Inter Workshop on Algorithmic Foundations of Robotics (WAFR), New York, NY, 2006

Quantitative Analysis of Nearest-Neighbors Search in High-Dimensional Sampling-Based Motion Planning

Erion Plaku and Lydia E. Kavraki

Department of Computer Science, Rice University
{plakue, kavraki}@cs.rice.edu

Abstract: We quantitatively analyze the performance of exact and approximate nearest-neighbors algorithms on increasingly high-dimensional problems in the context of sampling-based motion planning. We study the impact of the dimension, number of samples, distance metrics, and sampling schemes on the efficiency and accuracy of nearest-neighbors algorithms. Efficiency measures computation time and accuracy indicates similarity between exact and approximate nearest neighbors.

Our analysis indicates that after a critical dimension, which varies between 15 and 30, exact nearest-neighbors algorithms examine almost all the samples. As a result, exact nearest-neighbors algorithms become impractical for sampling-based motion planners when a considerably large number of samples needs to be generated. The impracticality of exact nearest-neighbors algorithms motivates the use of approximate algorithms, which trade off accuracy for efficiency. We propose a simple algorithm, termed Distance-based Projection onto Euclidean Space (DPES), which computes approximate nearest neighbors by using a distance-based projection of high-dimensional metric spaces onto low-dimensional Euclidean spaces. Our results indicate DPES achieves high efficiency and only a negligible loss in accuracy.

1 Introduction

Research in motion planning has in recent years focused on sampling-based algorithms [1,5,9,13,14,17,20,22] for solving problems involving multiple and highly complex robots. Such algorithms rely on an efficient sampling of the configuration space and compute nearest neighbors for the sampled points. In general, the k nearest neighbors of a point in a data set are defined as the k closest points in the data set according to a distance metric.

As research in motion planning progressively addresses problems of unprecedented complexity, nearest-neighbors computations based on arbitrary distance metrics and large high-dimensional data sets become increasingly challenging. Researchers have developed many nearest-neighbors algorithms, such as the kd-tree, R-tree, X-tree, M-tree, VP-tree, Gnat, iDistance, surveyed

in [7,10,11], and others [3]. Analysis has shown that for certain distance metrics and data distributions, the computational efficiency of such algorithms decreases as the dimension increases [4,6,12,18,23]. As summarized in [15], although the nearest neighbor of a point according to L_2 from an n-point d-dimensional data set can be computed in $O(d^{O(1)} \log n)$ time, the associated $n^{O(d)}$ space requirement is impractical. Reducing the space requirement to what is practical, i.e., $O(dn)$, increases the query time to $\min(2^{O(d)}, dn)$. In fact, after a critical dimension, the brute-force linear method, which examines the entire data set, is computationally faster than other exact nearest-neighbors algorithms. The work in [23] shows 610 as the theoretical bound on the critical dimension for L_2 and uniformly distributed points in $[0,1]^d$. The experiments in [23] however indicate 10 as the critical dimension, a much lower estimate than the theoretical bound. In [4,12], the critical dimension is estimated between 10–15 for L_p and several synthetic and image data sets.

Another viable approach for nearest-neighbors computations is to use approximate algorithms which trade off accuracy for efficiency [2, 10, 19, 21], where accuracy indicates similarity between exact and approximate nearest neighbors. As summarized in [15], in the case of L_2, approximate nearest neighbors can be computed probabilistically in $dn^{1/1+\epsilon}$ time and $O(dn)$ space or deterministically in $(d \log n/\epsilon)^{O(1)}$ time and $n^{1/\epsilon^{O(1)}}$ space. Such algorithms gain efficiency by projecting the data set onto low-dimensional spaces and achieve high accuracy when the projection results in low distortion of distances. The computational advantages of approximate nearest-neighbors algorithms are more evident when the dimension d of the data set is high. The problem however remains challenging for general metrics. As summarized in [16], any n-point metric space can be projected onto $\mathbb{R}^{O(\log^2 n)}$ with only $O(\log n)$ distortion. However, solving high-dimensional motion planning problems requires generating millions of samples, which makes $O(\log^2 n)$ impractical. By increasing the distortion to $O(n^{2/d} \log^{3/2} n)$, and thus reducing the accuracy, any n-point d-dimensional metric space could be projected onto $\mathbb{R}^d$. Therefore, the efficiency or accuracy of approximate nearest-neighbors algorithms is typically reduced when general metrics are used instead of L_2.

The analysis of nearest-neighbors algorithms generally assume a uniform distribution of points and the use of L_2. In motion planning, the distribution is impacted by the sampling scheme. Samples satisfy certain criteria, such as representing collision-free configurations, and, consequently, the distribution is usually non-uniform. Furthermore, distances between configurations are not necessarily defined by L_2, but instead attempt to capture the success of the local planner. Motion planners therefore exhibit a degree of flexibility which can be exploited to compute approximate instead of exact nearest neighbors. Research in [22] shows that in certain high-dimensional problems using random neighbors actually improves the performance of motion planners. Therefore, an understanding of the impact of these factors on the efficiency and accuracy

of nearest-neighbors algorithms employed by motion planners could provide valuable insight in addressing high-dimensional motion planning problems.

In this work, we quantitatively analyze exact and approximate nearest-neighbors algorithms in the context of high-dimensional sampling-based motion planning. We focus on roadmap-based algorithms, such as the Probabilistic RoadMap (`PRM`) method with uniform [17], bridge [13], Gaussian [5], and obstacle [1] sampling, and tree-based algorithms, such as the Rapidly-exploring Random Tree (`RRT`) [20] and the Expansive-Spaces Tree (`EST`) [14].

We address the following questions: (i) under what conditions, if any, should motion planners use the brute-force linear method instead of other exact nearest-neighbors algorithms? (ii) do approximate nearest-neighbors algorithms compute more efficiently nearest neighbors that are similar to exact nearest neighbors on high-dimensional motion planning problems? We study the impact of the dimension, number of samples, distance metrics, and sampling schemes on the efficiency and accuracy of nearest-neighbors algorithms.

Our analysis indicates that after a critical dimension the brute-force linear method is computationally more efficient than other exact nearest-neighbors algorithms. The critical dimension however depends on the number of samples, distance metric, and sampling scheme. We present results that quantify these dependencies.

Motivated by the impracticality of exact nearest-neighbors algorithms, we propose the use of approximate algorithms for the computation of neighbors in high-dimensional motion planning problems. In this work, we develop a simple algorithm, termed Distance-based Projection onto Euclidean Space (DPES), which computes approximate nearest neighbors by projecting high-dimensional metric spaces onto low-dimensional Euclidean spaces. The projection is based on distances between a set of selected points and points in the data set. Our experiments indicate DPES achieves high computational efficiency and only a negligible loss in accuracy.

The rest of the paper is organized as follows. In Sect. 2 we describe the methodology we use in our analysis and the DPES algorithm. In Sect. 3 we describe the experimental setup. In Sect. 4 we present the results of our analysis of nearest-neighbors algorithms. We conclude in Sect. 5 with a discussion.

2 Methods

In this section, we describe the nearest-neighbors algorithms we use in this paper including DPES. We also outline the motion planners and distance metrics we use in this study. We denote the data set, number of nearest neighbors, and distance metric by S, k, and $\rho : S \times S \to \mathbb{R}^{\geq 0}$, respectively.

2.1 Exact k Nearest-Neighbors Algorithms

We define the k nearest neighbors (kNN) of a point $s_i \in S$, denoted by $\mathrm{NN}_S(s_i, k)$, as the k closest points to s_i from $S - \{s_i\}$ according to ρ.

Linear This is a brute-force approach which resolves $\mathrm{NN}_S(s_i, k)$ by computing the distance from s_i to each point in S. The Linear method provides the basis for the comparison with other more sophisticated kNN algorithms.

Gnat Gnat [7] constructs a tree recursively by partitioning S into smaller sets and associating each set with a branch in the tree. Gnat then uses the triangle inequality to prune certain branches of the tree in order to compute kNN more efficiently. We choose Gnat in our analysis, since the results in [7] and our experiments in the context of motion planning with several kNN algorithms, such as kd-tree, M-tree, VP-tree, [3], etc., indicate Gnat to be more efficient especially on large data sets and metric spaces.

2.2 Approximate k Nearest-Neighbors Algorithms

We define approximate k nearest neighbors (kANN) of a point $s_i \in S$, denoted by $\mathrm{ANN}_S(s_i, k)$, as a subset of $S - \{s_i\}$ of cardinality k that according to certain measures is similar to $\mathrm{NN}_S(s_i, k)$.

Random This method selects $S' \subset S$ uniformly at random, $|S'| \gg k$, and computes $\mathrm{ANN}_S(s_i, k)$ as $\mathrm{NN}_{S'}(s_i, k)$. Random provides a basis for evaluating the quality of other kANN algorithms.

Distance-based Projection onto Euclidean Space (DPES) Our kANN algorithm is based on projecting each point $s_i \in S$ to a point $v(s_i) \in \mathbb{R}^m$, for some fixed $m > 0$. We then use L_2 to define distances between projected points and compute $\mathrm{ANN}_S(s_i, k)$ as

$$\mathrm{ANN}_S(s_i, k) = \{s' : v(s') \in \mathrm{NN}_{V(S)}(v(s_i), k)\}, \quad V(S) = \{v(s_i) : s_i \in S\}.$$

We thus compute $\mathrm{ANN}_S(s_i, k)$ according to the distance metric ρ by computing $\mathrm{NN}_{V(S)}(v(s_i), k)$ according to L_2. Any kNN data structure $\mathcal{A}$ can be used to compute $\mathrm{NN}_{V(S)}(v(s_i), k)$. DPES supports dynamic addition and removal of points. When a point s is added to or removed from S, the corresponding projection $v(s)$ is added to or removed from $\mathcal{A}$, respectively.

We obtain the projection by selecting m pivots $\{p_1, p_2, \cdots, p_m\} \subset S$ and setting each $v(s_i) \in \mathbb{R}^m$ to $v(s_i)[j] = \rho(s_i, p_j)$, $1 \leq j \leq m$. We select p_1 uniformly at random in S and each p_j, $2 \leq j \leq m$, as the point in S that maximizes $\min_{i=1}^{j-1} \rho(p_i, p_j)$. The objective is to select pivots that preserve relative distances between points in S when projected onto $\mathbb{R}^m$, e.g., projections of points in S that are close according to ρ should be close according to L_2.

DPES has certain computational advantages. The projection of S onto $\mathbb{R}^m$ greatly improves the efficiency, since, as shown in Sect. 4.2, typically fewer distance evaluations are necessary for computing nearest neighbors. In addition, evaluating L_2 is more efficient than evaluating distance metrics commonly used in motion planning, such as those in Sect. 2.3.

Quality Evaluation We determine the quality of $\mathrm{ANN}_S(s_i, k)$ by using two common measures based on distances between points in $\mathrm{ANN}_S(s_i, k)$ and $\mathrm{NN}_S(s_i, k)$. Similar to [10], we use the ratio of false dismissals:

$$\text{rfd}_\epsilon = \frac{1}{k} \sum_{s \in \text{ANN}_S(s_i,k)} \begin{cases} 1, & \rho(s, s_i) > (1+\epsilon) \max_{s' \in \text{NN}_S(s_i,k)} \rho(s_i, s'), \\ 0, & \text{otherwise.} \end{cases}$$

The rfd_ϵ error, $\epsilon \geq 0$, indicates the fraction of points in $\text{ANN}_S(s_i, k)$ that are $(1+\epsilon)$-times farther away from the k-th nearest neighbor of s_i. Note however that some $s', s'' \in \text{ANN}_S(s_i, k)$ could contribute the same value to rfd_ϵ even when $\rho(s', s_i) \gg \rho(s'', s_i)$. Thus, two different sets could have the same rfd_ϵ value even when points in one set are farther away from s_i than points in the other set. Therefore, as in [10], we also use the ratio of distance errors:

$$\text{rde} = 1 - \sum_{s \in \text{NN}_S(s_i,k)} \rho(s, s_i) / \sum_{s \in \text{ANN}_S(s_i,k)} \rho(s, s_i).$$

The range of rfd_ϵ and rde is $[0, 1]$ and smaller values indicate high quality.

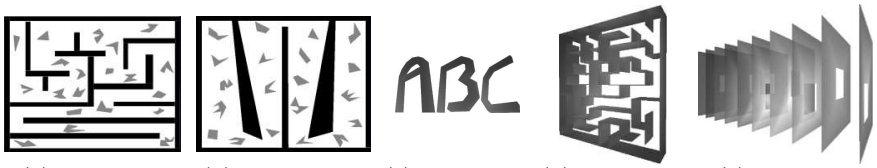
2.3 Sampling-Based Motion Planning and Distance Metrics

In this study, we use roadmap-based algorithms, such as `PRM` with uniform (`PRMu`) [17], bridge (`PRMb`) [13], Gaussian (`PRMg`) [5], and obstacle (`PRMo`) [1] sampling, and tree-based algorithms, such as bi-directional `RRT` [20] and `EST` [14]. We follow standard implementations as in [9, 22]. We consider problems with multiple robots moving freely in 2D or 3D workspaces with static obstacles. We gradually increase the number of robots until we reach the critical dimension. We create data sets using configurations of the roadmap in `PRM` and the initial tree in `RRT` and `EST`.

In 2D workspaces, we use $\rho_{\text{SE(2)}}$, the geodesic distance in SE(2) [8], as the distance between any two single robot configurations a and b, i.e., length of shortest path in SE(2) from a to b. We also use $\rho_{w\text{SE(2)}}$, which weighs, as discussed below, the geodesic distances in $\mathbb{R}^2$ and SO(2). Similarly, in 3D workspaces, we use $\rho_{\text{SE(3)}}$ [8], the geodesic distance in SE(3), and $\rho_{w\text{SE(3)}}$, which weighs the geodesic distances in $\mathbb{R}^3$ and SO(3). We also use ρ_{L_2}, which approximates the volume of the workspace region swept by the robot [9] . We experimented with several weighting schemes for $\rho_{w\text{SE(2)}}$ and $\rho_{w\text{SE(3)}}$, but found little variation in the results of nearest-neighbors algorithms. Therefore in this study we set the weights to one. In the case of multiple robots, we sum up $\rho_{\text{SE(2)}}$, $\rho_{w\text{SE(2)}}$, $\rho_{\text{SE(3)}}$, $\rho_{w\text{SE(3)}}$, and ρ_{L_2} distances between configurations for each robot to obtain $\rho^*_{\text{SE(2)}}$, $\rho^*_{w\text{SE(2)}}$, $\rho^*_{\text{SE(3)}}$, $\rho^*_{w\text{SE(3)}}$, and $\rho^*_{\text{L}_2}$, respectively.

3 Experimental Setup

Data Sets We use 2D and 3D workspaces, shown in Fig. 1, that provide a representative benchmark for motion planners. The "maze2d" workspace is a 2D maze, as in Fig. 1(a). Robots must move from one of the borders of the maze to the opposite border. The "narrow2d" workspace has several narrow



(a) "maze2d" (b) "narrow2d" (c) robots3d (d) "maze3d" (e) "cons3d"

Fig. 1. Workspaces. (a), (b) The black and gray polygons indicate obstacles and robots, respectively. (c),(d),(e) In the 3D workspaces, robots consist of 3D renderings of English letters and the i-th robot corresponds to the i-th letter.

Table 1. Summary of data sets.

motion planner	`PRM`[uniform, bridge, Gaussian, obstacle], `RRT`, `EST`	
number of points (n)	10000, 50000, 100000	
workspace	maze2d, narrow2d	maze3d, cons3d
distance	$\rho^*_{\mathrm{SE}(2)}$, $\rho^*_{w\mathrm{SE}(2)}$	$\rho^*_{\mathrm{SE}(3)}$, $\rho^*_{w\mathrm{SE}(3)}$, $\rho^*_{\mathrm{L}_2}$
dimension (d)	$3, 6, 9, \ldots, 60$	$6, 12, 18, \ldots, 60$

passages, as in Fig. 1(b). Robots must move from the left side to the right side of the box. In the 3D workspaces, robots are objects in the shape of letters, as in Fig. 1(c). The "maze3d" workspace is a 3D maze, as in Fig. 1(d). Robots must move from one corner of the maze to the other and always remain inside the maze. The "cons3d" workspace has ten consecutive walls each with a small hole, as in Fig. 1(e). Robots must move through all the ten holes.

We created many data sets as summarized in Table 1. We use 1, 2, ..., 20 and 1, 2, ..., 10 robots in each 2D and 3D workspace to obtain configurations with 3, 6, ..., 60 and 6, 12, ..., 60 dimensions, respectively. As an example, a 60-dimensional "maze3d" problem is obtained by placing 10 robots, consisting of 3D renderings of letters A through I as in Fig. 1(c), in the "maze3d" workspace. We note that the choice of letters for the robots does not affect the results of our experiments. For each dimension, we generate data sets with 10000, 50000, and 100000 points. For each dimension and number of points, we use each motion planner to generate data sets using $\rho^*_{\mathrm{SE}(2)}$ and $\rho^*_{w\mathrm{SE}(2)}$ in each 2D workspace and $\rho^*_{\mathrm{SE}(3)}$, $\rho^*_{w\mathrm{SE}(3)}$, and $\rho^*_{\mathrm{L}_2}$ in each 3D workspace. During data generation, each motion planner uses Linear for the kNN computations.

Experiments For each data set, we use Gnat and Linear to compute kNN queries and DPES and Random to compute kANN queries for various values of $k \in \{15, 45, 150\}$. We report only results obtained for $k = 45$, since the results for the other values of k are similar. In each case, we measure the time and distance evaluations required to compute nearest neighbors of a point $s \in S$ selected uniformly at random. In addition, for kANN algorithms, we measure the rfd_ϵ, $\epsilon \in \{0.00, 0.05, 0.10\}$, and rde errors. We obtain averages of these quantities by repeating the above step 100 times. We choose $|S'|$ such that the running time of Random is the same as that of DPES.

Platform We utilized three high-performance computing clusters, Rice Terascale Cluster, PBC Cluster, and Rice Cray XD1 Cluster ADA.

4 Results

We compare the computational efficiency of Gnat and DPES relative to Linear for kNN and kANN computations, respectively. We also focus on the accuracy of DPES and Random. The use of Linear and Random provides a normalization of the results obtained for data sets generated using various motion planners, distance metrics, number of points, dimensions, and workspaces. We present results for "maze2d" and "maze3d" workspaces, since the correlation with results for "narrow2d" and "cons3d" workspaces is above 90%.

4.1 Exact k Nearest-Neighbors Algorithms

We compare Gnat to Linear for various distance metrics.

Using $\rho^*_{\mathbf{SE(2)}}$ We present the results in Fig. 2. These results are indicative of other distances and illustrate general trends observed in kNN algorithms. We indicate the workspace, motion planner, distance metric, and number of points at the top and legend of each figure. In Fig. 2(a), we compare the computation time of Gnat relative to Linear on data sets generated using `PRM` with uniform sampling. We observe that Gnat is more efficient than Linear on low-dimensional data sets. The efficiency of Gnat increases even more when the number of points increases. However, as the dimension increases, the efficiency of Gnat deteriorates rapidly. In fact, after a certain dimension, $d > 18$, Gnat is even less efficient than Linear.

In Fig. 2(b), we focus on the number of distance evaluations. We observe trends similar to Fig. 2(a). We note that Gnat evaluates far fewer distances than Linear on low-dimensional data sets, especially on large low-dimensional data sets. However, as in Fig. 2(a), the number of distance evaluations by Gnat relative to Linear increases rapidly with the dimension and even approaches 1.0 when $d > 18$. Since Gnat has more computational overhead than Linear, we observed in Fig. 2(a), that for $d > 18$, Gnat is less efficient than Linear.

In Fig. 2(c), we compare the computation time of Gnat relative to Linear on data sets with $n = 100000$ points generated using `PRM` with different sampling schemes. We observe that Gnat is unable to take advantage of the different distributions that result from changing the sampling in `PRM`. There is almost no variation in the efficiency of Gnat when the sampling in `PRM` is changed from uniform to bridge, Gaussian, or obstacle. Similar observations also hold for the smaller data sets.

In Fig. 2(d) and (e), we focus on `RRT` and `EST`, respectively. As in Fig. 2(a), the efficiency of Gnat relative to Linear is significantly better on low-dimensional data sets, but quickly deteriorates as the dimension increases,

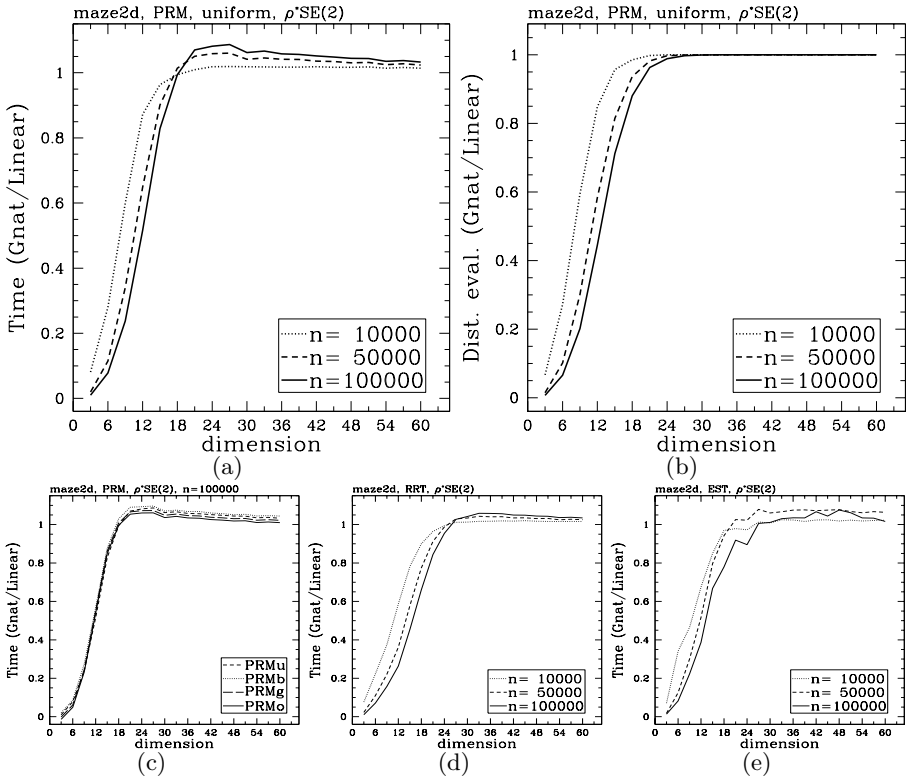


Fig. 2. Comparing Gnat to Linear when using $\rho^*_{\mathrm{SE}(2)}$.

and becomes worse after the critical dimension. However, the critical dimension is higher for `RRT` and `EST` than for `PRM` due to the local nature of `RRT` and `EST` which create samples that are more distinctly clustered and, consequently, can be used by Gnat to eliminate certain distance computations.

Using $\rho^*_{w\mathbf{SE(2)}}$, $\rho^*_{\mathbf{SE(3)}}$, $\rho^*_{w\mathbf{SE(3)}}$, and $\rho^*_{\mathbf{L_2}}$ We present the results in Fig. 3. We compare the computation time of Gnat relative to Linear. The filled region indicates the variation in the efficiency of Gnat for the different `PRM` versions, while the dashed and dotted lines indicate the results obtained for `EST` and `RRT`, respectively. We show results only for data sets with $n = 100000$ points, since we obtain similar results for the smaller data sets.

In Fig. 3(a), we focus on $\rho^*_{w\mathrm{SE}(2)}$. As in the case of $\rho^*_{\mathrm{SE}(2)}$, the efficiency of Gnat relative to Linear is at least one order of magnitude better on low-dimensional data sets, but rapidly decreases with the dimension. We note there is almost no variation in the efficiency of Gnat when the sampling in `PRM` is changed from uniform to bridge, Gaussian, or obstacle.

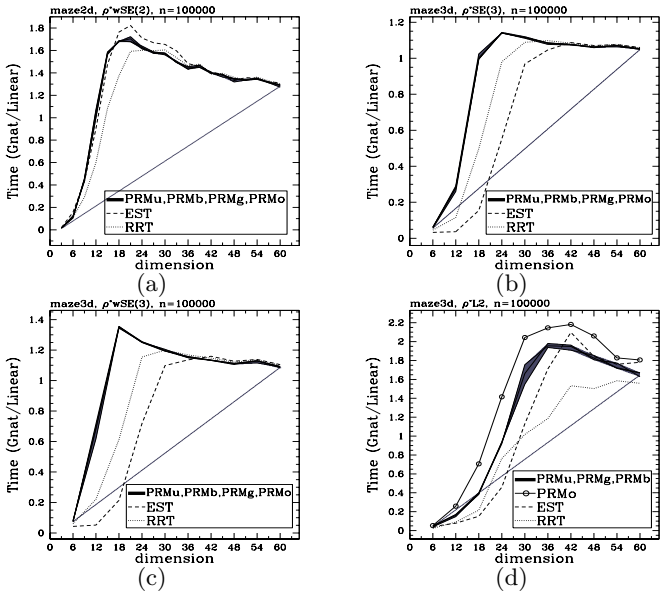


Fig. 3. Comparing Gnat to Linear when using $\rho^*_{w\mathrm{SE}(2)}$, $\rho^*_{\mathrm{SE}(3)}$, $\rho^*_{w\mathrm{SE}(3)}$, and $\rho^*_{\mathrm{L}_2}$.

Similarly, in Fig. 3(b) and (c), we observe that when using $\rho^*_{\mathrm{SE}(3)}$ or $\rho^*_{w\mathrm{SE}(3)}$, the efficiency of Gnat remains the same for `PRM` variants, but improves for `RRT` and `EST` due to the local sampling.

In Fig. 3(d), we focus on $\rho^*_{\mathrm{L}_2}$. The efficiency of Gnat remains the same when `PRM` uses uniform, bridge, or obstacle sampling, as indicated by the small area of the shaded region, but decreases when Gaussian sampling is used. As before, due to the local sampling Gnat is more efficient when `RRT` and `EST` are used.

A comparison between Fig. 2(c, d, e) and Fig. 3(a) and between Fig. 3(b) and Fig. 3(c) indicates that in general the efficiency of Gnat is better when geodesic distances are used instead of weighted distances. Our intuition is that this is due to the decoupling of translational and rotational components which reduces the number of distinct clusters in the data set. As mentioned in Sect. 2.3, we obtained similar results for several different weighting schemes.

4.2 Approximate k Nearest-Neighbors Algorithms

In this section, we analyze the efficiency and accuracy of DPES using Linear and Random as the basis of comparison, respectively. We present results for various distance metrics. In the experiments presented in this section, DPES uses $m = 15$ pivots for the projection. These results are indicative of the behavior of DPES. In the next section, we present results where we vary m.

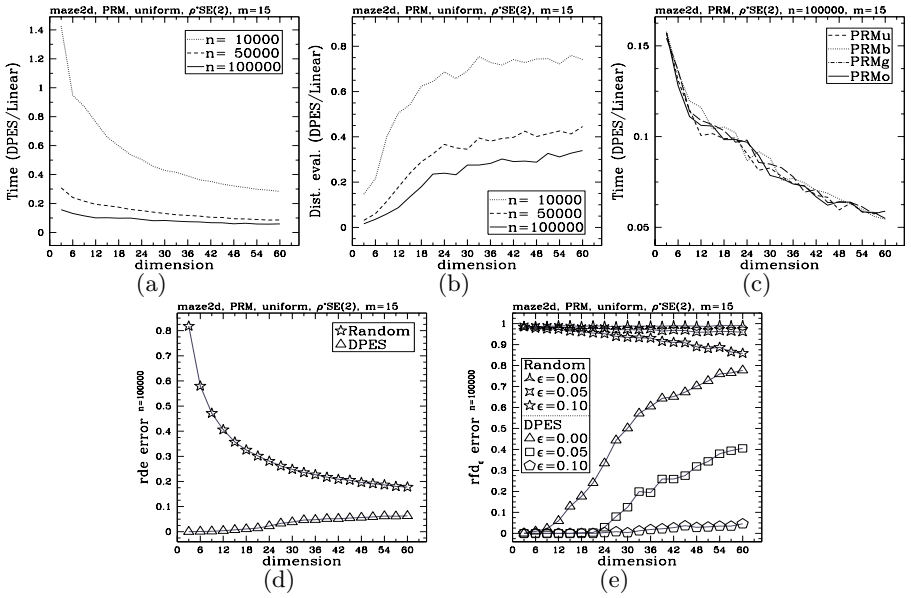


Fig. 4. kANN results for PRM and the $\rho^*_{\mathrm{SE}(2)}$ distance metric.

Using $\rho^*_{\mathrm{SE}(2)}$ We focus on PRM variants in Fig. 4 and RRT and EST in Fig. 5. In Fig. 4(a), we compare the computation time of DPES relative to Linear on data sets generated using PRM with uniform sampling. We observe that for low-dimensional and small data sets, DPES is less efficient than Linear, since data sets are projected onto $\mathbb{R}^{15}$. However, as the dimension increases, the efficiency of DPES relative to Linear improves rapidly. The improvement is even greater on the larger data sets.

In Fig. 4(b), we focus on distance evaluations for the same data sets as in Fig. 4(a). Recall that DPES uses L_2 in $\mathbb{R}^{15}$, while Linear uses $\rho^*_{\mathrm{SE}(2)}$. We note that although the number of distance evaluations by DPES relative to Linear increases with the dimension, it decreases with the number of points. In general, DPES evaluates only a fraction of distances to the query point.

In Fig. 4(c), we compare the computation time of DPES relative to Linear on data sets with $n = 100000$ points generated using different PRM variants. We observe only small changes in the computational time of DPES when the sampling in PRM is changed from uniform to bridge, Gaussian, and obstacle. We obtain similar results for the smaller data sets as well.

In Fig. 4(d), we compare the rde error of Random and DPES on data sets with $n = 100000$ points generated using PRM with uniform sampling. The rde error of Random is high on low-dimensional data sets but decreases with the dimension. This is due to the sparsity of data on high-dimensional spaces which as shown in [4] implies that the relative distances between points decrease as the dimension increases. On the other hand, the rde error of DPES

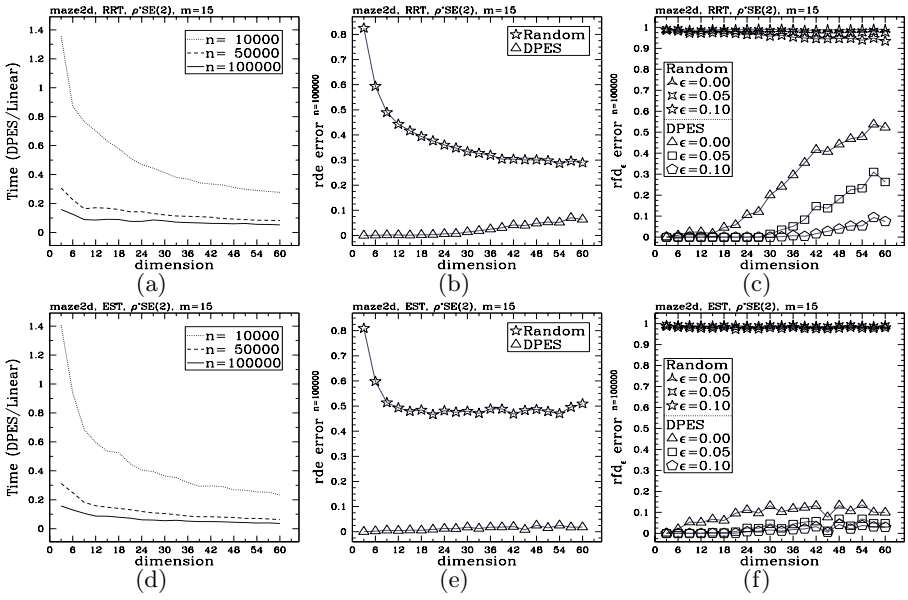


Fig. 5. kANN results for `RRT` and `EST` and the $\rho^*_{\mathrm{SE}(2)}$ distance metric.

is small on low-dimensional data sets but increases, although only slightly, with the dimension. The increase in the rde error of DPES is a consequence of the projection onto a low-dimensional Euclidean space, i.e., $\mathbb{R}^{15}$. In all cases however the rde error of DPES relative to Random is at least 2.5 times smaller.

In Fig. 4(e), we focus on the rfd error using the same data sets as in Fig. 4(d). The rfd error of Random remains very high even when the dimension increases. Such high values indicate that even though relative distances between points decrease with the dimension, as seen in Fig. 4(d), there is a clear distinction between the nearest neighbors and other points in the data set. A similar observation has also been made in [12], where it is shown that under certain conditions nearest neighbors are meaningful on high-dimensional data sets. In the case of DPES, we observe that the rfd error increases with the dimension. The increase is more rapid when $\epsilon = 0.00$. This is expected since $\mathrm{rfd}_{0.00}$ indicates how many points in $\mathrm{ANN}_S(s_i, k)$ are not in $\mathrm{NN}_S(s_i, k)$. However, as ϵ increases, the rfd error of DPES decreases significantly and for $\epsilon = 0.10$ comes close to zero even on the high-dimensional data sets.

The accuracy results in Fig. 4(d) and (e) indicate that although the approximate nearest neighbors computed by DPES are not the same as the exact nearest neighbors, the differences between them are small. In Sect. 4.3, we show how to further improve the accuracy of DPES.

In Fig. 5(a) and (d), we compare the efficiency of DPES relative to Linear on data sets generated using `RRT` and `EST`, respectively. As in the case of `PRM`

in Fig. 4(a), the relative efficiency improves rapidly with the dimension and for $d > 12$, DPES is several times faster than Linear.

In Fig. 5(b) and (e), we compare the rde error of Random and DPES on data sets with $n = 100000$ points generated using `RRT` and `EST`, respectively. We obtain similar results on the smaller data sets. In addition to the observations made for Fig. 4(d), we note that for `RRT` and especially `EST` the rde error of Random is larger while the rde error of DPES is smaller than for `PRM`. This is due to the local sampling of `RRT` and `EST`, which generate data sets where relative distances between points are more distinct, especially in the case of `EST` which expands slower than `RRT`. Consequently, the likelihood that a random point is a nearest neighbor decreases while the projection done by DPES better preserves the relative distances between points.

In Fig. 5(c) and (f), we focus on the rfd error using the same data sets as in Fig. 5(b) and (e). In addition to the observations made for Fig. 4(e), we note that the rfd error of Random remains high, while the rfd error of DPES decreases when `RRT` and especially `EST` are used instead of `PRM`. In fact, in the case of `EST`, even the $\text{rfd}_{0.00}$ error of DPES is less than 0.1, which indicates that DPES computes above 90% of the exact nearest neighbors.

Using $\rho^*_{w\text{SE(2)}}$, $\rho^*_{\text{SE(3)}}$, $\rho^*_{w\text{SE(3)}}$, and $\rho^*_{\text{L}_2}$ We summarize the results obtained for the other distance metrics in Table 2. We focus on data sets with $n = 100000$ points and $d = 60$ dimensions generated using the "maze2d" and "maze3d" workspaces. The results for the other data sets are similar. For each motion planner, we present the computation time of DPES relative to Linear and the rde and rfd_ϵ, $\epsilon \in \{0.00, 0.05, 0.10\}$, errors of DPES and Random.

As in the case of $\rho^*_{\text{SE(2)}}$, DPES is more efficient than Linear. The improvements vary between 2–4 and 12–16 times on the high-dimensional data sets, in the worst and best cases, corresponding to $\rho^*_{\text{L}_2}$ and $\rho^*_{\text{SE(3)}}$, respectively. In addition, the efficiency of DPES remains almost the same when the sampling in `PRM` is changed from uniform to bridge, Gaussian, or obstacle. However, DPES is generally more efficient in the case of `RRT` and `EST`.

We observe in Table 2 that DPES achieves high quality especially in the case of $\rho^*_{\text{L}_2}$. The small values of rde indicate that approximate nearest neighbors computed by DPES are very close to exact nearest neighbors. This is further confirmed by the small values of the rfd error of DPES for $\epsilon \geq 0.05$ when $\rho^*_{\text{L}_2}$ is used and $\epsilon \geq 0.10$ when the other distance metrics are used.

4.3 Improving the Quality of *k*ANN Queries Computed by DPES

The quality of DPES can be improved by increasing the dimension of the Euclidean space onto which data sets are projected. The results in Sect. 4.2 are obtained by using $m = 15$ pivots. In Fig. 6, we present results where we vary the number of pivots $m \in \{10, 30, 50\}$. We focus on large data sets generated using $\rho^*_{\text{SE(2)}}$ and `PRM` with uniform sampling, since we obtain similar results with the other data sets and distance metrics.

Table 2. Summary of kANN results for various distance metrics.

$n = 100000$ $d = 60$	Efficiency ($t_{\text{DPES}}/t_{\text{Linear}}$)	Accuracy (•DPES ⋆Random)								
		rde		rfd						
				ϵ=0.00		ϵ=0.05		ϵ=0.10		
		•	⋆	•	⋆	•	⋆	•	⋆	
PRMu	0.40	0.07	0.17	0.80	0.99	0.46	0.95	0.04	0.84	$\rho^*_{w\text{SE(2)}}$
PRMb	0.38	0.06	0.17	0.79	0.99	0.42	0.95	0.04	0.84	
PRMg	0.42	0.07	0.17	0.80	0.99	0.45	0.95	0.05	0.84	
PRMo	0.39	0.06	0.17	0.79	0.99	0.42	0.96	0.03	0.85	
RRT	0.36	0.06	0.28	0.54	0.99	0.30	0.97	0.06	0.93	
EST	0.24	0.03	0.49	0.14	0.99	0.08	0.98	0.06	0.98	
PRMu	0.08	0.07	0.19	0.78	0.99	0.45	0.96	0.08	0.88	$\rho^*_{\text{SE(3)}}$
PRMb	0.08	0.07	0.19	0.79	0.98	0.46	0.96	0.08	0.88	
PRMg	0.08	0.07	0.19	0.78	0.99	0.46	0.96	0.09	0.89	
PRMo	0.08	0.07	0.19	0.80	0.99	0.51	0.96	0.11	0.88	
RRT	0.06	0.05	0.25	0.55	0.99	0.20	0.97	0.02	0.94	
EST	0.07	0.06	0.25	0.58	0.99	0.27	0.97	0.06	0.94	
PRMu	0.14	0.07	0.17	0.81	0.99	0.48	0.95	0.07	0.84	$\rho^*_{w\text{SE(3)}}$
PRMb	0.15	0.07	0.17	0.81	0.99	0.49	0.96	0.08	0.85	
PRMg	0.14	0.07	0.17	0.82	0.99	0.49	0.96	0.09	0.86	
PRMo	0.14	0.07	0.17	0.82	0.99	0.50	0.95	0.10	0.85	
RRT	0.11	0.05	0.25	0.56	0.99	0.22	0.97	0.03	0.95	
EST	0.11	0.05	0.25	0.57	0.99	0.25	0.97	0.04	0.94	
PRMu	0.50	0.02	0.24	0.39	0.99	0.04	0.97	0.00	0.94	$\rho^*_{\text{L}_2}$
PRMb	0.46	0.02	0.24	0.38	0.99	0.03	0.97	0.00	0.94	
PRMg	0.49	0.02	0.24	0.37	0.99	0.04	0.97	0.00	0.94	
PRMo	0.46	0.03	0.22	0.45	0.99	0.07	0.97	0.01	0.93	
RRT	0.29	0.01	0.29	0.20	0.99	0.00	0.98	0.00	0.96	
EST	0.44	0.01	0.28	0.14	0.99	0.00	0.98	0.00	0.95	

In Fig. 6(a), we compare the computation time of DPES relative to Linear. As expected, the computation time of DPES relative to Linear increases as the number of pivots increases. However, even for $m = 50$, DPES is still several times faster than Linear. As the dimension increases, the improvement in the computation time of DPES relative to Linear increases as well.

In Fig. 6(b), we focus on the rde error of DPES. As before, we observe that the rde error of DPES increases with the dimension but still remains small even when $d = 60$ and $m = 10$. Furthermore, as the number of pivots increases, the rde error of DPES quickly approaches zero.

In Fig. 6(c), we focus on the rfd error for $\epsilon = 0.00$. We note that the $\text{rfd}_{0.00}$ error of DPES increases with the dimension but decreases rapidly as the number of pivots increases. In fact, when $d = 54$ and $m = 50$, the $\text{rfd}_{0.00}$ error is less than 0.20. This indicates that DPES includes at least 80% of the exact k nearest neighbors in the computed approximate k nearest neighbors.

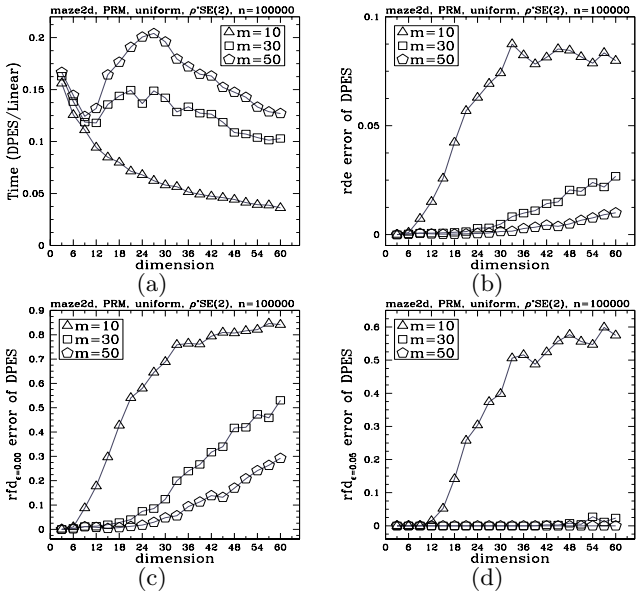


Fig. 6. Improved kANN results for PRM and the $\rho^*_{\mathrm{SE}(2)}$ distance metric.

In Fig. 6(d), we focus on the rfd error for $\epsilon = 0.05$. We again note that the $\mathrm{rfd}_{0.05}$ error of DPES increases with the dimension but decreases rapidly and approaches 0 as the number of pivots increases. In fact, when $m = 50$, the $\mathrm{rfd}_{0.05}$ error of DPES is 0.00, i.e., all the approximate nearest neighbors are no more than 1.05 times farther away from the k-th nearest neighbor.

5 Discussion

In this work, we quantitatively analyzed exact and approximate nearest-neighbors algorithms for points obtained from sampling-based motion planning methods in high-dimensional problems.

Our analysis indicates that the computational efficiency of exact nearest-neighbors algorithms deteriorates rapidly as the dimension increases. After a critical dimension, which in our experiments varied between 15 and 30, exact nearest-neighbors algorithms evaluate almost as many distances as the brute-force Linear method and are thus impractical when a considerably large number of samples is necessary for solving motion planning problems.

Motivated by the impracticality of exact nearest-neighbors algorithms on high-dimensional motion planning problems, we developed a simple approximate nearest-neighbors algorithm, DPES, which achieves high computational efficiency and only a negligible loss in accuracy. The computational efficiency of DPES relative to Linear improves rapidly as the dimension increases. This

is due to (i) the distance-based projection of high-dimensional data sets onto low-dimensional Euclidean spaces reduces to a certain extent the computational dependencies on the dimension (ii) the number of distance computations by DPES relative to Linear is only a small fraction; and (iii) DPES uses L_2 which is computationally more efficient than $\rho^*_{\mathrm{SE}(2)}$, $\rho^*_{\mathrm{SE}(2)}$, $\rho^*_{w\mathrm{SE}(3)}$, $\rho^*_{w\mathrm{SE}(3)}$, or $\rho^*_{\mathrm{L}_2}$. Our analysis also shows that DPES is highly accurate. In the computed queries, DPES includes many of the exact nearest neighbors and the rest are close to the exact nearest neighbors.

Since in motion planning the purpose of nearest neighbors is to provide candidates which the local planner can connect to the query point, using approximate nearest neighbors that are similar to exact nearest neighbors may indeed be sufficient. This paper shows that in high-dimensional motion planning problems nearest neighbors can be computed more efficiently by using highly accurate approximate nearest-neighbors algorithms, such as DPES, instead of exact nearest-neighbors algorithms.

Acknowledgement. Work on this paper by the authors has been supported in part by NSF 0205671, NSF 0308237, ATP 003604-0010-2003, NIH GM078988, and a Sloan Fellowship to LK. The Rice Terascale Cluster, PBC Cluster, and Rice Cray XD1 Cluster ADA used in this work are supported by EIA 0216467, CNS 0454333, CNS 0421109, AMD, Cray, Intel, and Hewlett Packard.

References

1. N. M. Amato, B. Bayazit, L. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3d workspaces. In P. Agarwal, L. E. Kavraki, and M. Mason, editors, *Robotics: The Algorithmic Perspective*, pages 156–168. AK Peters, 1998.
2. S. Arya, D. M. Mount, and S. Nathan. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.
3. A. Atramentov and S. M. LaValle. Efficient nearest neighbor searching for motion planning. In *IEEE International Conference on Robotics and Automation*, pages 632–637, Washington, DC, 2002.
4. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? *Lecture Notes in Computer Science*, 1540:217–235, 1999.
5. V. Boor, M. H. Overmars, and A. F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *IEEE International Conference on Robotics and Automation*, pages 1018–1023, Detroit, MI, 1999.
6. A. Borodin, R. Ostrovsky, and Y. Rabani. Lower bounds for high dimensional nearest neighbor search and related problems. In *ACM Symposium on Theory of Computing*, pages 312–321, Atlanta, GA, 1999.
7. S. Brin. Near neighbor search in large metric spaces. In *International Conference on Very Large Data Bases*, pages 574–584, San Francisco, California, 1995.
8. F. Bullo and R. M. Murray. Proportional derivative (PD) control on the Euclidean group. In *European Control Conference*, pages 1091–1097, Rome, Italy, 1995.

9. H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, 2005.
10. B. Cui, H. T. Shen, J. Shen, and K.-L. Tan. Exploring bit-difference for approximate knn search in high-dimensional databases. In *Australasian Database Conference*, pages 165–174, Newcastle, Australia, 2005.
11. V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.
12. A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *International Conference on Very Large Data Bases*, pages 506–515, Cairo, Egypt, 2000.
13. D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *IEEE International Conference on Robotics and Automation*, pages 4420–442, Taipei, Taiwan, 2003.
14. D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255, 2002.
15. P. Indyk. Nearest neighbors in high-dimensional spaces. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 877–892. CRC Press, 2004.
16. P. Indyk and J. Matoušek. Low-distortion embeddings of finite metric spaces. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 177–196. CRC Press, 2004.
17. L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
18. F. Korn, B.-U. Pagel, and C. Faloutsos. On the 'dimensionality curse' and the 'self-similarity blessing'. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):96–111, 2001.
19. E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal of Computing*, 30(2):457–474, 2000.
20. S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.
21. T. Liu, A. W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 825–832. MIT Press, Cambridge, MA, 2005.
22. E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki. Sampling-based roadmap of trees for parallel motion planning. *IEEE Transactions on Robotics*, 21(4):597–608, 2005.
23. R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *International Conference on Very Large Data Bases*, pages 194–205, New York, NY, 1998.