Reporting Neighbors in High-Dimensional Euclidean Space *

Dror Aiger† Haim Kaplan‡ Micha Sharir§

Abstract

We consider the following problem, which arises in many database and web-based applications: Given a set P of n points in a high-dimensional space $\mathbb{R}^d$ and a distance r, we want to report all pairs of points of P at Euclidean distance at most r. We present two randomized algorithms, one based on randomly shifted grids, and the other on randomly shifted and rotated grids. The running time of both algorithms is of the form $C(d)(n+k)\log n$, where k is the output size and $C(d)$ is a constant that depends on the dimension d. The $\log n$ factor is needed to guarantee, with high probability, that all neighbor pairs are reported, and can be dropped if it suffices to report, in expectation, an arbitrarily large fraction of the pairs. When only translations are used, $C(d)$ is of the form $(a\sqrt{d})^d$, for some (small) absolute constant $a \approx 0.484$; this bound is worst-case tight, up to an exponential factor of about 2^d. When both rotations and translations are used, $C(d)$ can be improved to roughly 6.74^d, getting rid of the super-exponential factor $\sqrt{d}^d$. When the input set (lies in a subset of d-space that) has low *doubling dimension* δ, the performance of the first algorithm improves to $C(d,\delta)(n+k)\log n$ (or to $C(d,\delta)(n+k)$), where $C(d,\delta) = O((ed/\delta)^\delta)$, for $\delta \le \sqrt{d}$. Otherwise, $C(d,\delta) = O\left(e^{\sqrt{d}}\sqrt{d}^\delta\right)$.

We also present experimental results on several large datasets, demonstrating that our algorithms run significantly faster than all the leading existing algorithms for reporting neighbors.

1 Introduction

Problems involving distances between points in high-dimensional Euclidean space are of major importance in many areas, including pattern recognition [28], searching in multimedia data [28], vector compression [22], computational statistics [18], and data mining [39]. Most of these applications involve huge data sets (tens of billions of images or documents) and the dimensionality of the points (that is, number of real attributes representing or encoding

*Work by Haim Kaplan and Micha Sharir has been supported by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11). Work by Haim Kaplan has also been supported by grant 822/10 from the Israel Science Fund and by Grant 2006/204 from the U.S.-Israel Binational Science Foundation. Work by Micha Sharir has also been supported by Grant 338/09 from the Israel Science Fund, and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University. A preliminary version of the paper has appeared in *Proc. 24th ACM-SIAM Sympos. Discrete Algorithm*, 2013.

†Google Inc.; email: aigerd@google.com

‡School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel; email: haimk@tau.ac.il

§School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel, and Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA; email: michas@tau.ac.il

a point) is usually large as well (typically in the hundreds). It is therefore important to have algorithms that scale well with the input size and with the dimension.

The class of proximity problems of this kind includes nearest neighbor searching, near neighbor reporting, closest pair, diameter, minimum spanning tree and a variety of clustering problems; see [23]. In this paper we focus on the *all near-neighbors reporting problem*. That is, given an input set P of n points in $\mathbb{R}^d$, we wish to report all pairs of points of P at (Euclidean) distance at most some given value r.

We continue with a review of some (rather small) subset of the vast work on such proximity problems which is most relevant to our work.

Background. For arbitrary fixed dimension d and for the L_∞-distance, the grid technique used by Lenhof and Smid [32] can be modified to solve the all near-neighbors problem in optimal $O(n+k)$ time, where k is the output size. The algorithm simply inspects all pairs of points that lie in adjacent grid cells (or in the same cell) of some fixed uniform grid of cell size r. The efficiency of the algorithm follows from a packing lemma, established in [32] and similar to an earlier variant by Salowe [36]. It asserts that, for any $r > 0$, the number of pairs at L_∞-distance at most $2r$ is at most a constant multiple of n plus the number of pairs at L_∞-distance at most r; see also Chan [13] for a simplified version.

A major drawback of the algorithms in [13, 32] is that their analysis only caters to L_∞-distances. Although the algorithms also work for any L_p-distance, no analysis of the corresponding variant of the packing lemma is provided; consequently, no sharp bounds on the performance of the algorithms have been established. Our paper considers the case of Euclidean distance and provides a rather intricate analysis of this case, which replaces the simpler L_∞-based analysis of [13, 32].

The use of the L_2-metric as a proximity measure is common, in particular in computer vision applications, for identifying similar images based on distances between feature vectors extracted from the images [9]. In high dimensional spaces the sparsity of the data makes the estimation of similarity by distance problematic. Specifically, consider the L_p norm for some $p \geq 1$. It has been shown [11, 27] (see also [1]) that for n points drawn independently at random from any distribution in d dimensions[1] (think of n as fixed and of d as going to infinity) the difference between the distances of the farthest point and of the nearest point from the origin increases at a rate of $d^{\frac{1}{p}-\frac{1}{2}}$. The expression $d^{\frac{1}{p}-\frac{1}{2}}$ goes to 0 for $p > 2$, equals 1 for $p = 2$, and goes to ∞ for $1 \leq p < 2$. This suggests L_1 and L_2 as the best simple norms for similarity estimation in high dimensional spaces, with an advantage for L_1. In this paper we consider only the L_2-metric. Extending our ideas to deal with the L_1-metric is an interesting open problem.

We note that the constants of proportionality in the bounds in [13, 32] are exponential in d, which might make the algorithms rather inefficient when d is large (as is generally the case in practice). In fact, this "curse of dimensionality" is omnipresent in most of the algorithms for proximity problems in higher dimensions, and the bounds in our paper are no exception, although we make every effort to reduce the dependence on d.

The results mentioned so far are for "one-shot" problems, where we are given an input set and wish to compute certain proximity attributes thereof. A considerably larger effort has been invested in preprocessing-and-query problems, where the goal is to construct some

[1] Technically it is a different arbitrary distribution for each dimension d.

data structure on the input point set which supports proximity queries of a certain kind. The most ubiquitous of this kind of problems is *nearest neighbor searching*, where we are given a set P of n points in a high-dimensional space $\mathbb{R}^d$, and wish to construct a data structure that, given a query point q, finds the point in P closest to q. Extensive research on this problem has led to a variety of interesting solutions, both exact and approximate. Here too the dependence on d of the performance of the resulting algorithms is at least exponential.

Many of the known exact and approximate nearest neighbor searching data structures can be modified to report all (or most) points of P that are within a certain given distance r from a query point q. Consequently, one can use such a structure to solve the all near-neighbors reporting problem, by first building a nearest-neighbor data structure for the points of P, and then querying this structure with each $q \in P$, thereby obtaining the points in P at distance at most r from q. The running time of such a solution is the time it takes to build the nearest neighbor data structure plus the time for n queries. The space required is the total space used by the data structure plus the output size.

We give a brief review of the state-of-the-art approximate nearest neighbor data structures. For more information and related references see Har-Peled's recent book [24]. These approximate nearest neighbor data structures return, for any query point q, a point p whose distance from q is at most $(1+\varepsilon)$ times the distance between q and its nearest neighbor.

A data structure based on Box-Decomposition Trees, partitioning space into axis-aligned boxes, was given by Arya et al. [7]. This structure takes $O(n)$ space, can be constructed in $O(n \log n)$ time, and answers a query in $O(\frac{1}{\varepsilon^d} \log n)$ time. Several improvements to this data structure have been given [10, 14, 15, 19]; in particular, the query time was reduced to $O(\frac{1}{\varepsilon^d} + \log n)$.[2] A nice feature of this data structure is that ε does not have to be specified at the preprocessing stage but only at query time (and can vary with the query). In subsequent work it was shown that the query time can be reduced further, at the expense of larger storage. One of the main tools used to obtain these results is the approximate Voronoi diagram of Har-Peled, with a quadtree-like search structure on top of it [25]. To date, many trade-offs between query time and space have been achieved, and in all of the more efficient ones the product of the term depending on ε in the storage and the square of the term depending on ε in the query time is roughly $\frac{1}{\varepsilon^d}$ [6].

To overcome the exponential dependence on d of the performance of all these data structures, Indyk and Motwani introduced a different technique called *Locally Sensitive Hashing (LSH)* [30]. The first component in this method is a reduction from the problem of approximate nearest neighbor search to the problem of finding a neighbor at distance $\leq (1+\varepsilon)r$ if there exists a neighbor at distance r, for some prespecified $r > 0$. Then the latter problem is solved using a family of hash functions that tend to map close points to the same bin (an LSH family in short). If we neglect polylogarithmic factors (but this time do not hide factors exponential in the dimension) then the solution of Indyk and Motwani answers a query in $O(n^{\frac{1}{1+\varepsilon}})$ time and takes $O(n^{1+\frac{1}{1+\varepsilon}})$ preprocessing time and storage. This was later improved, using more complex hash functions, to $O(n^{\frac{1}{(1+\varepsilon)^2}})$ query time and $O(n^{1+\frac{1}{(1+\varepsilon)^2}})$ preprocessing and storage [3, 17]. For a survey on the LSH technique see [5].

Our results. In this paper we present two simple randomized algorithms for reporting

[2] We still hide factors exponential in d which are independent of ϵ.

all near neighbors in a set P of n points in a high-dimensional Euclidean space $\mathbb{R}^d$. That is, given a threshold distance r, we want to report all (or most) pairs of input points at Euclidean distance at most r.

Our algorithms are straightforward to implement, and indeed this work has started with the implementation of these algorithms, and with the realization that they work very well in practice. We have then aimed at a rigorous analysis of the worst-case performance of the algorithms, and its dependence on n, on the output size k, and, equally importantly, on the dimension d. This analysis has turned out to be rather intricate; it requires the exploitation of several interesting properties of balls, Minkowski sums, and other structures in high-dimensional spaces.

Our first (and simpler) algorithm lays down a randomly shifted uniform grid of certain cell size c (that is, each cell is an axis-parallel cube with side-length c), collects the points of P in each grid cell, and inspects *all* pairs of points within each cell (this latter feature is similar to the approach of [32]), reporting those that are at Euclidean distance at most r from each other. This is repeated a suitable number of times to guarantee that, with high probability, all, or most pairs of points at Euclidean distance at most r will be detected, that is, will both fall into the same grid cell. See later for full details concerning the issue of reporting all vs. most pairs.

There are two factors that determine the efficiency of the algorithm. The first factor is the number of repetitions of the grid layout procedure just described. This number depends (reciprocally) on the probability that, for a specific pair a, b of points at Euclidean distance at most r, both a and b will fall into the same cell of a randomly shifted grid of cell size c.

For a very naive lower bound on this probability, put $x = a - b$, so $\|x\| \leq r$. The probability that a and b fall into different cells is at most $(\sum_{i=1}^{d} x_i)/c$, which, by the Cauchy–Schwarz inequality, is at most $\|x\|\sqrt{d}/c \leq r\sqrt{d}/c$. This means that two points a and b at distance at most 1 end up in the same grid cell with probability at least $1 - r\sqrt{d}/c$. Notice however that this bound is meaningful only for $c \geq r\sqrt{d}$. Our experiments however showed that one can get very good results with much smaller cells, of size even close to r. This has motivated us to prove a better lower bound on this probability which is meaningful also for small values of c. The analysis of this probability estimation is presented in Section 2.

We note that a related approach is to cover the point set P with random balls, such that each point is assigned to one particular ball. As shown in [16, Section 3] it is rather easy to construct such a cover with balls of diameter c such that the probability that two points a and b are in different balls is at most $\|x\|\sqrt{d}/c$ where $x = a - b$ as before. This is similar to the randomly shifted grid technique but harder to implement as it requires to draw a random center in a union of balls. We also recall that the hash functions underlying the most efficient variants of LSH map points into balls such that close points tend to map to the same ball. Whether such a scheme based on balls can be made practical and if so how well does it perform in practice are open questions.

The other factor determining the performance of such partitioning schemes is the ratio between the number of pairs that the algorithm inspects (which, for a fixed grid, is $\sum_\tau \binom{|P\cap\tau|}{2}$, where the sum is over all cells τ of the grid), and the output size k, namely, the number of pairs at Euclidean distance at most r. The intuition is that if a cell τ contains many points, and if the cell size c is relatively small, many pairs of these points should be close to each other in the Euclidean metric. While this intuition is correct, sharp calibration

of the number of such pairs (especially its dependence on the dimension d) turns out to be rather involved. This analysis is presented in Section 3.

There is a trade-off in the choice of the cell size c. On one hand we would like to make it large, to increase the probability of capturing near neighbors in the same grid cell (and thereby decrease the number of repetitions of the basic grid layout procedure), but on the other hand a large value of c will increase the ratio between the number of pairs inspected by the algorithm and the number of pairs at (Euclidean) distance at most r (the output size), which will increase the running time of each step. One interesting finding of our analysis is that in most cases the (theoretically) best choice for c is a value very close to r.

Specifically, we show (see Corollary 3.4) that, for a given set P of n points in $\mathbb{R}^d$ and for a threshold distance r, the running time of the algorithm is $C(d)(n+k)\log n$ (for reporting, with high probability, all pairs at distance at most r), or $C(d)(n+k)$ (for reporting, in expectation, an arbitrarily large fraction of these pairs), where $C(d) = O\left(e^{\sqrt{d}}\frac{(\sqrt{d})^d e^{\frac{3}{2}(\pi/2)^{1/3}d^{2/3}}}{(\sqrt{\pi e/2})^d}\right)$. Here the "big-O" notation only hides factors depending polynomially on d, k is the output size, and the success probability or the expected fraction of reported pairs can be made arbitrarily close to 1 by increasing $C(d)$ by an appropriate absolute constant factor.

A major drawback of the first algorithm is that the cell size c must clearly be greater than r (in particular, if we want to report all pairs). In this case the ratio between the number of pairs of points in a cell and the number of these pairs at Euclidean distance at most r is large in the worst case, and its dependence on d is super-exponential, about $\sqrt{d}^d$. Informally, this is a consequence of the fact that the volume of the d-dimensional Euclidean ball is much smaller than the volume of a cube of the same size (the ratio between the volumes involves the super-exponential factor just mentioned).

To overcome this super-exponential dependence on d, we turn to our second algorithm, which is similar to the first one, except that at each step it first applies a random rotation of the coordinate frame, and then lays down a randomly shifted grid of an appropriate cell size c. As might seem surprising at first glance, here we can choose c to be much smaller than r; specifically, we choose c to be proportional to $r/\sqrt{d}$. An appropriate such choice of c still ensures non-negligible positive probability of capturing a pair of points at Euclidean distance at most r in the same grid cell. This probability is much smaller than the corresponding probability in the first algorithm (where c was chosen to be much larger), and consequently the number of repetitions of the basic step is much larger. However, because the cell size is now so small, we more than compensate for this degradation by drastically reducing the ratio between the number of inspected pairs and the output size. Overall, the dependence on d of the performance of the algorithm now becomes only exponential in d (where the best base that this method seems to yield is about 6.74). Specifically, the running time of the algorithm is $O(6.74^d(n+k)\log n)$ (for reporting, with high probability, all pairs at distance at most r), or $O(6.74^d(n+k))$ (for reporting, in expectation, an arbitrarily large fraction of these pairs), where the constant of proportionality depends polynomially on d, k is the output size, and the success probability or the expected fraction of reported pairs can be made arbitrarily close to 1 by increasing the bound by an appropriate absolute constant factor. This algorithm and its analysis are presented in Section 5.

In spite of all this progress, an exponential dependence on d is far from satisfactory, especially when d is really large. In retrospect, though, it appears that a major factor

affecting the observed efficiency of the algorithms in practice is the fact that the input point sets tend to be of low *doubling dimension* $\delta \ll d$ with respect to the Euclidean metric [8]. That is, for every Euclidean ball B, the set $B \cap P$ can be covered by at most 2^δ balls of half the radius. As it turns out, our first algorithm is suitable for handling such input sets, and its analysis can be modified, to yield a much improved dependence of its performance on d and δ. Specifically, the dependence is now sub-exponential in d and exponential only in δ; see Corollary 4.2 for the precise bound. This provides theoretical substantiation for the efficiency of the algorithm in practice. Low doubling dimension occurs naturally in practical applications and was exploited before for fast algorithms for approximate nearest neighbor searching [26, 31].

As already mentioned, our work has started with the implementation of our algorithms and with testing them on large data sets, observing that they work very well in practice. After providing the theoretical analysis that supports these findings, we end the paper by presenting some experimental results that manifest the efficiency of our implementations. In these results we compare our first algorithm (based on randomly shifted grids) with several of the leading existing software packages for reporting neighbors. As the results show, our algorithm runs significantly faster than any of its competitors. These results are presented in Section 6.

The techniques used by our algorithms are not new. Randomly shifted grids and random rotations have been used in the past in algorithms for various proximity problems and metric embeddings. Examples can be found in Har-Peled's book [24] and in the survey on metric embeddings by Indyk and Matoušek [29]. In particular, randomly shifted grids have been suggested and analyzed as an LSH family for the L_1-distance in [4]. A similar LSH family for L_2-distance based on projecting into quantized random lines was analyzed in [17]. Nevertheless, our precise and intricate theoretical and experimental analysis of algorithms based on randomly shifted grids for reporting all near neighbors appears to be new.

2 Randomly shifted grids

In this section we present and analyze our first algorithm, which is based on randomly shifted grids. To simplify the notation, and without loss of generality, we consider the problem of reporting all (or most) pairs of points at Euclidean distance at most 1 in a set P of n points in $\mathbb{R}^d$.

The simple algorithm consists of the following steps.

(a) Fix a parameter $c > 1$ and fairly close to 1; the exact choice of c will be discussed in detail below.

(b) Place a randomly shifted uniform axis-parallel grid $\mathcal{G}$ of cell size c in $\mathbb{R}^d$, and compute, for each cell τ of $\mathcal{G}$, the subset $P_\tau = P \cap \tau$.

(c) For each cell τ with $|P_\tau| \geq 2$, go over all pairs of points of P_τ and report those pairs at Euclidean distance at most 1.

(d) Repeat steps (b) and (c) M times, for an appropriate parameter M (that depends on c, d, and optionally on n; see below for its precise choice). In doing so we filter out pairs that have already been reported.

The algorithm is clearly very simple to implement. Step (b) can be implemented in $O(n)$ time if we assume availability of a constant-cost implementation of the floor function and of hashing. Similarly, each filtering operation in step (d) can be implemented in $O(1)$ time using constant-cost hashing. All the other steps are trivial to implement.

As the analysis below will show, for appropriate choices of c and M, the algorithm has the following properties: (i) With high probability, every pair at distance at most 1 will be reported; alternatively, with an appropriate choice of a smaller value of M, an arbitrarily large fraction of these pairs will be reported in expectation. (ii) The algorithm is output-sensitive, in the sense that the number of pairs that it examines is at most $C(d)(n+k)\log n$ (for reporting all pairs), or $C(d)(n+k)$ (for reporting an expected arbitrarily large fraction of the pairs), where $C(d)$ is a constant that depends on d, and k is the output size (the number of pairs at Euclidean distance at most 1).

The constant $C(d)$ that we obtain grows super-exponentially in d; specifically, it is of the form $(a\sqrt{d})^d$, for some (small) absolute constant a. We show that an approach like ours must incur a constant that in the worst case is at least roughly $\left(\sqrt{d}/\sqrt{2\pi e}\right) \approx (0.242\sqrt{d})^d$. so the challenge is to make the base a in our bound as small as possible. The best parameter that we can obtain is $a \approx \sqrt{\frac{2}{\pi e}} \approx 0.484$, when d is sufficiently large, using some non-trivial analysis. This gets us reasonably close to the worst-case lower bound (within a factor of roughly 2^d). Later, in Section 5, we will be able to reduce the constant to roughly 6.74^d, thereby getting rid of the super-exponential factor $\sqrt{d}^d$. This is achieved by using in addition random rotations of the coordinate frame.

Capturing a unit vector in a grid cell. One of the main steps in the analysis is to establish property (i) of the algorithm, namely that every pair of points at Euclidean distance at most 1 will be captured in the same cell of a randomly shifted grid with at least some probability $\zeta > 0$. Then, repeating the grid construction $M = \frac{b}{\zeta}\log n$ times, the probability that a fixed pair will not be captured will be at most $(1-\zeta)^M \approx e^{-b\log n} = 1/n^b$, so, by the probability union bound, the probability that all appropriate pairs will be captured will be at least $1 - \binom{n}{2}/n^b > 1 - 1/(2n^{b-2})$, which we can make arbitrarily small by choosing $b > 2$ sufficiently large. Alternatively, repeating the process only $M' = \frac{b}{\zeta}$ times, the success probability of capturing any fixed pair is $\approx 1 - e^{-b}$. Hence, choosing b arbitrarily large, we can ensure that an arbitrarily large fraction of the desired pairs will be reported. As noted, this is sufficient in many applications.

We now proceed to estimate ζ. As is intuitively clear, the worst lower bound for ζ occurs when the distance between the points in the pair is 1; this will be briefly justified more rigorously at the end of the analysis.

So let $c > 1$ be fixed, and let uv be a unit vector in $\mathbb{R}^d$. Let $\mathcal{G}$ be a randomly shifted uniform axis-parallel grid in $\mathbb{R}^d$ of cell size c; that is, we draw a point o uniformly at random from $[0,c]^d$ and shift the grid so that o is one of its vertices. Our goal is to derive a lower bound for the probability ζ that both u and v lie in the same grid cell.

Write $\vec{uv} = (x_1, x_2, \ldots, x_d)$, with $x_1^2 + \cdots + x_d^2 = 1$. For each $i = 1, \ldots, d$, the probability that the x_i-projections of u and v both lie in the same edge of a grid cell is $(c - |x_i|)/c$, so

the probability of capturing both u and v in the same grid cell is

$$\frac{(c-|x_1|)(c-|x_2|)\cdots(c-|x_d|)}{c^d}.$$

In other words, the problem can be stated as follows. Let σ denote the unit $(d-1)$-sphere in $\mathbb{R}^d$, given by $x_1^2+x_2^2+\cdots+x_d^2=1$, and let $c>1$ be a parameter. We want to find the minimum value of the function

$$F_c(\mathbf{x})=\left(1-\frac{|x_1|}{c}\right)\left(1-\frac{|x_2|}{c}\right)\cdots\left(1-\frac{|x_d|}{c}\right)$$

for $\mathbf{x}=(x_1,\ldots,x_d)\in\sigma$.

To simplify the analysis, let σ^+ denote the portion of σ in the positive orthant. It suffices to find the minimum of F_c on σ^+, in which case we can rewrite $F_c(\mathbf{x})$ as

$$F_c(\mathbf{x})=\left(1-\frac{x_1}{c}\right)\left(1-\frac{x_2}{c}\right)\cdots\left(1-\frac{x_d}{c}\right).$$

The case $c>\sqrt{2}$. Assume first that $c>\sqrt{2}$. In this case, using Lagrange multipliers, we claim that the unique extremum of F_c in the *relative interior* of σ^+ (namely, at points whose coordinates are all positive) is attained at

$$x_1=x_2=\cdots=x_d=\frac{1}{\sqrt{d}},$$

and the value of F_c at that point is

$$F_c^{(d)}=\left(1-\frac{1}{c\sqrt{d}}\right)^d.$$

To see this, note that the Lagrange multipliers technique yields the equations

$$2\lambda x_i=-\frac{1}{c-x_i}F_c(\mathbf{x}),\quad \text{for } i=1,\ldots,d, \tag{1}$$

or, equivalently, all the products $x_i(c-x_i)$ are equal. Thus $x_i(c-x_i)=x_j(c-x_j)$ for any $i\neq j$, which is equivalent to $(x_i-x_j)(x_i+x_j-c)=0$. However, since $x_i^2+x_j^2\leq 1$ we have $x_i+x_j\leq\sqrt{2}<c$, so we must have $x_i=x_j$, implying that all coordinates at the extremum are equal.

The other local extrema of F_c on σ^+ must occur on the boundary of σ^+, at points where some coordinates are 0. By symmetry, it suffices to examine $d-1$ additional subproblems, where in the j-th subproblem, $j=1,\ldots,d-1$, we seek the extrema of F_c on

$$\sigma_j^+=\sigma^+\cap\{\mathbf{x}\mid x_{j+1}=x_{j+2}=\cdots=x_d=0\}.$$

Arguing as in the case of the whole σ^+, the unique extremum of F_c in the relative interior of σ_j^+ is attained at

$$x_1=x_2=\cdots=x_j=\frac{1}{\sqrt{j}},\quad x_{j+1}=x_{j+2}=\cdots=x_d=0,$$

and the value of F_c at that point is

$$F_c^{(j)} = \left(1 - \frac{1}{c\sqrt{j}}\right)^j.$$

In other words, the minimum of F_c over σ is the minimum of the sequence

$$s_j(c) = \left(1 - \frac{1}{c\sqrt{j}}\right)^j, \qquad j = 1, \ldots, d.$$

To simplify matters further, consider instead the minimum of the sequence

$$t_j = \ln s_j(c) = j \ln \left(1 - \frac{1}{c\sqrt{j}}\right), \qquad j = 1, \ldots, d,$$

or rather its continuous version

$$f(x) = x \ln \left(1 - \frac{1}{c\sqrt{x}}\right).$$

We have

$$f'(x) = \ln \left(1 - \frac{1}{c\sqrt{x}}\right) + \frac{x}{1 - \frac{1}{c\sqrt{x}}} \cdot \frac{1}{2cx\sqrt{x}} =$$

$$\ln \left(1 - \frac{1}{c\sqrt{x}}\right) + \frac{1}{2(c\sqrt{x} - 1)}.$$

Put $y = c\sqrt{x}$. Since we are interested in the range $x \geq 1$, we have $y > \sqrt{2}$. However, to accommodate also the case $1 < c \leq \sqrt{2}$, discussed next, we consider the extended range $y > 1$. We need to solve

$$2 \ln \left(1 - \frac{1}{y}\right) + \frac{1}{y - 1} = 0.$$

Note that this equation is independent of c. Using the *WolframAlpha* software and some numerical calculations, we find that there is a unique zero at $y_0 \approx 1.39795$, where f attains its *maximum* (f' is positive for $y < y_0$ and negative for $y > y_0$). That means that the maximum of f is attained at $x_0 = y_0^2/c^2 < 0.977 < 1$ (for $c > \sqrt{2}$).

In particular, f decreases for $x > x_0$, and therefore the minimum value of $s_j(c)$, for $j \geq 1 > x_0$, is attained at $j = d$. That is, we have shown that

$$F_c(\mathbf{x}) \geq \left(1 - \frac{1}{c\sqrt{d}}\right)^d \tag{2}$$

for all $\mathbf{x} \in \sigma$, provided that $c > \sqrt{2}$.

The case $1 < c \leq \sqrt{2}$. We first note that the analysis given for the preceding case continues to apply here too, in the sense that all the values $s_j(c)$, for $1 \leq j \leq d$, continue to be local extrema of F_c (more precisely, $s_j(c)$ continues to be a local extremum in the relative interior of a $(j-1)$-dimensional sub-sphere of σ). The maximum of the corresponding continuous function $f(x)$ is still attained at $x = y_0^2/c^2$ which is smaller than 2 for any $c > 1$,

implying that the suffix $s_2(c), s_3(c), \ldots, s_d(c)$ is decreasing, so its minimum is still $s_d(c)$, as given in (2). Here however we have a new contender for the minimum, namely

$$s_1(c) = 1 - \frac{1}{c},$$

which, when c is very close to 1, can indeed be smaller than $s_d(c)$. To avoid this case, we will require that

$$1 - \frac{1}{c} \geq e^{-\frac{\sqrt{d}}{c}} \geq \left(1 - \frac{1}{c\sqrt{d}}\right)^d .$$

The right inequality always holds, being an instance of the inequality $e^{-x} \geq 1 - x$ for $x \geq 0$. The left inequality will hold, for $c \leq \sqrt{2}$, if we choose

$$c \geq \frac{1}{1 - e^{-\sqrt{d/2}}},$$

which in turn holds if

$$c \geq 1 + 2e^{-\sqrt{d/2}}. \tag{3}$$

The latter implication follows since $1 + 2x \geq \frac{1}{1-x}$ for $x \leq 1/2$, and $e^{-\sqrt{d/2}} \leq 1/2$ for $d \geq 2$.

In the remainder of this section we will only consider the case where c satisfies (3).

Equation (3) guarantees that $s_1(c) \geq s_d(c)$ but this still does not make $s_d(c)$ the minimum of $F_c(x)$ over σ^+ since for $1 < c \leq \sqrt{2}$, F_c has some additional local extrema. Specifically, using the Lagrange multipliers technique, and the implied analysis following Equation (1), we conclude that, at an extremum $\mathbf{x}$ of F_c on σ^+, some coordinates x_i may be zero, and any pair x_i, x_j of nonzero coordinates must satisfy either $x_i = x_j$ or $x_i + x_j = c$. This implies right away that (on σ^+) the nonzero coordinates of $\mathbf{x}$ have only two distinct values, which we denote as t and $c - t$, and assume that $t \leq c - t$, so $c - 1 \leq t \leq \frac{c}{2}$.

Suppose that we have a coordinates equal to t and b coordinates equal to $c - t$, with $a + b \leq d$. We then must have

$$at^2 + b(c - t)^2 = 1. \tag{4}$$

Since $t \leq c - t$ we get that $c - t > 1/2$, which is easily seen to imply that $b \leq 3$.

Assume first that $t \geq 1 - \frac{1}{\sqrt{2}}$. Then

$$a = \frac{1 - b(c-t)^2}{t^2} \leq \frac{1 - b/4}{t^2} \leq \frac{3/4}{3/2 - \sqrt{2}} = 3(3/2 + \sqrt{2}) < 9,$$

so $a \leq 8$. In this case we have

$$F_c(\mathbf{x}) = \left(\frac{t}{c}\right)^b \left(1 - \frac{t}{c}\right)^a \geq \left(\frac{t}{c}\right)^3 \left(1 - \frac{t}{c}\right)^8 .$$

Since $t/c \geq t/\sqrt{2} \geq (\sqrt{2} - 1)/2$ and $1 - t/c = (c - t)/c \geq (c - t)/\sqrt{2} \geq 1/(2\sqrt{2})$, $F_c(\mathbf{x})$ is at least some absolute constant

$$\mu \geq \left(\frac{\sqrt{2} - 1}{2}\right)^3 \left(\frac{1}{2\sqrt{2}}\right)^8 .$$

Since $s_d(c)$ decreases to 0 as d grows (as argued above, it is upper bounded by $e^{-\sqrt{d}/c} \leq e^{-\sqrt{d/2}}$), $F_c(\mathbf{x})$, for any new local extremum $\mathbf{x}$, will be larger than $s_d(c)$ when d is at least some sufficiently large constant d_0 (independent of c).

If $t \leq 1 - \frac{1}{\sqrt{2}}$ then $1 - t \geq \frac{1}{\sqrt{2}}$. In this case $(c-t)^2 > 1/2$ and then (4) implies that $b = 1$, and also that

$$a = \frac{1-(c-t)^2}{t^2} < \frac{2t - t^2}{t^2} = \frac{2}{t} - 1 < \frac{2}{t}.$$

Substituting a and b into the expression for $F_c(\mathbf{x})$, we get that

$$F_c(\mathbf{x}) \geq \frac{t}{c}\left(1 - \frac{t}{c}\right)^{2/t} \geq \frac{t}{c}e^{-4/c} \ .$$

Here we use the inequality $1 - x \geq e^{-2x}$, which holds for $x = \frac{t}{c} < 1/2$. If

$$te^{-4/c} \geq ce^{-\sqrt{d}/c} \tag{5}$$

then $F_c(\mathbf{x}) = \frac{t}{c}e^{-4/c}$ will be larger than $s_d(c)$. Inequality (5) is equivalent to

$$t \geq \frac{c}{e^{\frac{\sqrt{d}-4}{c}}} \ .$$

If $c - 1 \geq \frac{c}{e^{\frac{\sqrt{d}-4}{c}}}$ we are done since we always have $t \geq c - 1$, so the new extremal values of F_c will all be greater than $s_d(c)$. (Note that for this inequality to hold, d must be at least 17.) So assume that $c - 1 < \frac{c}{e^{\frac{\sqrt{d}-4}{c}}}$, or

$$c < 1 + \frac{c}{e^{\frac{\sqrt{d}-4}{c}}} \leq 1 + \frac{\sqrt{2}}{e^{\frac{\sqrt{d}-4}{\sqrt{2}}}} \leq 1 + \sqrt{2}e^{2\sqrt{2}}e^{-\sqrt{d/2}} < 1 + 24e^{-\sqrt{d/2}}. \tag{6}$$

It remains to consider the case where c satisfies this inequality and $c - 1 < t \leq \frac{c}{e^{\frac{\sqrt{d}-4}{c}}}$. Here we use the fact that $a \leq d - 1$. So we also have the bound

$$F_c(\mathbf{x}) \geq (t/c)(1 - t/c)^{d-1}.$$

The function $g(t) = (t/c)(1 - t/c)^{d-1}$ on the right-hand side has a maximum at $t = c/d$, as is easily verified. Assume that d is sufficiently large so that

$$\frac{c}{e^{\frac{\sqrt{d}-4}{c}}} \leq \frac{c}{d} \ .$$

Then, for $c - 1 < t \leq \frac{c}{e^{\frac{\sqrt{d}-4}{c}}} \leq \frac{c}{d}$, we have $F_c(\mathbf{x}) > g(c-1) = (c-1)/c^d$.

However, for d sufficiently large, we have

$$\frac{c-1}{c^d} \geq \left(1 - \frac{1}{c\sqrt{d}}\right)^d = s_d(c).$$

Indeed, recall that c is assumed to satisfy (3) and (6); that is,

$$1 + 2e^{-\sqrt{d/2}} \leq c < 1 + 24e^{-\sqrt{d/2}}.$$

We then have (using the inequality $1 + x < e^x$ for $x > 0$)

$$\frac{c-1}{c^d} > \frac{2e^{-\sqrt{d/2}}}{(1+24e^{-\sqrt{d/2}})^d} > \frac{2e^{-\sqrt{d/2}}}{e^{24de^{-\sqrt{d/2}}}},$$

which, for d sufficiently large, will be larger than

$$e^{-\sqrt{d/2}} \geq e^{-\sqrt{d}/c} \geq \left(1 - \frac{1}{c\sqrt{d}}\right)^d ,$$

as claimed.

In summary, we have obtained the following result.

Theorem 2.1 *There exists some threshold dimension d_0 so that, for every $d \geq d_0$ and for every c satisfying (3), we have $F_c(\mathbf{x}) \geq s_d(c)$ for every $\mathbf{x} \in \sigma$. As a consequence, the probability of capturing both endpoints of a unit vector in $\mathbb{R}^d$ inside the same cell of a randomly shifted grid of cell size c is at least $s_d(c)$.*

Remarks. (1) The analysis so far has only considered unit vectors, but it can easily be adapted to apply to vectors of smaller length. The simplest way of doing this is perhaps as follows. If the length of the vector uv is $r < 1$ then, by scaling space by the factor $1/r$, we turn uv into a unit vector, and c is replaced by the larger parameter c/r. The probability of capturing u and v in the same cell is now at least $s_d(c/r)$, which is larger than $s_d(c)$.

(2) It is perhaps tempting to use a larger value of c so as to increase the probability $s_d(c)$ of capturing u and v in the same cell. The value of $s_d(c)$, for c close to 1, e.g., for c meeting the lower bound in (3), is rather small; namely it is approximately $e^{-\sqrt{d}/c} \approx e^{-\sqrt{d}}$. However, to make this probability really large, we need to choose c quite large, about $\sqrt{d}$. The penalty for using such a large grid size will be incurred in having to inspect too many pairs of points in P. This trade-off, between (i) the success probability of capturing a unit vector in a grid cell, and (ii) the ratio between the number of pairs at Euclidean distance at most 1 and the overall number of pairs in a grid cell, is a major theme in our analysis, here and in the following sections.

(3) It would be interesting to obtain a sharp calibration of the value of the threshold dimension d_0, and to analyze the situation when $d < d_0$.

As already noted at the beginning of the section, if we repeat the randomly shifted grid construction $\frac{b}{s_d(c)} \log n$ (or just $\frac{b}{s_d(c)}$) times, for some $b > 2$ sufficiently large, we ensure, with high probability, that every pair of points at Euclidean distance at most 1 will be captured in the same cell of one of the grids (or ensure an arbitrarily large expected number of such pairs). This however is only one side of the story, because, as noted in Remark (2) above, the algorithm examines all pairs of points in the same grid cell, and we wish to ensure that the latter number is not significantly larger than the number of pairs, within the same cell, at Euclidean distance at most 1. This analysis will be undertaken in the next section.

3 The number of Euclidean neighbors in a grid cell

Let $\mathcal{G}$ be a randomly shifted grid, of cell size c, with c satisfying the condition in (3). Let τ be a cell of $\mathcal{G}$ and put $P_\tau = P \cap \tau$. The algorithm of Section 2 examines every pair of points

of P_τ, and reports those pairs at Euclidean distance at most 1. In this section we argue that the number of pairs examined by the algorithm, namely $\binom{|P_\tau|}{2}$, is comparable with the number $N_2(P_\tau)$ of pairs of points of P_τ at Euclidean distance at most 1. Specifically, we show that $N_2(P_\tau) \geq \gamma(d)\binom{|P_\tau|}{2} - \frac{1}{2}|P_\tau|$, where $\gamma(d)$ is some constant fraction that depends on d.

To establish this inequality, we fix a grid cell τ of size $c > 0$ (we will later on consider values of c smaller than 1, and the analysis in this section applies to these choices too). For simplicity, we regard P_τ as the entire set P, and denote its size by n.

An upper bound on $\gamma(d)$. To understand what values of $\gamma(d)$ one might expect to get, we start with a simple upper bound on that quantity. Let P be a set of n points drawn independently and uniformly at random in τ. A ball of radius 1 around one such point p contains, in expectation, at most $\frac{V_d \cdot (n-1)}{c^d}$ points of $P \setminus \{p\}$, where

$$V_d = \frac{\pi^{d/2}}{\Gamma(1 + d/2)}$$

is the volume of a unit ball in $\mathbb{R}^d$; $\Gamma(\cdot)$ is the Gamma function; see, e.g., [38, p. 136]. This bound on the expectation follows from the fact that the points of P are drawn independently, so the probability of another point $q \in P$ to fall into the ball is the volume of its portion within τ divided by $\mathrm{Vol}(\tau) = c^d$. The point p forms with each point $q \in P$ in this ball a pair at Euclidean distance at most 1, and every such pair (p, q) arises exactly twice in this manner, once for the ball centered at p and once for the ball centered at q.

Summing up over all points $p \in P$, we get that the expected number of pairs at Euclidean distance at most 1 is at most

$$\frac{n}{2} \cdot \frac{V_d \cdot (n-1)}{c^d} = \binom{n}{2} \frac{V_d}{c^d} \ . \tag{7}$$

This shows that we cannot hope for a worst-case lower bound for $\gamma(d)$ larger than V_d/c^d. Using the above expression for V_d and applying Stirling's approximation, we get that, up to factors polynomial in d,

$$\gamma(d) \leq \frac{V_d}{c^d} = \frac{\pi^{d/2}}{\Gamma(1 + d/2)c^d} \approx \frac{\left(\sqrt{2\pi e}\right)^d}{\sqrt{d}^d c^d} \ . \tag{8}$$

In other words, in this case (assuming that c is sufficiently close to 1) the number of pairs that the algorithm examines within τ is at least $\sqrt{d}^d c^d / \left(\sqrt{2\pi e}\right)^d \approx \sqrt{d}^d / 4.14^d$ times the number of pairs at Euclidean distance 1. This shows that to get rid of the super-exponential factor $\sqrt{d}^d$ we have to use a very small value of c, proportional to $1/\sqrt{d}$. We clearly cannot do that when only random translations are used. However, as will be shown in Section 5, one can choose such small values for c when random rotations are also employed.

A lower bound for $\gamma(d)$. We partition the cell τ of size c into a grid of $(c\sqrt{d})^d$ smaller cells of size $\frac{1}{\sqrt{d}}$ each. Let ξ be one of the small cells, and let n_ξ be the number of points

of P in ξ. Every pair of points in ξ are at distance at most 1, so we have at least $\sum_\xi \binom{n_\xi}{2}$ pairs at distance at most 1 in τ. Using the Cauchy-Schwarz inequality we get that

$$N_2(P_\tau) \geq \sum_\xi \binom{n_\xi}{2} = \sum_\xi \frac{n_\xi(n_\xi - 1)}{2} \geq \frac{n^2}{2(c\sqrt{d})^d} - \frac{n}{2}. \tag{9}$$

Another lower bound for $\gamma(d)$. We enlarge τ by shifting each of its facets f by $1/2$ away from τ in the direction orthogonal to f; let τ^* denote the cube formed by the hyperplanes that support the shifted facets. We then consider a ball of radius $1/2$ whose center is picked uniformly at random in τ^*. The probability that this ball contains a point $p \in \tau$ is at least $V_d(\frac{1}{2})^d/(c+1)^d$, which is the volume of such a ball divided by the volume of τ^*. (The random ball contains p iff we pick its center in a ball of radius $1/2$ around p, and this latter ball is contained in its entirety in τ^*.) It follows, in particular, that there is a ball of radius $1/2$ that contains at least $\frac{V_d(\frac{1}{2})^d}{(c+1)^d} n$ points of P. We take one such ball, and remove from P all points in that ball. As long as we have not removed more than $n/2$ points, a probabilistic argument similar to the one just given implies that there is a ball of radius $1/2$ containing at least $\frac{V_d(\frac{1}{2})^d}{(c+1)^d} \frac{n}{2}$ remaining points of P. So we keep collecting such balls, each containing a set of at least $\frac{V_d(\frac{1}{2})^d}{(c+1)^d} \frac{n}{2}$ points of P, so that these sets are pairwise disjoint, until we are left with fewer than $n/2$ points. The number of balls that we collect is therefore at most $\frac{(c+1)^d}{V_d(\frac{1}{2})^d}$. Let n_i be the number of points associated with the ith ball. By construction we have $\sum_i n_i \geq n/2$. Since each pair of points associated with a particular ball are at Euclidean distance at most 1, the Cauchy-Schwarz inequality implies, as above, that at least

$$\sum_i \binom{n_i}{2} = \sum_i \frac{n_i(n_i-1)}{2} \geq \frac{n^2}{8} \frac{V_d(\frac{1}{2})^d}{(c+1)^d} - \frac{n}{2} \tag{10}$$

pairs of points are at Euclidean distance at most 1. Substituting the explicit expression for V_d, and using Stirling's approximation, as in Equation (8), we get that this lower bound is approximately

$$\frac{n^2}{8} \frac{\left(\sqrt{\pi e/2}\right)^d}{\sqrt{d}^d (c+1)^d} - \frac{n}{2}\,.$$

The main drawback in this second lower bound is the presence of the factor $(c+1)^d$ in the denominator. (If we could have replaced $(c+1)^d$ by c^d then since $\sqrt{\pi e/2} \approx 2.07$ the second lower bound in Equation (10) would have clearly dominated the first lower bound in Equation (9).) The factor $(c+1)^d$ essentially makes the lower bound in Equation (10) useless when c is very small, such as $O(\frac{1}{\sqrt{d}})$; such small values of c will indeed be used in Section 5. In the next subsection we overcome this drawback by drawing the center of the random ball of radius $1/2$ in the Minkowski sum of τ and a ball of radius $1/2$, rather than in the much larger cell τ^* (and we will show that the Minkowski sum is indeed much smaller than τ^*).

3.1 The Minkowski sum of a cube and a ball

Let K denote the unit cube $[0,1]^d$ in $\mathbb{R}^d$, and let $r > 0$ be a parameter. Let B_r denote the ball of radius r centered at the origin, and let $K_r = K \oplus B_r$ denote their Minkowski

sum. Our next task is to estimate the volume of K_r, which is accomplished in the following lemma.[3]

Lemma 3.1

$$\mathrm{Vol}(K_r) \leq \beta e^{\frac{3}{2}(2\pi)^{1/3} d^{2/3} r^{2/3}}, \tag{11}$$

where β is a constant that depends polynomially on d.

Proof. We note that $\mathrm{Vol}(K_r)$ can be expressed as a special case of Steiner's formula, which asserts that, for any convex body K in $\mathbb{R}^d$,

$$\mathrm{Vol}(K \oplus B_r) = \sum_{j=0}^{d} \binom{d}{j} W_j(K) r^j,$$

where $W_j(K)$, called the j-th *quermassintegral* of K, is the *mixed volume*

$$V(\underbrace{K, K, \ldots, K}_{d-j \text{ times}}, \underbrace{B, B, \ldots, B}_{j \text{ times}})$$

of $d-j$ copies of K and j copies of the unit ball B; see, e.g., [20, 37]. As follows from the forthcoming analysis, the unit cube K has the property that $W_j(K)$ is the volume V_j of the j-dimensional unit ball, for $j = 0, \ldots, d$. So far we did not manage to find in the literature an explicit statement of this property, although we are fairly certain that it has been made. It can be established using properties of Minkowski sums involving segments (the cube, as well as any of its faces, is such a sum); see, e.g., formula (A.43), p. 407, in [20]. We establish this property in the following explicit manner.

We first obtain the following representation of K_r. For each $j = 0, \ldots, d$ there are $2^j \binom{d}{j}$ $(d-j)$-faces of K, so that each such face f is defined in terms of a subset σ of $\{1, \ldots, d\}$ of size j and a mapping $\delta : \sigma \mapsto \{0, 1\}$, so that

$$f = \{\mathbf{x} \in K \mid x_i = \delta(i), \text{ for } i \in \sigma\}.$$

That is, f is defined by fixing j coordinates, each to 0 or 1. Given σ and δ, we denote by $f_{\sigma,\delta}$ the corresponding face of K.

For each such face $f_{\sigma,\delta}$, we "push it outwards" by forming its Minkowski sum with an appropriate portion of a j-dimensional ball of radius r, in the directions of the coordinates that are fixed on $f_{\sigma,\delta}$. Technically, for $i \in \sigma$ and for $\varepsilon \in \{0, 1\}$, put

$$H_{i,\varepsilon} = \begin{cases} \{x_i \geq 0\} & \text{if } \varepsilon = 1 \\ \{x_i \leq 0\} & \text{if } \varepsilon = 0. \end{cases}$$

Now put

$$K_r(f_{\sigma,\delta}) = f_{\sigma,\delta} \oplus B_r^{(j)}(\delta),$$

where $B_r^{(j)}$ is the j-dimensional ball of radius r centered at the origin, within the subspace spanned by the fixed coordinates of σ, and

$$B_r^{(j)}(\delta) = B_r^{(j)} \cap \bigcap_{i \in \sigma} H_{i,\delta(i)}.$$

[3] Note that we simplify the analysis by considering a unit cube K instead of the grid cell τ. We will later scale the bound that we obtain to fit it to the actual setup with an arbitrary cell size c.

With all these preparations, we can finally write K_r as the *disjoint* union

$$K_r = \bigcup_{j=0}^{d} \bigcup_{\sigma,\delta} K_r(f_{\sigma,\delta}),$$

where the inner union is over all $\binom{d}{j}$ possible choices σ of j coordinates and 2^j 0/1 "sign patterns" δ on these coordinates. Note that the special case $j = 0$ yields K itself (there are no coordinates to be "pushed"). (The proofs of the above equality and of the disjointness of the constituents of the union are straightforward albeit a bit tedious, and we omit them.)

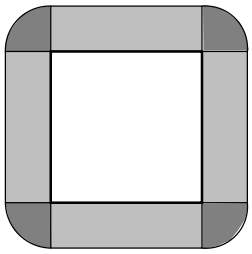

Figure 1: The Minkowski sum of a square and a disk.

The reader is invited to check Figure 1 for an illustration of the case $d = 2$, and to work out the concrete form of the above expression for $d = 3$, where the "fringe" of K_r consists of six boxes (obtained by pushing out the facets), twelve quarters of cylinders (obtained by pushing out the edges), and eight eighths of balls (obtained by pushing out the vertices).[4]

The volume of K_r. Note that each of the sums $K_r(f_{\sigma,\delta}) = f_{\sigma,\delta} \oplus B_r^{(j)}(\delta)$ is a $(1/2^j)$-portion of the cylinder $f_{\sigma,\delta} \oplus B_r^{(j)}$ whose axis is $f_{\sigma,\delta}$ and whose radius is r. Since the volume of any face f of K (within its affine hull) is always 1, the volume of $K_r(f_{\sigma,\delta})$ is $1/2^j$ of the volume of $B_r^{(j)}$, which we can write as $V_j r^j$, where, as before,

$$V_j = \frac{\pi^{j/2}}{\Gamma(1 + j/2)}$$

is the volume of a unit j-dimensional ball. Altogether we thus obtain

$$\mathrm{Vol}(K_r) = \sum_{j=0}^{d} \binom{d}{j} \cdot 2^j \cdot \frac{1}{2^j} \cdot V_j r^j = \sum_{j=0}^{d} \binom{d}{j} \frac{\pi^{j/2} r^j}{\Gamma(1 + j/2)}. \tag{12}$$

Note that this establishes our claim that the j-th quermassintegral $W_j(K)$ of the unit cube K is V_j, for each $j = 0, \ldots, d$.

We approximate the sum, up to factors that depend polynomially on d, as follows. We use the inequalities

$$\binom{d}{j} \leq \frac{d^j}{j!} \qquad \text{and}$$

[4]The 3-dimensional case is also presented in detail in [20].

$$\Gamma(1+j/2) \approx \frac{j(j-2)(j-4)\cdots}{2^{j/2}} \geq \frac{(j!)^{1/2}}{2^{j/2}},$$

where $\approx$ denotes equality up to a factor that depends *polynomially* in d. Hence

$$\mathrm{Vol}(K_r) \leq \beta_0 \sum_{j=0}^{d} \frac{d^j}{j!} \cdot \frac{2^{j/2}}{(j!)^{1/2}} \cdot \pi^{j/2} r^j,$$

for a suitable constant β_0 depending (polynomially) on d. Using Stirling's formula we further obtain that

$$\mathrm{Vol}(K_r) \leq \beta_1 \sum_{j=0}^{d} \left(\frac{2^{1/2}\pi^{1/2}e^{3/2}dr}{j^{3/2}} \right)^j, \tag{13}$$

for another suitable constant β_1.

To obtain an upper bound for this sum, we replace its terms by the corresponding function

$$f(x) = \left(\frac{A}{x^{3/2}} \right)^x, \qquad \text{where} \qquad A = 2^{1/2}\pi^{1/2}e^{3/2}dr,$$

over $x \geq 1$. It is simpler to analyze

$$g(x) = \ln f(x) = x \ln A - \frac{3}{2} x \ln x.$$

We have

$$g'(x) = \ln A - \left(\frac{3}{2} \ln x + \frac{3}{2} \right),$$

so $g'(x) = 0$ at $x = x_0 = A^{2/3}/e = (2\pi)^{1/3} d^{2/3} r^{2/3}$. It can easily be checked that $g(x)$ attains its maximum at this value, and so does $f(x)$. The maximum value of f is therefore

$$f(x_0) = f(A^{2/3}/e) = e^{\frac{3}{2}x_0} = e^{\frac{3}{2}(2\pi)^{1/3} d^{2/3} r^{2/3}}.$$

Hence $\mathrm{Vol}(K_r)$ is at most $(d+1)\beta_1 f(x_0)$; that is,

$$\mathrm{Vol}(K_r) \leq \beta f(x_0) = \beta e^{\frac{3}{2}(2\pi)^{1/3} d^{2/3} r^{2/3}},$$

for another suitable constant $\beta = (d+1)\beta_1$ that depends polynomially on d. This completes the proof of Lemma 3.1. □

Hence, as long as $r = o(d^{1/2})$, $\mathrm{Vol}(K_r) = e^{o(d)}$.

Remark. To appreciate the significance of this bound, note that if $r = \sqrt{d}$ (choosing r to be any constant multiple of $\sqrt{d}$ would lead to a similar phenomenon) then K_r fully contains the cube $[-1, 2]^d$, namely the cube obtained by "pushing" K by 1 in all directions. Indeed the points of the expanded cube that are farthest from K are its vertices, and each of them lies at distance $\sqrt{d}$ from a corresponding vertex of K. In other words, for $r = \sqrt{d}$ the volume of K_r is at least 3^d, but for sufficiently smaller values of $r \ll \sqrt{d}$, its volume becomes, relatively speaking, negligible. More precisely, it is then larger than $\mathrm{Vol}(K) = 1$ by only a sub-exponential factor.

The case of a grid cell of size c. We can apply the preceding analysis to the case where K is a grid cell of arbitrary size $c \neq 1$. To do so, we simply scale d-space by the factor $1/c$, so K becomes a unit cube and B_r becomes the ball $B_{r/c}$. We then scale space back, and obtain the bound

$$\text{Vol}(K_r) \leq \beta c^d e^{\frac{3}{2}(2\pi)^{1/3}(dr/c)^{2/3}}. \tag{14}$$

The preceding remark now observes that $\text{Vol}(K_r)/\text{Vol}(K)$ is sub-exponential as long as $r \ll c\sqrt{d}$. This will become significant in the next section, where we choose $c = \Theta(1/\sqrt{d})$.

An improved lower bound for $\gamma(d)$. We next improve our second lower bound, using the bound on $\text{Vol}(K_r)$ just derived (in (14)). We now draw a ball of radius $1/2$ whose center is picked uniformly at random in $K_{1/2}$, rather than in the enlarged cube. A probabilistic argument similar to the one given above shows that there exists such a ball that contains at least $\frac{V_d\left(\frac{1}{2}\right)^d}{\text{Vol}(K_{1/2})}n$ points of P. We remove these points and repeat the drawing process. The same argument as before implies that at least

$$\frac{n^2}{8} \cdot \frac{V_d\left(\frac{1}{2}\right)^d}{\text{Vol}(K_{1/2})} - \frac{n}{2} \geq \beta' n^2 \frac{\left(\sqrt{\pi e/2}\right)^d}{\sqrt{d}^d c^d e^{\frac{3}{2}(\pi/2)^{1/3}(d/c)^{2/3}}} - \frac{n}{2}$$

pairs of points are at Euclidean distance at most 1, where β' is yet another constant that depends polynomially on d, and where we have estimated $\text{Vol}(K_{1/2})$ using (14), and have approximated V_d using Stirling's formula as before. We summarize this finding in the following lemma.

Lemma 3.2 *Let P be a set of n points in a cube of size $c > 0$ in $\mathbb{R}^d$. Then at least*

$$\beta' n^2 \frac{\left(\sqrt{\pi e/2}\right)^d}{\sqrt{d}^d c^d e^{\frac{3}{2}(\pi/2)^{1/3}(d/c)^{2/3}}} - \frac{n}{2}$$

pairs of points of P are at Euclidean distance at most 1, where β' is a constant that depends polynomially on d.

In other words, the constant fraction multiplying n^2 that we get here is larger than the one we got before roughly by the factor

$$\left(\frac{c+1}{c}\right)^d \cdot \frac{1}{e^{\frac{3}{2}(\pi/2)^{1/3}(d/c)^{2/3}}} \approx e^{d/c - \frac{3}{2}(\pi/2)^{1/3}(d/c)^{2/3}},$$

which is a significant improvement as long as $c \ll d$.

Combining the bound in Lemma 3.2 with the one in Section 2, in which we approximate $s_d(c)$ by $e^{-\sqrt{d}/c}$, we obtain the following summary result.

Theorem 3.3 *Given a set P of n points in $\mathbb{R}^d$ and a parameter $c \geq 1 + 2e^{-\sqrt{d/2}}$, we can report, with high probability, all pairs of points of P at Euclidean distance at most 1 in time $C(d)(n+k)\log n$, where*

$$C(d) = O\left(e^{\sqrt{d}/c} \frac{\left(\sqrt{d}\right)^d c^d e^{\frac{3}{2}(\pi/2)^{1/3}(d/c)^{2/3}}}{\left(\sqrt{\pi e/2}\right)^d}\right).$$

Here the "big-O" notation only hides factors depending polynomially on d, and k is the output size. Alternatively, we can report, in expectation, an arbitrarily large fraction of the pairs of points of P at Euclidean distance at most 1 in time $C(d)(n+k)$, with the same asymptotic bound on $C(d)$. The probability of reporting all pairs in the first variant, and the expected fraction of reported pairs in the second variant, can be made arbitrarily close to 1 by increasing $C(d)$ by a sufficiently large absolute constant factor.

Remark. One might attempt to minimize the value of $C(d)$ as a function of c. Straightforward calculation shows that $C(d)$ is minimized at $c = c_0 = z^3/\sqrt{d} \approx 2.6/\sqrt{d}$, where $z \approx 1.375$ is the positive root of

$$z^3 - \left(\tfrac{\pi}{2}\right)^{1/3} z - 1 = 0.$$

Of course, we cannot choose this value of c, which is much smaller than 1 (for $d \geq 7$). Since $C(d)$ increases for $c > c_0$, we conclude that our best choice for c is the smallest possible value, namely, $c = 1 + 2e^{-\sqrt{d/2}}$. This implies, as is easily checked, the following corollary.

Corollary 3.4 *With the above choice of c, the running time of the algorithm is*

$$O\left(e^{\sqrt{d}}\frac{(\sqrt{d})^d e^{\frac{3}{2}(\pi/2)^{1/3}d^{2/3}}}{(\sqrt{\pi e/2})^d}\right)(n+k)\log n,$$

for reporting, with high probability, all pairs at Euclidean distance at most 1, and it is

$$O\left(e^{\sqrt{d}}\frac{(\sqrt{d})^d e^{\frac{3}{2}(\pi/2)^{1/3}d^{2/3}}}{(\sqrt{\pi e/2})^d}\right)(n+k),$$

for reporting, in expectation, an arbitrarily large fraction of the pairs; the "big-O" notation hides factors depending polynomially on d, and k is the output size.

Remark. Ignoring subexponential factors, the preceding bound is roughly $\sqrt{d}^d/\sqrt{\pi e/2}^d$, which is roughly twice as large as the lower bound.

3.2 Packing lemma

As another interesting application of Lemma 3.1, we derive in this subsection the following so-called *packing lemma*. It is reminiscent of a similar packing lemma used in Lenhof and Smid [32], and it can also be regarded as a more natural extension of Lemma 3.2, in which the points were confined to lie in a fixed cube.

Lemma 3.5 (Packing Lemma) *Let P be a set of n points in $\mathbb{R}^d$, and let $r > 0$ be a parameter. Let $N_\infty(P)$ (resp., $N_2(P)$) be the number of pairs of points of P at L_∞-distance (resp., L_2-distance) at most r. Then we have*

$$N_\infty(P) \leq B_d(N_2(P) + n),$$

where

$$B_d = O\left((4/\sqrt{2\pi e})^d d^{d/2} e^{\frac{3}{4}(9\pi)^{1/3}d^{2/3}}\right),$$

where the "big-O" hides factors depending polynomially on d.

Proof. As done so far in this paper, we may assume, without loss of generality, that $r = 1$. Cover $\mathbb{R}^d$ by balls of diameter 1. An old result of Rogers (see, e.g., [12, Section 8.2]) asserts that this can be done (by a lattice covering) so that each point lies in at most $\lambda = O(d \log d)$ balls. Let $\mathcal{B}$ denote the (necessarily finite) sub-collection of the cover consisting of those balls containing points of P. For each ball $B \in \mathcal{B}$, let n_B denote the number of points of $P \cap B$. We clearly have $\sum_{B\in\mathcal{B}} n_B \leq \lambda n$ and $\sum_{B\in\mathcal{B}} \binom{n_B}{2} \leq \lambda N_2(P)$.

Let K denote the cube $[-1, 1]^d$, and let $B(r)$ denote the ball of radius r centered at the origin, for any $r > 0$. Let p and q be a pair of points of P with $\|p-q\|_\infty \leq 1$. Let B_p and B_q be balls of $\mathcal{B}$ containing p and q, respectively, and assume, without loss of generality, that $n_{B_q} \leq n_{B_p}$. Since $q \in p + K$, it follows that $B_q \subset B_p \oplus K \oplus B(1)$. Since (i) the Minkowski sum is associative, (ii) B_p is congruent to $B(1/2)$, and (iii) $B(1/2) \oplus B(1) = B(3/2)$, an application of Lemma 3.1, or rather of the bound in (14), with $c = 2$ and $r = 3/2$, yields

$$\mathrm{Vol}(B_p \oplus K \oplus B(1)) \leq \beta 2^d e^{\frac{3}{4}(9\pi)^{1/3}d^{2/3}}.$$

The number M of balls of $\mathcal{B}$ that are fully contained in $B_p \oplus K \oplus B(1)$ satisfies

$$\begin{aligned} M &\leq \frac{\lambda \mathrm{Vol}(B_p \oplus K \oplus B(1))}{V_d(1/2)^d} \\ &\leq \frac{\lambda\beta 4^d e^{\frac{3}{4}(9\pi)^{1/3}d^{2/3}}\Gamma(1 + d/2)}{\pi^{d/2}} \\ &= O\left((4/\sqrt{\pi})^d e^{\frac{3}{4}(9\pi)^{1/3}d^{2/3}}(d/(2e))^{d/2}\right) \\ &= O\left((4/\sqrt{2\pi e})^d d^{d/2} e^{\frac{3}{4}(9\pi)^{1/3}d^{2/3}}\right), \end{aligned}$$

where the final constant of proportionality depends polynomially on d.

We charge the pair (p, q) to B_p, and note that every ball $B \in \mathcal{B}$ is charged by at most

$$Mn_B^2 = 2M\binom{n_B}{2} + Mn_B$$

such pairs (recall our assumption that $n_{B_q} \leq n_{B_p}$). Summing over all balls of $\mathcal{B}$, and using the inequalities observed at the beginning of the proof, we obtain

$$N_\infty(P) \leq M\sum_{B\in\mathcal{B}} n_B^2 = 2M\sum_{B\in\mathcal{B}}\binom{n_B}{2} + M\sum_{B\in\mathcal{B}} n_B \leq 2M\lambda N_2(P) + M\lambda n,$$

and the lemma follows. □

Remark. It is perhaps simpler to absorb the last sub-exponential factor of M in its first exponential factor, by slightly increasing its base. Also, this base is smaller than 1, so a simpler, slightly weaker form of the lemma is that $N_\infty(P) \leq \sqrt{d}^d(N_2(P) + n)$.

4 Low doubling dimension

As noted in the introduction, in many applications the input set P has a restricted structure, in the sense that its points have much fewer "degrees of freedom" than the ambient

dimension d. For example, all the points of P might lie on, or near, some manifold of much smaller dimension. In these cases one hopes for a better performance of the algorithm, in the sense that the constant $C(d)$ in the bounds in Theorem 3.3 can be replaced by a much smaller one. We show in this section that this is indeed the case.

One formal way of capturing such a special structure of P is through the notion of *doubling dimension*; see [26, 31]. Specifically, a metric space (X, ρ) has doubling dimension δ if every ball in X of any radius r can be covered by at most 2^δ balls of radius $r/2$. In what follows we assume that P, under the Euclidean metric in d-space, has doubling dimension $\delta \ll d$.

The mechanism that has led to Theorem 3.3 can easily be adapted to this case. Specifically, we use a randomly shifted grid of cell size c, as in Section 2, and conclude that any fixed unit vector $\mathbf{x}$ is captured in a grid cell with probability at least $s_d(c) \approx e^{-\sqrt{d}/c}$. (This part of the analysis does not seem to benefit from the fact that P has low doubling dimension, but at any rate the dependence of $s_d(c)$ on d is "tame", that is, sub-exponential.)

What can be improved is the analysis that leads to Lemma 3.2. Specifically, fix a grid cell τ, containing n_τ points of P. Observe that τ is fully contained in a ball of radius $c\sqrt{d}/2$, and in a ball of radius $c\sqrt{d}$ centered at any point of $P_\tau = P \cap \tau$. It easily follows from the definition of doubling dimension that we can cover P_τ by $(2c\sqrt{d})^\delta$ balls of radius $1/2$. Within each such ball, every pair of points is at Euclidean distance at most 1.

Let $\mathcal{B}$ be a minimum-size cover of P_τ by balls of radius $1/2$. As just shown, we have $M = |\mathcal{B}| \leq (2c\sqrt{d})^\delta$. We claim that no point of P_τ is contained in more than 2^δ balls of $\mathcal{B}$. Indeed, suppose to the contrary that there exists a point $p \in P_\tau$ that is contained in more than 2^δ balls of $\mathcal{B}$. Then all these balls are fully contained in the ball $B_1(p)$ of radius 1 centered at p. Cover $P \cap B_1(p)$ by at most 2^δ balls of radius $1/2$, and replace the balls of $\mathcal{B}$ that are contained in $B_1(p)$ by these 2^δ balls. This gives us a cover of P_τ by fewer balls, a contradiction that establishes the claim.

It follows that no pair of points of P_τ can lie together in more than 2^δ balls of $\mathcal{B}$, so we have

$$\sum_{B \in \mathcal{B}} \binom{|P_\tau \cap B|}{2} \leq 2^\delta N_2(P_\tau),$$

where $N_2(P_\tau)$ is the number of pairs of points of P_τ at Euclidean distance at most 1. On the other hand, using the Cauchy-Schwarz inequality, as in the derivation of Equation (9), we have

$$\sum_{B \in \mathcal{B}} \binom{|P_\tau \cap B|}{2} \geq \frac{1}{2M} \left(\sum_B |P_\tau \cap B| \right)^2 - \frac{1}{2} \left(\sum_B |P_\tau \cap B| \right) \geq \frac{|P_\tau|^2}{2M} - 2^{\delta-1}|P_\tau| \ .$$

Combining the two inequalities, we obtain

Lemma 4.1 *Let P be a set of n points in a cube of size $c > 1/(2\sqrt{d})$ in $\mathbb{R}^d$, of doubling dimension δ (with respect to the Euclidean distance). Then at least*

$$\frac{n^2}{2(4c\sqrt{d})^\delta} - \frac{n}{2}$$

pairs of points of P are at Euclidean distance ≤ 1.

Corollary 4.2 *Given a set P of n points in $\mathbb{R}^d$ of doubling dimension δ (with respect to the Euclidean distance), and a parameter $c \geq 1 + 2e^{-\sqrt{d/2}}$, we can report, with high probability, all pairs of points of P at Euclidean distance at most 1 in time $C(d,\delta)(n+k)\log n$, where*

$$C(d,\delta) = O\left(e^{\sqrt{d}/c}(4c\sqrt{d})^{\delta}\right),$$

and where k is the output size. Alternatively, we can report, in expectation, an arbitrarily large fraction of the pairs of points of P at Euclidean distance at most 1 in time $C(d,\delta)(n+k)$, with the same asymptotic bound on $C(d)$. The probability of reporting all pairs in the first variant, and the expected fraction of reported pairs in the second variant, can be made arbitrarily close to 1 by increasing $C(d,\delta)$ by a sufficiently large absolute constant factor.

Remark. Simple calculation shows that the bound on $C(d,\delta)$ is minimized at $c = \sqrt{d}/\delta$, and then $C(d,\delta) = O((4ed/\delta)^{\delta})$. Of course, this choice of c makes sense only when δ is smaller than $\sqrt{d}$; otherwise we choose $c = 1 + 2e^{-\sqrt{d/2}}$, as in Corollary 3.4, and get $C(d,\delta) = O\left(e^{4\sqrt{d}}\sqrt{d}^{\delta}\right)$.

5 Randomly rotated and shifted grids

Clearly, in the preceding analysis we cannot choose $c \leq 1$ because some unit vectors, such as axis-parallel unit vectors, will have zero probability of being captured in a grid cell. However, as we show in this section, if we also apply a random rotation, we will be able to capture any vector of length (at most) 1 within a grid cell of size $c \leq 1$ (of a randomly shifted grid), with some fixed positive probability (depending, as usual, on c and d), provided that c is not too small. Although this probability will be significantly smaller than the corresponding probability obtained in the preceding sections, the small value of c will lead to much fewer repetitions of the procedure and consequently to a significant improvement in the overall running time of the algorithm.

Fix some vector $\mathbf{x}_0$ of length 1 (as before, this is clearly the worst-case assumption; vectors of smaller length will be captured with higher probability). The probabilistic model assumed in this section is that we choose a random rotation ρ of d-space (there are several ways of doing this; see, e.g., Genz [21] for an $O(d^2 \log d)$ procedure) followed by placing a randomly shifted grid of cell size c in the rotated space. We want to derive a lower bound on the probability of the event $A(\mathbf{x}_0)$ that both endpoints of the rotated image $\rho(\mathbf{x}_0)$ of $\mathbf{x}_0$ fall into the same grid cell.

The random rotation ρ maps $\mathbf{x}_0$ to a random vector $\mathbf{x} = \rho(\mathbf{x}_0)$ (uniformly) on σ. The probability of capturing $\mathbf{x} = (x_1, \ldots, x_d)$ in a cell of a randomly shifted grid of cell size $c \leq 1$ is

$$F_c(\mathbf{x}) = \left(1 - \frac{|x_1|}{c}\right)^+ \cdots \left(1 - \frac{|x_d|}{c}\right)^+ ,$$

where $w^+ = \max\{w, 0\}$, for $w \in \mathbb{R}$. This is the probability of the event $A(\mathbf{x}_0)$ conditioned on the choice of the rotation ρ, so the actual probability of $A(\mathbf{x}_0)$ is the expectation (average) of $F_c(\rho(\mathbf{x}_0))$ over the space of rotations.

The symmetry of the space of rotations is easily seen to imply that the expected value μ of $F_c(\rho(\mathbf{x}_0))$, over the space of rotations, is the same as the expected value of $F_c(\mathbf{x})$ over σ

(because a random rotation takes $\mathbf{x}_0$ to a random point on σ, and each point has the same (differential) probability of being reached). That is,

$$\mu = \frac{1}{|\sigma|}\int_\sigma F_c(\mathbf{x})d\mathbf{x},$$

where $|\sigma|$ is the surface volume of σ, which is (see [38])

$$|\sigma| = \frac{2\pi^{d/2}}{\Gamma(d/2)}.$$

To estimate μ, we use the easily established fact, observed by Muller [35], that if we draw d independent normal variables x_i, $1 \le i \le d$, from the standard normal distribution $\mathbf{N}(0;1)$ and put $\mathbf{x} = (x_1, \ldots, x_d)$, then $\mathbf{x}/\|\mathbf{x}\|$ is a uniform random direction (a random point on σ). It follows from this observation that

$$\mu = \frac{1}{(2\pi)^{d/2}}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\left(1-\frac{|x_1|}{c\|\mathbf{x}\|}\right)^+\cdots\left(1-\frac{|x_d|}{c\|\mathbf{x}\|}\right)^+ e^{-\|\mathbf{x}\|^2/2}dx_1\cdots dx_d\ . \tag{15}$$

To obtain a lower bound on μ we fix some threshold radius r_0, and write $\mu = \mu^- + \mu^+$, where μ^- (resp., μ^+) is the portion of μ obtained by integrating over the multivariate normal distribution within the region $\|\mathbf{x}\| \le r_0$ (resp., $\|\mathbf{x}\| \ge r_0$). Clearly $\mu \ge \mu^+$ so we focus on lower-bounding μ^+.

For $\|\mathbf{x}\| \ge r_0$ we have

$$\left(1-\frac{|x_1|}{c\|\mathbf{x}\|}\right)^+\cdots\left(1-\frac{|x_d|}{c\|\mathbf{x}\|}\right)^+ \ge \left(1-\frac{|x_1|}{cr_0}\right)^+\cdots\left(1-\frac{|x_d|}{cr_0}\right)^+,$$

so

$$\mu^+ \ge \frac{1}{(2\pi)^{d/2}}\int_{\|\mathbf{x}\|\ge r_0}\left(1-\frac{|x_1|}{cr_0}\right)^+\cdots\left(1-\frac{|x_d|}{cr_0}\right)^+ e^{-\|\mathbf{x}\|^2/2}dx_1\cdots dx_d.$$

Write this right-hand integral as $I - I_0$, where I (resp., I_0) is obtained by integrating over the entire $\mathbb{R}^d$ (resp., over $\|\mathbf{x}\| \le r_0$).

I is easy to compute (here we use the independence of the choice of coordinates x_i), and we get

$$\begin{aligned}
I &= \frac{1}{(2\pi)^{d/2}}\left(\int_{-\infty}^{\infty}\left(1-\frac{|x|}{cr_0}\right)^+ e^{-x^2/2}dx\right)^d \\
&= \frac{1}{(2\pi)^{d/2}}\left(2\int_0^{cr_0}\left(1-\frac{x}{cr_0}\right)e^{-x^2/2}dx\right)^d \\
&= \left(\frac{2}{\pi}\right)^{d/2}\left(\int_0^{cr_0}e^{-x^2/2}dx - \frac{1}{cr_0}\int_0^{cr_0}xe^{-x^2/2}dx\right)^d \\
&= \left(\frac{2}{\pi}\right)^{d/2}\left(\int_0^{cr_0}e^{-x^2/2}dx - \frac{1-e^{-(cr_0)^2/2}}{cr_0}\right)^d.
\end{aligned}$$

Using integration by parts, it is easily checked that

$$\frac{e^{-(cr_0)^2/2}}{cr_0} = \int_{cr_0}^{\infty}e^{-x^2/2}dx + \int_{cr_0}^{\infty}\frac{1}{x^2}e^{-x^2/2}dx,$$

which implies that

$$\int_0^{cr_0} e^{-x^2/2}dx + \frac{e^{-(cr_0)^2/2}}{cr_0} =$$
$$\int_0^{cr_0} e^{-x^2/2}dx + \int_{cr_0}^{\infty} e^{-x^2/2}dx + \int_{cr_0}^{\infty} \frac{1}{x^2}e^{-x^2/2}dx =$$
$$\int_0^{\infty} e^{-x^2/2}dx + \int_{cr_0}^{\infty} \frac{1}{x^2}e^{-x^2/2}dx \geq \sqrt{\pi/2}\ ,$$

where in the final inequality we neglect the second integral. Hence

$$I \geq \left(\frac{2}{\pi}\right)^{d/2}\left(\sqrt{\pi/2} - \frac{1}{cr_0}\right)^d = \left(1 - \frac{\sqrt{2}}{\sqrt{\pi}cr_0}\right)^d.$$

We obtain an upper bound for I_0 by upper bounding $\left(1 - \frac{|x_1|}{cr_0}\right)^+ \cdots \left(1 - \frac{|x_d|}{cr_0}\right)^+$ by 1, so we have

$$I_0 \leq \frac{1}{(2\pi)^{d/2}} \int_{\|\mathbf{x}\| \leq r_0} e^{-\|\mathbf{x}\|^2/2} dx_1 \cdots dx_d.$$

We express this integral in spherical coordinates (r, θ), where r is the radius (distance from the origin) and $\theta \in \sigma$ is the orientation. The spherical symmetry of the multivariate normal distribution is easily seen to yield

$$I_0 \leq \frac{1}{(2\pi)^{d/2}} \int_{\sigma} 1 d\theta \int_0^{r_0} r^{d-1} e^{-r^2/2} dr, \tag{16}$$

where in the first integral θ denotes a point on σ and $d\theta$ denotes a surface volume element on σ. Informally, the identity follows since a volume element at a point $\mathbf{x} = (r, \theta)$ can be written as $r^{d-1}d\theta$ (the surface volume element on the sphere through $\mathbf{x}$, of radius r) times dr.

We note that $\int_\sigma 1 d\theta$ is simply the surface volume of σ, which is $|\sigma| = \frac{2\pi^{d/2}}{\Gamma(d/2)}$, and we thus get

$$I_0 \leq \frac{1}{(2\pi)^{d/2}} \cdot \frac{2\pi^{d/2}}{\Gamma(d/2)} \int_0^{r_0} r^{d-1} e^{-r^2/2} dr = \frac{1}{2^{d/2-1}\Gamma(d/2)} \int_0^{r_0} r^{d-1} e^{-r^2/2} dr.$$

Replacing r by $\sqrt{2z}$, putting $r_0 = \sqrt{td}$, for some $0 < t$, and writing $d/2 = k$, we get

$$I_0 \leq \frac{1}{\Gamma(k)} \int_0^{tk} z^{k-1} e^{-z} dz.$$

The function $z^{k-1}e^{-z}$ is maximized at $z = k - 1$. If we assume that $t \leq 1 - 2/d$ then $k - 1 \geq tk$, so the integrand is an increasing function in the range $0 \leq z \leq tk$, and we get

$$\int_0^{tk} z^{k-1} e^{-z} dz \leq tk \cdot (tk)^{k-1} e^{-tk} = (tk)^k e^{-tk}.$$

We can also lower bound $\Gamma(k)$, using Stirling's formula, by $\sqrt{2\pi/k}(k/e)^k$. Hence we get

$$I_0 \leq (k/(2\pi))^{1/2}(e/k)^k (tk)^k e^{-tk}$$
$$= (k/(2\pi))^{1/2}(et)^k e^{-tk} = (k/(2\pi))^{1/2}(te^{1-t})^k.$$

Putting

$$\beta(t) := t^{1/2} e^{(1-t)/2},$$

we conclude that

$$I_0 \leq (k/(2\pi))^{1/2} \beta(t)^d. = (d/(4\pi))^{1/2} \beta(t)^d.$$

Note that $\beta(t) < 1$ for any $t < 1$.

Putting the lower bound on I and the upper bound on I_0 together, we get

$$\mu \geq \mu^+ \geq I - I_0 \geq \left(1 - \frac{\sqrt{2}}{\sqrt{\pi} c r_0}\right)^d - (d/(4\pi))^{1/2} \beta(t)^d.$$

Choosing c. Note that we can, and indeed intend to, take c to be much smaller than 1. The only constraint that we need to enforce is that

$$1 - \frac{\sqrt{2}}{\sqrt{\pi} c r_0} > \left(\frac{d}{4\pi}\right)^{1/(2d)} \beta(t).$$

The maximum value of $\left(\frac{d}{4\pi}\right)^{1/(2d)}$ is smaller than 1.015, as is easily checked, so we replace this constraint by

$$1 - \frac{\sqrt{2}}{\sqrt{\pi} c r_0} > 1.015 \cdot \beta(t).$$

Recalling the choice of $r_0 = \sqrt{dt}$ and the definition of $\beta(t)$, this becomes

$$c > \frac{a(t)}{\sqrt{d}}$$

where

$$a(t) = \frac{\sqrt{2}}{\sqrt{\pi t}(1 - 1.015 t^{1/2} e^{(1-t)/2})} > \frac{\sqrt{2}}{\sqrt{\pi t}} \ . \tag{17}$$

That is, we can take t to be some constant fraction (the optimal choice of t will be discussed later in the section) and then take c to be slightly larger than $\frac{a(t)}{\sqrt{d}}$; that is, we take c to be some constant multiple of $1/\sqrt{d}$. In this case the (lower bound on the) expected success probability μ will decrease exponentially with d. In contrast, choosing c to be some fixed absolute fraction, similar to the choice in the purely translational case treated in Sections 2 and 3, yields a lower bound for μ of the form $e^{-\alpha\sqrt{d}/c}$, for $\alpha \approx \sqrt{\frac{2}{\pi t}}$ (in this case any choice of t not too close to 1 will do).

The rationale for taking c so small is that this (significant) degradation in the success probability will be more than offset by the ratio (arising in the analysis of our algorithm) between the number of pairs of points in a grid cell and the number of such pairs at Euclidean distance at most 1, making the combination of these two factors only exponential in d, rather than super-exponential, as was the case in Section 3.

Let $a(t)$ be as defined in (17), and let us choose $c = a/\sqrt{d}$, for some slightly larger $a > a(t)$. As can be easily checked, the smallest value for $a(t)$, attained at $t \approx 0.11$, is about 5.07. (Observe that we cannot choose c to be smaller than $1/\sqrt{d}$, because then no

unit vector will fit into a cell of size c.) The best choice of the constant a is different though, and is discussed next.

For $c = a/\sqrt{d}$, μ is at least roughly $\left(1 - \frac{\sqrt{2}}{a\sqrt{\pi t}}\right)^d$. That is, for any fixed unit vector $\mathbf{x}_0$, a random rotation followed by a placement of a randomly shifted grid of cell size c, for $c = a/\sqrt{d}$ as above, captures both endpoints of $\mathbf{x}_0$ in the same grid cell with probability at least $\left(1 - \frac{\sqrt{2}}{a\sqrt{\pi t}}\right)^d$, so the expected number of trials until a successful one (for the fixed vectors $\mathbf{x}_0$) is at most $M_0 = 1/\left(1 - \frac{\sqrt{2}}{a\sqrt{\pi t}}\right)^d$. More precisely, repeating this step $bM_0 \log n$ times ensures with high probability that every pair at Euclidean distance at most 1 is captured in a grid cell. Repeating this step only bM_0 times ensures that an arbitrarily large expected fraction of these pairs (a fraction approaching 1 when b increases) will be captured.

On the other hand, assume that a grid cell τ of size $c = a/\sqrt{d}$ contains n_τ points of P. Using Lemma 3.2, we conclude that the number of pairs of points of $P \cap \tau$ at Euclidean distance at most 1 is at least

$$\beta' n_\tau^2 \frac{\left(\sqrt{\pi e/2}\right)^d}{\sqrt{d}^d c^d e^{\frac{3}{2}(\pi/2)^{1/3}(d/c)^{2/3}}} - \frac{n_\tau}{2}.$$

Substituting $c = a/\sqrt{d}$, this becomes

$$\beta' \left(\frac{\sqrt{\pi e/2}}{a e^{\frac{3}{2}(\pi/(2a^2))^{1/3}}}\right)^d n_\tau^2 - \frac{n_\tau}{2}.$$

It follows that, with high probability, we report each of the k pairs of points at Euclidean distance at most 1 (resp., in expectation, an arbitrarily large fraction of these pairs) by inspecting no more than $C(d,a) \cdot (n+k) \log n$ (resp., $C(d,a) \cdot (n+k)$) pairs of points of P, where $C(d,a)$ is

$$\begin{aligned} C(d,a) &= O\left(\frac{1}{\left(1 - \frac{\sqrt{2}}{a\sqrt{\pi t}}\right)^d} \cdot \left(\frac{a e^{\frac{3}{2}\left(\pi/(2a^2)\right)^{1/3}}}{\sqrt{\pi e/2}}\right)^d\right) \\ &= O\left(\left(\frac{a^2 e^{\frac{3}{2}\left(\pi/(2a^2)\right)^{1/3}}}{a\sqrt{\pi e/2} - \sqrt{e/t}}\right)^d\right). \end{aligned}$$

As before the "big-O" notation here hides factors depending polynomially on d. Increasing $C(d,a)$ by a constant factor increases the overall success probability that we indeed report all pairs at distance 1 (or the constant fraction of the pairs reported).

We thus get rid of the super-exponential factor $d^{d/2}$ that we encounter when c is a fixed fraction or when no rotation is applied, and obtain an overall exponential factor.

Recall that $a = a(t)$ is a function of t given by Equation (17). So the best lower bound on the base in the "overhead" constant factor $C(d,a)$ that this method yields is about

$$\min_t \left\{ \frac{a(t)^2 e^{\frac{3}{2}\left(\pi/(2a(t)^2)\right)^{1/3}}}{a(t)\sqrt{\pi e/2} - \sqrt{e/t}} \right\},$$

and numerical calculations show that this minimum value is about 6.74, obtained for $t \approx 0.25$.

In conclusion, we obtain the following main result.

Theorem 5.1 *Given a set P of n points in $\mathbb{R}^d$, we can report, with high probability, all pairs of points of P at Euclidean distance at most 1, in time $O(6.74^d(n+k)\log n)$, where k is the output size. Alternatively, we can report, in expectation, an arbitrarily large fraction of these pairs in time $O(6.74^d(n+k))$; the success probability and the expected fraction of reported pairs can be made arbitrarily close to 1 by increasing the constant factor hidden by the "big-O".*

Remark. The case where P has low doubling dimension $\delta \ll d$ does not seem to fit the framework of this section because, as in the preceding sections, we do not know how to make the success probability of capturing a unit vector within a grid cell depend also on δ. While this probability was only sub-exponentially small (in d) in the case of pure translation, here, with the very small value of c that we choose, this probability becomes exponentially small. This in turn makes the constant in the bound depend (at least) exponentially on d, a much larger value than that obtained in Section 4.

6 Experimental results

We have implemented both algorithms in C++ and have compared them to several leading software packages for reporting neighbors. These do not include algorithms that lay down a (fixed) grid of cell size r and test each point in each nonempty cell with the points in its neighboring nonempty cells, such as the one in [32]. These algorithms are very slow due to the large number of neighboring cells, each of which must be tested to determine whether it is empty or not.

A natural approach to solve our problem is by using certain "off the shelf" data structures for (approximate) nearest neighbor queries. We first construct such a data structure on our input set and then query it with each input point. Evidently in order to find all pairs of points at distance at most r we need each query with a point q to return all points at distance at most r from q, rather than just the nearest neighbor of q. Fortunately, the available packages do provide functions that can be adapted for this task. Since these packages approximate the nearest neighbors they in fact only find "most" neighbors of distance at most r. By fine-tuning the parameters of the appropriate package, we can control the expected fraction of the pairs at distance at most r that this package reports.

We have implemented such a scheme using three publicly available software packages: the E2LSH package for Locality Sensitive Hashing [2, 3], the ANN package by Arya et al. [7], and the FLANN package [33, 34]. ANN and FLANN use various hierarchical space decompositions (such as kd-trees and box decomposition trees).

The E2LSH package provides a function to learn the configuration parameters needed for a particular success probability. This function takes a data set to index, a set of queries, a target probability p, and a distance r. It then computes the parameters required to ensure that for each query q we report every point w at distance at most r from q, with probability p. We applied this function with target probability of $p = 0.9$ to a moderate-size data set

(arising from 900 images, see below) which we used both as the data set to index, and as the set of queries. Then in all our experiments we used the configuration parameters produced by this training phase.

The ANN and FLANN packages support a search for the k nearest neighbors and allow to stop the search when the distance to the jth neighbor (for some $j < k$) exceeds some prespecified value r. We used this search with a large k and our target distance r. As an outcome for a query point q and a parameter ε, we got all points p at distance at most $r/(1+\varepsilon)$ from q, and some points p at distance at most r from q (we never get points lying further away from q). We have set $\varepsilon = 1.4$ since for this value the number of points reported by ANN and FLANN was comparable to the number of points reported by E2LSH (with $p = 0.9$) on the training data set mentioned above. In our first algorithm, which uses a randomly shifted grid, we set the size of a grid cell to be $c = 3.6r$ and we used 5 different randomly shifted grids. For these parameters the total number of pairs that we report on the training data set was comparable to the number of pairs reported by the other packages with the parameters mentioned above.

Our points lie in Euclidean space of dimension $d = 64$, but $c/r = 3.6 < \sqrt{d} = 8$. This is exactly the situation mentioned in the introduction, where the easy lower bound of $1-r\sqrt{d}/c$ on the success probability is meaningless. Nevertheless our choice of c is not as small as our analysis suggests. The reason for that is the small intrinsic ("doubling") dimension of our data sets (which is probably closer to 10 than to 64). This small dimension makes the number of points that we see in a grid cell relatively small and as a result moderate values of c work better in practice.

Note that by running the training phase of E2LSH, it is guaranteed to report 90% of the pairs of distance at most r of each other. ANN and FLANN do not provide such a guarantee directly but they do so indirectly since we tuned their parameters to report about the same number of pairs as E2LSH. ANN and FLANN are guaranteed, however, to report all pairs at distance smaller than $r/(1+\varepsilon)$. Our theoretical analysis in Section 2 guarantees that a pair of distance r falls into the same grid cell of one random grid is at least $s_d(c/r) = s_d(3.6) \approx 0.1$. So the probability that we discover a close pair using 5 grids is $1 - 0.9^5 = 0.41$. The fact that with 5 grids we get an output size comparable to that of E2LSH indicates that our lower bound, $s_d(c/r)$, on the probability of locating each pair is pessimistic, and that this probability is larger in practice.

We have constructed our input set by extracting representative descriptors in $\mathbb{R}^{64}$ from images, using the Speeded Up Robust Feature (SURF) method [9], which is used for computer vision problems such as object recognition and 3D reconstruction. SURF first identifies interesting points at significant corner locations of the image. Then it summarizes the region around each interesting point, using the Haar wavelet transform, into a descriptor of 64 numbers in $[0, 1]$. The vectors of these descriptors form the points in $\mathbb{R}^{64}$ that we used as our input data set.

Finding all pairs of descriptors at distance at most r from each other is a common method for identifying similar images. Images are declared to be similar if there are sufficiently many pairs of descriptors, one from each image, at distance at most r from each other. We used $r = 0.08$ which is appropriate for the underlying image matching application. We used a collection of images containing pairs that were taken at the same geographic location or of the same object. Such pairs of images have many close descriptors. These pairs of close

descriptors are the pairs of points in $\mathbb{R}^{64}$ that our algorithm tries to identify. (For training LSH we used a set of descriptors extracted from 900 images from the same set.)

The table in Figure 2 shows the results for four data sets obtained respectively from 50, 100, 200, and 400 images. The second column gives the total number of points (in $\mathbb{R}^{64}$) extracted from these images. This is about 260 points per image. The third column gives the exact number of pairs of points at distance at most r. The last four columns give the results for the four algorithms that we compared. We show two numbers in each entry of these columns: the number of reported pairs at distance at most r, and in parenthesis the running time in seconds. We can see from this table that the fraction of reported pairs in all instances is about 96% of the total number of pairs at distance at most r, except for FLANN, which failed to report a sufficiently large number of pairs for 400 images. Automatic parameter selection for FLANN gave bad results and we set manually the parameters to get the best performance. Specifically, we set the index parameter to 4 and search parameter to 64. We can see that FLANN is considerably slower than all the other algorithms. Our algorithm is the fastest.

#images	#pts	#r-close pairs	ANN	FLANN	LSH	Ours
50	12981	1024	998 (0.31)	1012 (1.19)	1013 (0.46)	1001 (0.07)
100	25532	1784	1734 (0.64)	1755 (2.3)	1755 (0.89)	1705 (0.16)
200	53106	4628	4347 (1.61)	4154 (5.12)	4532 (1.95)	4405 (0.46)
400	106336	8538	8040 (4.46)	7295 (10.4)	8319 (4.16)	8416 (1.17)

Figure 2: Comparison of the numbers of reported points and the running times (in seconds) for the different algorithms.

To appreciate the asymptotic improvement of our algorithm, we also compared it to ANN and E2LSH on much larger data sets. FLANN is much slower and was not able to compete with the other algorithms on these instances. Figure 3 shows the running times of the three algorithms as a function of the size of the data set. The asymptotic supremacy of our method can clearly be observed. The reported number of pairs is shown in Figure 4.

6.1 Using rotations

The number of pairs of points in a grid cell of size c typically increases rapidly with c, so, as discussed earlier, we want to use very small values of c which, combined with a random rotation of the coordinate frame, yields a procedure for which we were able to obtain a much improved theoretical upper pound on its expected performance.

On our real data sets, however, as it turns out, rotations do not yield an advantage. This is probably due to the fact that the data sets have low doubling dimension, and that the algorithm that uses rotation does not take advantage of this property, as does the purely translational algorithm. Decreasing the value of c indeed decreases the number of pairs

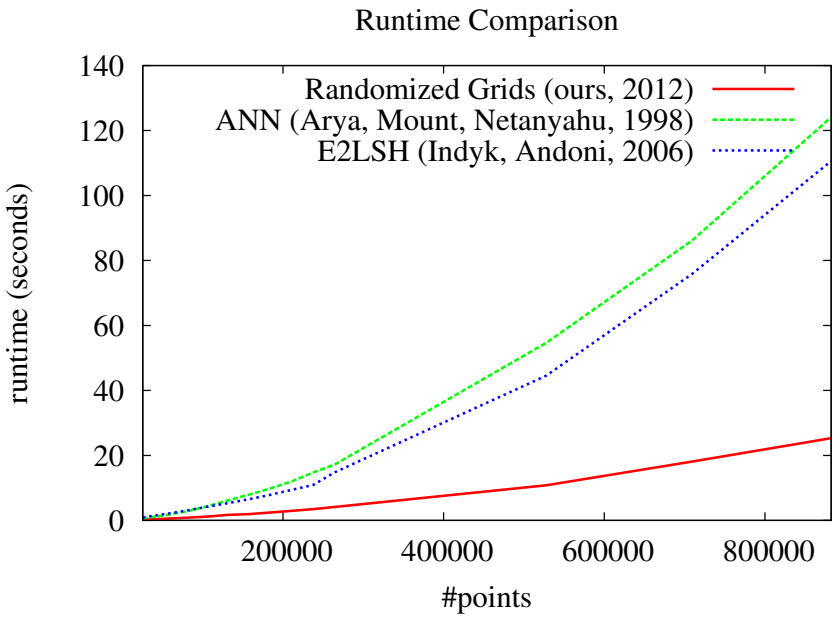

Figure 3: Running times on inputs of increasing sizes.

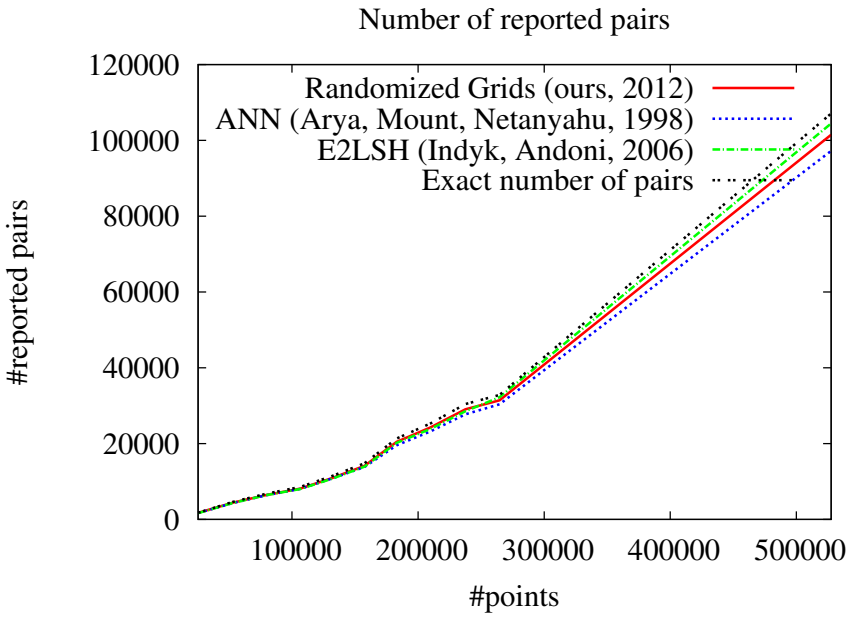

Figure 4: Number of reported neighboring pairs.

which we inspect.[5] But the gain in running time due to this decrease is subsumed by the loss incurred by the larger number of experiments which we need to perform. Figure 5 shows the running times on our data set produced from 100 images, of the algorithms with rotations and without rotations for various values of c. In all experiments we reported about 1700 pairs. The best running time is obtained for $c = 3.6$ without rotation. A peculiar phenomenon about this data set is that for higher values of c the algorithm with rotation becomes faster. This phenomenon is somewhat surprising and we do not fully understand it. It may be related to a special initial alignment of this particular point set. (Points in this data set seem to tend to be near one of the main diagonals of the unit cube.) For $c = 7$ without rotation we inspect 8528906 pairs out of which 1782 are at distance at most r. When we rotate the points we inspect only 1626949 pairs out of which 1713 are at distance at most r.

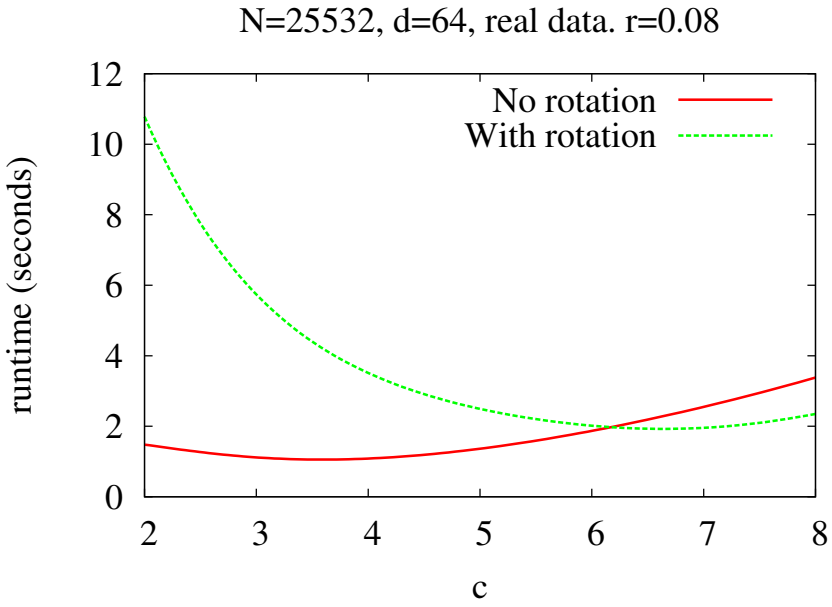

Figure 5: Running times with and without rotation on real data as a function of the cell size.

For other data sets containing points which are distributed differently, random rotations can give a substantial advantage. To demonstrate this we experimented with a random data set of 100000 points in the unit cube of dimension 16 (which do not have low doubling dimension). We set $r = 0.5$ for this data set. There are 3600 pairs at distance at most r in the data set of which we reported about 3500 in all our experiments. Here the number of inspected pairs increases more drastically with c.[6]. As a result it pays off to use rotations with a smaller value of c and a larger number of experiments. Figure 6 shows the running times for the algorithms with rotations and without rotations for different values of c. We see in this plot that the best running time is obtained for $c = 1.3$ with rotations. Recall that this is obtained despite the overhead of $O(d^2n)$ time which we invest in each experiment to

[5] For our data set produced from 100 images we inspect about 40, 240, 4700, and 90000 pairs for c equals to 0.8, 1, 2, and 3.6, respectively.

[6] We inspect about 80000 pairs for $c = 0.8$, 500000 pairs for $c = 1.0$, and 120000000 pairs for $c = 2.0$. For $c = 3.6$ the number of pairs is too large for the computation to finish in a reasonable amount of time.

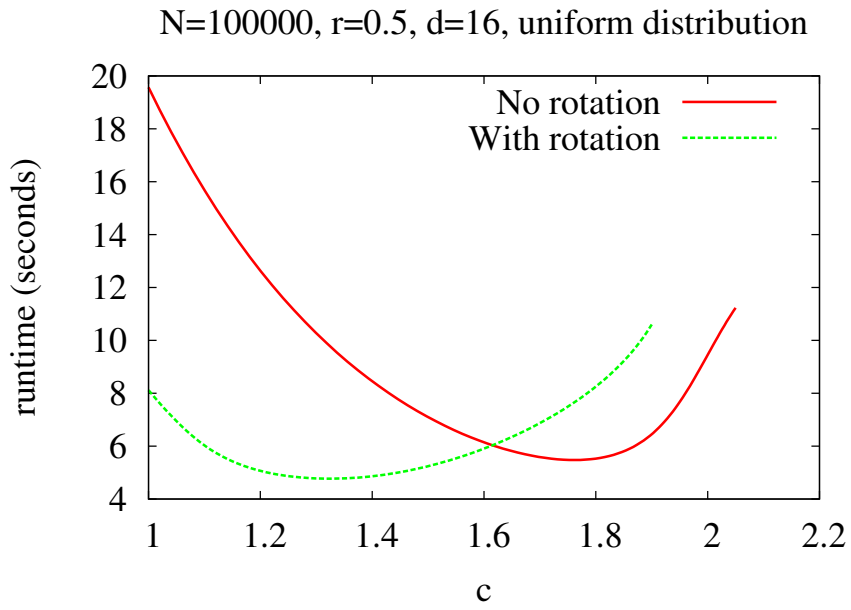

Figure 6: Running times with and without rotation on random data as a function of the cell size.

randomly rotate the points.

Acknowledgements. The authors would like to thank Shiri Artstein-Avidan for helpful discussions concerning Minkowski sums, and Sariel Har-Peled and Amir Rothschild for helpful discussions about the problem and its existing literature.

References

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, On the surprising behavior of distance metrics in high dimensional spaces, in *Proc. 8th Internat. Conf. Database Theory*, 2001, 420–434.

[2] A. Andoni, Implementation of LSH: E2LSH, `http://www.mit.edu/~andoni/LSH/`

[3] A. Andoni and P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, in *Proc. 47th Annu. IEEE Sympos. Found. Comput. Sci.*, 2006, 459–468.

[4] A. Andoni and P. Indyk, Efficient algorithms for substring near neighbor problem, in *Proc. 17th Annu. ACM-SIAM Sympos. Discrete Algorithms*, 2006, 1203–1212.

[5] A. Andoni and P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbors in high dimensions, *Commun. ACM* 51(1) (2008), 117–122.

[6] S. Arya, T. Malamatos, and D. M. Mount, Space-time tradeoffs for approximate nearest neighbor searching, *J. ACM*, 57(1) (2009), 1:1–1:54.

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

[8] P. Assouad, Plongements Lipschitziens dans R, *Bull. Soc. Math. France*, 111(4) (1983), 429–448.

[9] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, SURF: Speeded Up Robust Features, *Computer Vision and Image Understanding* 110(3) (2008), 346–359.

[10] S. N. Bespamyatnikh, Dynamic algorithms for approximate neighbor searching, in *Proc. 8th Canad. Conf. Comput. Geom.*, 1996, 252–257.

[11] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, When is "Nearest Neighbor" meaningful?, in *Proc. Internat. Conf. Database Theory*, 1999, 217–235.

[12] K. Böröczky, *Finite Packing and Covering*, Cambridge Tracts in Mathematics, Vol. 154, Cambridge University Press, Cambridge 2004.

[13] T. M. Chan, On enumerating and selecting distances, *Internat. J. Comput. Geom. Appls.* 11(3) (2001), 291–304.

[14] T. M. Chan, Closest-point problems simplified on the RAM, in *Proc. 13th Annu. ACM-SIAM Sympos. Discrete Algorithms*, 2002, 472–473.

[15] T. M. Chan, A minimalist's implementation of an approximate nearest neighbor algorithm in fixed dimensions. Manuscript. `www.cs.vwaterloo.ca/tmchan/sss.ps`.

[16] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. A. Plotkin, Approximating a finite metric by a small number of tree metrics, in *Proc. 39th Annu. Sympos. Found. Comput. Sci.*, 1998, 379–388.

[17] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in *Proc. 20th Annu. Sympos. Comput. Geom.*, 2004, 253–262. Also Chapter 3 in [28], pp. 61–72.

[18] L. Devroye and T. J. Wagner, Nearest neighbor methods in discrimination, in *Handbook of Statistics*, volume 2, P. R. Krishnaiah and L. N. Kanal, editors, North-Holland, Amsterdam 1982, 193–197.

[19] C. A. Duncan, M. T. Goodrich, and S. Kobourov, Balanced aspect ratio trees: Combining the advantages of k-d trees and octrees, *J. Algorithms* 38 (2001), 303–333.

[20] R. J. Gardner, *Geometric Tomography*, 2nd edition, Encyclopedia of Mathematics and its Applications, Vol 58, Cambridge University Press, New York, 2006.

[21] A. Genz, Methods for generating random orthogonal matrices, in *Monte Carlo and Quasi Monte Carlo Methods 1998*, H. Niederreite and J. Spanier (Eds.), Springer-Verlag, Berlin (1999), pp. 199–213.

[22] A. Gersho and R. M. Gray, *Vector Quantization and Data Compression*, Kluwer Academic Press, Boston, 1991.

[23] J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, Second Edition, CRC Press LLC, Boca Raton, FL, 2004.

[24] S. Har-Peled, *Geometric Approximation Algorithms*, Mathematical Surveys and Monographs, volume 173, AMS Press, Providence, RI, 2011.

[25] S. Har-Peled, A replacement for Voronoi diagrams of near linear size, in *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, 2001, 94–103.

[26] S. Har-Peled and M. Mendel, Fast construction of nets in low-dimensional metrics and their applications, *SIAM J. Comput.* 35(5) (2006), 1148–1184.

[27] A. Hinneburg, C. C. Aggarwal, and D. A. Keim, What is the nearest neighbor in high dimensional spaces?, in *Proc. 26th Internat. Conf. Very Large Data Bases*, 2000, 506–515.

[28] G. Shakhnarovish, T. Darrell, and P. Indyk, Eds., *Nearest-Neighbor Methods in Learning and Vision*, MIT Press, Cambridge MA, 2006.

[29] P. Indyk and J. Matoušek, Low-distortion embeddings of finite metric spaces, in [23, pp. 177–196].

[30] P. Indyk and R. Motwani, Approximate nearest neighbors: Towards removing the curse of dimensionality, in *Proc. 30th Annu. ACM Sypos. Theory Comput.*, 1998, 604–613.

[31] R. Krauthgamer and J. R. Lee, Navigating nets: Simple algorithms for proximity search, in *Proc. 15th Annu. ACM-SIAM Sympos. Discrete Algorithms*, 2004, 798–807.

[32] H.-P. Lenhof and M. Smid, Sequential and parallel algorithms for the k closest pairs problem, *Internat. J. Comput. Geom. Appls.* 5 (1995), 273–288.

[33] M. Muja and D. G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in *Internat. Conf. Computer Vision Theory Appls.* (VISAPP'09), 2009, 331–340

[34] M. Muja and D. G. Lowe, The FLANN implementation, `http://people.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN`

[35] M. E. Muller, A note on a method for generating points uniformly on n-dimensional spheres, *Commun. ACM* 2(4) (1959), 19–20.

[36] J. S. Salowe, Enumerating interdistances in space, *Internat. J. Comput. Geom. Appls.* 2 (1992), 49–59.

[37] R. Schneider, *Convex Bodies: The Brunn–Minkowski Theory*, Cambridge University Press, Cambridge, 1993.

[38] D. M. Y. Sommerville, *An Introduction to the Geometry of n Dimensions*, Dover, New York, 1958.

[39] G. T. Toussaint, Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining, *Internat. J. Comput. Geom. Appls.* 15 (2) (2005), 101–150.