#_{\text{CD}}$	$\#_{\text{LP}}$	$\#_{\text{CD in LP}}$
266	4K	2.5K	4K	8.1K	496K
704	4K	1.7K	4K	7.7K	352K
1,262	4K	0.99K	4K	24.5K	271K
1,476	4K	0.7K	4K	28.7K	232K

Table 2: Average number of calls for the main primitive operations for different values of m (chosen arbitrarily).

function of m for two sets of environments. As anticipated, the ratio in both sets of environments decays polynomially as m grows. See [23] for further discussion on the results.

4.4 NN-sensitive parameters

In all planning algorithms, one of the critical user-defined parameters, is the step size (STEP); see Sec. 3. Using STEP which is too small may cause LP to be over-conservative and costly. Choosing larger values which are still appropriate for the scenario at hand allows to decrease the portion of time spent on CD checks.

We demonstrate how RRT* becomes NN-sensitive under certain step-size values. We ran RRT* for $N = 25\mathsf{K}$ iterations on the 3D Cubicles scenario (Fig. 2c). In order to modify the step size in OMPL, one needs to specify a state validity-checking resolution. This value, which we denote by RES, is specified as a fraction of the space's extent, that is, the maximal possible distance between two configurations in the C-space. Using larger values of RES may yield paths that are invalid. Thus, when increasing RES, we also used a model of the robot which was inflated accordingly to ensure that all paths are collision free (see [25, Ch.5.3.4]). A linear correlation between RES and $\chi_{\text{RRT}^*}(N)$ was obtained. The ratio has increased from 0.05 for the default OMPL value of 1% to 0.4 for RES=10%, which was the largest value yielding valid paths. See [23] for more details.

5 Asymptotically-optimal motion-planning using R-NN

In this section we address an existing gap in the literature of sampling-based MP algorithms: How to use Eq. 1 in non-Euclidean spaces, which are prevalent in motion planning. Specifically, we derive closed-form expressions for the volume of the unit ball in several common C-spaces and distance metrics and discuss how to effectively use this value.

Closing this gap allows to evaluate the connection scheme of an algorithm. Namely, should one choose connections using R-NN or K-NN. In NN-sensitive

settings this choice may have a dramatic effect on the performance of the algorithm since (i) the number of reported neighbors may differ and (ii) the cost of the two query types for a certain NN data structure may be different. Indeed, we show empirically that there are scenarios where using R-NN, a solution path of a specific cost is produced ten-fold (and more) faster than with K-NN.

Due to lack of space, some of the technical details and experiments appear only in the extended version of this paper [23].

5.1 Well-behaved spaces and the volume of balls

Recall that $\mathcal{X}$ denotes a C-space and that given a set $A \subseteq \mathcal{X}$, $\mu(A)$ denotes the Lebesgue measure of A. Let $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ denote a distance metric and let $\mathcal{B}^{\rho}_{\mathcal{X}}(r,x) := \{y \in \mathcal{X} | \rho(x,y) \leq r\}$ and $\mathcal{S}^{\rho}_{\mathcal{X}}(r,x) := \{y \in \mathcal{X} | \rho(x,y) = r\}$ denote the ball and sphere of radius r (defined using ρ) centered at $x \in \mathcal{X}$, respectively. Finally, let $\mathbb{B}^{\rho}_{\mathcal{X}}(r) := \mu\left(\mathcal{B}^{\rho}_{\mathcal{X}}(r,0)\right)$ and $\mathbb{S}^{\rho}_{\mathcal{X}}(r) := \mu\left(\mathcal{S}^{\rho}_{\mathcal{X}}(r,0)\right)$. We will often omit the superscript ρ or the subscript $\mathcal{X}$ when they will be clear from the context.

We now define the notion of a *well-behaved* space in the context of metrics; for a detailed discussion on well-behaved spaces see [27]. In such spaces there is a derivative relationship between $\mathbb{S}(r)$ and $\mathbb{B}(r)$. Formally,

Definition 3 *A space $\mathcal{X}$ is well behaved when $\frac{\partial \mathbb{B}_{\mathcal{X}}(r)}{\partial r} = \mathbb{S}_{\mathcal{X}}(r)$. Conversely, we say that $\mathcal{X}$ is well behaved when $\int_{\varrho \in [0,r]} \mathbb{S}_{\mathcal{X}}(\varrho) d\varrho = \mathbb{B}_{\mathcal{X}}(r)$.*

We continue with the definition of a *compound space* which is the Cartesian product of two spaces. Let $\mathcal{X}_1, \mathcal{X}_2$ be two C-spaces with distance metrics ρ_1, ρ_2, respectively. Define $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$ to be their compound space. We adopt a common way to define the (weighted) distance metric over $\mathcal{X}$, when using weights $w_1, w_2 \in \mathbb{R}^+$ and some constant p [25, Chapter 5]: $\rho_{\mathcal{X}} = \left(w_1 \rho_1^p + w_2 \rho_2^p\right)^{1/p}$.[7]

The following Lemma states that the volume of balls in a compound space $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$ where $\mathcal{X}_1$ is well behaved can be expressed analytically.

Lemma 4 *Following the above notation, if $\mathcal{X}_1$ is well behaved then*

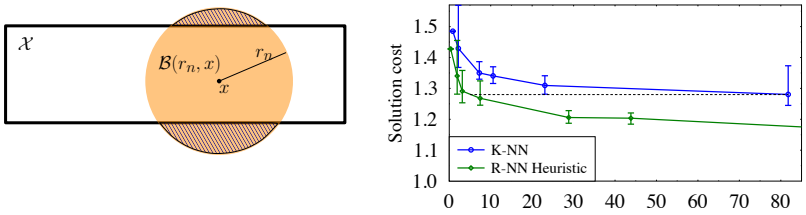
$$\mathbb{B}_{\mathcal{X}_1 \times \mathcal{X}_2}(r) = \int_{\varrho \in [0, r/w_1^{1/p}]} \mathbb{S}_{\mathcal{X}_1}(\varrho) \cdot \mathbb{B}_{\mathcal{X}_2}\left(\left(\frac{r^p - w_1 \varrho^p}{w_2}\right)^{1/p}\right) d\varrho. \tag{3}$$

Proof. By definition, $\mathbb{B}_{\mathcal{X}}(r) = \int_{x \in \mathcal{B}_{\mathcal{X}}(r)} dx$. Using Fubini's Theorem [28],

$$\mathbb{B}_{\mathcal{X}_1 \times \mathcal{X}_2}(r) = \int_{x_1 \in \mathcal{B}_{\mathcal{X}_1}(r/w_1^{1/p})} \left(\int_{x_2 \in \mathcal{B}_{\mathcal{X}_2}\left(\left(\frac{r^p - w_1 x_1^p}{w_2}\right)^{1/p}\right)} dx_2 \right) dx_1.$$

The inner integral is simply the volume of a ball of radius $\left(\frac{r^p - w_1 x_1^p}{w_2}\right)^{1/p}$ in $\mathcal{X}_2$. In addition, we know that $\mathcal{X}_1$ is well behaved, thus $\mathbb{B}_{\mathcal{X}_1}(r) = \int_{x_1 \in \mathcal{B}_{\mathcal{X}_1}(r)} dx =$

[7] This distance metric is often used due to its computational efficiency and simplicity. However, alternative methods exist, which exhibit favorable properties such as invariance to rotation of the reference frame; see, e.g., [8].



(a) **(b)**

Fig. 7: R-NN heuristic (a) A two-dimensional C-space for which Assumption 5 does not hold. (b) Solution cost (optimal cost has a value of 1) as a function of time for Lazy-sPRM* running in the Cubicles scenario (Fig. 2c). The dashed line visualizes the difference in time for obtaining a solution of cost 1.28 between R-NN heuristic and K-NN.

$\int_{\varrho\in[0,r]} \mathbb{S}_{\mathcal{X}_1}(\varrho)d\varrho$. By changing the integration variable, substituting the inner integral and using the fact that $\mathcal{X}_1$ is well behaved we obtain Eq. 3. □

For a full list of C-spaces for which a closed-form expression for $\mathbb{B}_{\mathcal{X}}(r)$ was derived, we refer the reader to the extended version of this paper [23].

5.2 Effective use of R-NN in MP algorithms in practice

We now discuss how to effectively use the radial connection scheme of asymptotically (near) optimal MP algorithms. We first describe a common scenario for which the computed radii are practically useless, and continue by suggesting a simple heuristic to overcome this problem.

The proofs of asymptotic optimality provided by Karaman and Frazzoli [21] and by Janson et al. [20] rely on the following implicit assumption:

Assumption 5 *For $x \in \mathcal{X}$, w.h.p. $\mathbb{B}_{\mathcal{X}}(r, x) = \mu(\mathcal{B}_{\mathcal{X}}(r, x) \cap \mathcal{X})$.*

This assumption does not hold when the center of the ball is close to the boundary of the C-space $\mathcal{X}$. However, since the proofs consider balls of radii proportional to r_n which tends to zero as $n \to \infty$ (as in Eq. 1), there exists a value n_0 for which r_{n_0} is sufficiently small such that the sequence of balls of radius r_{n_0} covering a given solution path does not intersect the boundary of $\mathcal{X}$.

In many common settings, Assumption 5 does not hold for practical values of n. Consider, for example, Fig. 7a, which depicts a two-dimensional rectangular C-space where one dimension is significantly larger than the other. For small values of n, any ball of radius r_n intersects the boundary of $\mathcal{X}$ (as the ball $\mathcal{B}(r_n, x)$, drawn in orange, for which $\mathcal{B}(r_n, x) \setminus \mathcal{X} \neq \emptyset$). As a result, the number of configurations of distance at most r_n from a configuration x might be too small, and this, in turn, may cause the roadmap of n vertices to remain disconnected.

We start by formally describing the setting and propose a heuristic to choose a larger radius. Let $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$ be a d-dimensional compound C-space and assume

that $\mu(\mathcal{X}_1) \geq \mu(\mathcal{X}_2)$. Let d_1, d_2 denote the dimensions of $\mathcal{X}_1, \mathcal{X}_2$, respectively. Finally, let $\rho_{\max(\mathcal{X}_2)}$ be the maximal distance between any two points in $\mathcal{X}_2$ and assume that $\rho = w_1\rho_1 + w_2\rho_2$. When Assumption 5 does not hold, as in Fig. 7, the intuition is that the "effective dimension" of our C-space is closer to d_1 than to $d_1 + d_2$. If $r_n > w_2\rho_{\max(\mathcal{X}_2)}$ then $\forall x \in \mathcal{X}\ \mathbb{B}_{\mathcal{X}}(r_n, x) > \mu(\mathcal{B}_{\mathcal{X}}(r_n, x) \cap \mathcal{X})$. In such cases, we suggest to project all points to $\mathcal{X}_1$ and use the critical connection radius that we would have used had the planning occurred in $\mathcal{X}_1$.

To evaluate the proposed heuristic, we ran Lazy-sPRM* on the Cubicles scenario (Fig. 2c) using R-NN with and without the heuristic, and also using K-NN strategy (with the standard k_n value). We measured the cost of the solution path as a function of the running time. As depicted in Fig. 7b, the heuristic was able to find higher-quality solutions using less samples, resulting in a ten-fold speedup in obtaining a solution of a certain cost, when compared to K-NN. Moreover, R-NN without the heuristic was practically inferior, as it was not able to find a solution even for large values of n; results omitted.

References

[1] Arslan, O., Tsiotras, P.: Use of relaxation methods in sampling-based algorithms for optimal motion planning. In: ICRA. pp. 2413–2420 (2013)

[2] Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. Journal of the ACM 45(6), 891–923 (1998)

[3] Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM 18, 509–517 (1975)

[4] de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer-Verlag, 3rd edn. (2008)

[5] Bialkowski, J., Otte, M.W., Karaman, S., Frazzoli, E.: Efficient collision checking in sampling-based motion planning via safety certificates. I. J. Robotic Res. 35(7), 767–796 (2016)

[6] Bohlin, R., Kavraki, L.E.: Path planning using lazy PRM. In: ICRA. pp. 521–528 (2000)

[7] Brin, S.: Near neighbor search in large metric spaces. In: VLDB. pp. 574–584 (1995)

[8] Chirikjian, G.S., Zhou, S.: Metrics on motion and deformation of solid models. J. Mech. Des. 120(2), 252–261 (1998)

[9] Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementation. MIT Press (June 2005)

[10] Cohen, J.D., Lin, M.C., Manocha, D., Ponamgi, M.: I-COLLIDE: an interactive and exact collision detection system for large-scale environments. In: Symposium on Interactive 3D Graphics. pp. 189–196, 218 (1995)

[11] Şucan, I.A., Moll, M., Kavraki, L.E.: The Open Motion Planning Library. IEEE Robotics & Automation Magazine 19(4), 72–82 (2012)

[12] Friedman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Softw. 3(3), 209–226 (1977)

[13] Gammell, J.D., Srinivasa, S.S., Barfoot, T.D.: Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In: IROS. pp. 2997–3004 (2014)

[14] Hauser, K.: Lazy collision checking in asymptotically-optimal motion planning. In: ICRA. pp. 2951–2957 (2015)
[15] Hemmer, M., Kleinbort, M., Halperin, D.: Optimal randomized incremental construction for guaranteed logarithmic planar point location. Comput. Geom. 58, 110–123 (2016)
[16] Hubbard, P.M.: Approximating polyhedra with spheres for time-critical collision detection. ACM Trans. Graph. 15(3), 179–210 (1996)
[17] Ichnowski, J., Alterovitz, R.: Fast nearest neighbor search in SE(3) for sampling-based motion planning. In: WAFR. pp. 197–214 (2014)
[18] Indyk, P.: Nearest neighbors in high-dimensional spaces. In: Goodman, J.E., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, chap. 39, pp. 877–892. CRC Press LLC, Boca Raton, FL, 2nd edn. (2004)
[19] Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: STOC. pp. 604–613 (1998)
[20] Janson, L., Schmerling, E., Clark, A.A., Pavone, M.: Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. I. J. Robotic Res. 34(7), 883–921 (2015)
[21] Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. I. J. Robotic Res. 30(7), 846–894 (2011)
[22] Kleinbort, M., Salzman, O., Halperin, D.: Efficient high-quality motion planning by fast all-pairs r-nearest-neighbors. In: ICRA. pp. 2985–2990 (2015)
[23] Kleinbort, M., Salzman, O., Halperin, D.: Collision detection or nearest-neighbor search? On the computational bottleneck in sampling-based motion planning. CoRR abs/1607.04800 (2016)
[24] Larsen, E., Gottschalk, S., Lin, M.C., Manocha, D.: Fast proximity queries with swept sphere volumes. Tech. rep., Department of Computer Science, University of North Carolina (1999), TR99-018
[25] LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
[26] Lin, M.C., Manocha, D.: Collision and proximity queries. In: Goodman, J.E., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, chap. 35, pp. 767–786. CRC Press LLC, Boca Raton, FL, 2nd edn. (2004)
[27] Marichal, J.L., Dorff, M.: Derivative relationships between volume and surface area of compact regions in $\mathbb{R}^d$. Rocky Mountain J. Math. 37(2), 551–571 (2007)
[28] Mazzola, G., Milmeister, G., Weissmann, J.: Comprehensive Mathematics for Computer Scientists 2, chap. 31, pp. 87–95. Springer, Berlin, Heidelberg (2005)
[29] Pan, J., Chitta, S., Manocha, D.: FCL: A general purpose library for collision and proximity queries. In: ICRA. pp. 3859–3866 (2012)
[30] Plaku, E., Kavraki, L.E.: Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning. In: WAFR. pp. 3–18 (2006)
[31] Salzman, O., Halperin, D.: Asymptotically-optimal motion planning using lower bounds on cost. In: ICRA. pp. 4167–4172 (2015)
[32] Salzman, O., Halperin, D.: Asymptotically near-optimal RRT for fast, high-quality, motion planning. IEEE Trans. Robotics 32(3), 473–483 (2016)
[33] Solovey, K., Salzman, O., Halperin, D.: New perspective on sampling-based motion planning via random geometric graphs. Robots Science and Systems (RSS) (2016)
[34] Yershova, A., LaValle, S.M.: Improving motion-planning algorithms by efficient nearest-neighbor searching. IEEE Trans. Robotics 23(1), 151–157 (2007)