Chapter 4

Incremental Constructions along Space-Filling Curves in Higher Dimensions

Summary. In this chapter we present an analysis of the incremental construction along space-filling curves of a Delaunay tessellation which applies to a wider class of distributions. We will focus on uniformly distributed points in higher dimensions. We also apply the analysis to normally distributed points in the plane. This analysis differs from the analysis for the two-dimensional uniform case in the following way: While the analysis given previously was mainly based on a refined analysis of the general straight line walk and of the structure of a Delaunay triangulation near the boundary, the analysis in this chapter focuses on the properties of the space-filling curve heuristic. It also applies to other traveling salesperson problem heuristics with similar properties. After the analysis we present experimental results on incremental constructions of Delaunay tessellations using space-filling curves. Furthermore, we analyze a different incremental construction of Delaunay tessellations using nearest neighbors which also runs in linear expected time on uniformly distributed points in higher dimensions.

4.1 Introduction

The main difficulty in the analysis in the previous chapter is the boundary analysis for the Delaunay tessellation. In the analysis of this chapter we count the number of intersections of a space-filling curve tour and a Delaunay tessellation differently which prevents a separate analysis for the boundary of the Delaunay tessellation. In brief, instead of counting for every walking step the expected number of points which may interfere with it, we count for every point the expected number of walking steps with which it may interfere. Thus, we analyze more the structure of the space-filling curve tour than the structure of the Delaunay tessellation. This makes the analysis easier in higher dimensions as well as near the boundary if the density of the point distribution is sufficiently smooth.

The analysis in this chapter is not a generalization of the analysis in the previous chapter. In the previous chapter we considered uniformly distributed points in a bounded convex region. If the region is not a square then the point distribution is not smooth in the bounding square used for the space-filling curve. Thus, the analysis in this chapter is not directly applicable to this case. Further, we do not analyze the case where points are inserted during the walk. In this chapter the analysis is applied to the case of uniformly distributed points in a d-cube and to normally distributed points in the plane but most of the analysis holds also for other point distributions.

4.2 Analysis

4.2.1 Counting Scheme

Setting. As in the analysis of the previous chapter we consider the following situation: Let $x_1, \dots, x_n$ and $y_1, \dots, y_m$ be points in $\mathbb{R}^d$. Assume we want to insert $y_1, \dots, y_m$ into the Delaunay tessellation $DT(x_1, \dots, x_n)$ of the points $x_1, \dots, x_n$. We insert $y_1, \dots, y_m$ along a space-filling curve order denoted by $T(y_1, \dots, y_m)$ which is given by a permutation $\pi\colon \{1, \dots, m\} \to \{1, \dots, m\}$. Let $f(x, DT(x_1, \dots, x_n))$ denote the number of d-dimensional faces incident to x in the Delaunay tessellation $DT(x_1, \dots, x_n)$, e.g., in the plane the number of triangles incident to x. Let $F(DT(x_1, \dots, x_n))$ denote the total number of d-dimensional faces of $DT(x_1, \dots, x_n)$. Let B_{y_i,y_j} denote the ball with the line segment (y_i, y_j) as a diameter. Furthermore, let

$$b(x, T(y_1, \dots, y_m)) := \sum_{i=1}^{m-1} \mathbf{1}_{B_{y_i,y_{i+1}}}(x),$$

i.e., the number of balls around tour segments in which x lies.

We denote the random variables corresponding to $x_1, \dots, x_n$ and $y_1, \dots, y_m$ as $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$, respectively.

We will analyze the case where new points are located first in the Delaunay tessellation of the previous round and from there using a history.

Counting Scheme. For points in general position the faces of the Delaunay tessellation intersected by tour segments are $(d-1)$-dimensional or d-dimensional with these two cases alternating along the tour segment. Of these, we will count the d-dimensional faces.

Let I be the number of intersections between d-simplices of the Delaunay tessellation and line segments of the space-filling curve tour. We split the number of intersections into $I = I_1 + I_2$ where

- I_1 is the number of intersections where the d-simplex is in conflict with one of the endpoints of the tour segment,

- I_2 is the number of intersections where the d-simplex is not in conflict with the endpoints of the tour segment.

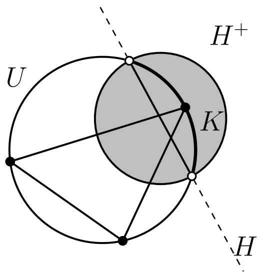


Figure 4.1: Illustration of the proof of Lemma 4.1.

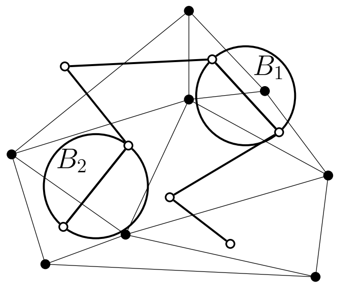


Figure 4.2: Balls along the space-filling curve tour.

We first prove the following lemma.

Lemma 4.1. *Let Δ be a d-simplex and s a line segment intersecting Δ. If the endpoints of s lie outside of the circumsphere of Δ then the ball with s as diameter contains a vertex of Δ.*

Proof. The proof is illustrated in Figure 4.1. Let U be the circumsphere of Δ and let K be the ball with s as a diameter. Without loss of generality we can assume that the endpoints of s lie on U. Otherwise we can shrink s by which we also shrink K. We want to show that K contains a vertex of Δ. We can assume that the radius of K is less than the radius of U. Otherwise we have $U = \partial K$ and all vertices of Δ lie in K.

Now consider the spherical cap $U \cap K$. Its boundary lies in a hyperplane H and the cap itself in a half-space H^+ defined by H. In particular,

$$U \cap H^+ = (U \cap K) \cap H^+ \subset K.$$

Since s intersects Δ there is a vertex x of Δ in H^+. For this vertex we have

$$x \in U \cap H^+ \subset K.$$

□

Consider a fixed line segment $(y_{\pi(i)}, y_{\pi(i+1)})$ on the space-filling curve tour $(1 \leq i \leq m)$. By Lemma 4.1 any d-face of the Delaunay tessellation intersecting this segment and not in conflict with one of the endpoints of the tour segment must have one vertex in the ball with the tour segment as diameter. Thus, for any intersection counted in I_2 the corresponding Delaunay simplex has a vertex in the ball with the corresponding tour segment as a diameter. We will use this to bound I_2.

Example. The counting scheme is illustrated in Figure 4.2: It shows a Delaunay tessellation and points along a space-filling curve. First consider the ball B_1. In the algorithm the Delaunay tessellation is traversed along the corresponding tour segment. During this, d-faces and $(d-1)$-faces are processed which are triangles and edges in the two-dimensional case. We are only interested in the d-faces, i.e., the triangles. There is one point in the ball B_1. Thus triangles at this point might contribute to I_2. The number of contributing triangles which are incident to this point can be bounded by the total number of triangles at this point.

The situation that a face does not contribute to I_2 because all its vertices lie outside the ball is illustrated in the figure by the ball B_2. The intersecting $(d-1)$-face has all its vertices outside the ball. This gives that the adjacent d-faces are in conflict with an endpoint of the tour segment and are therefore counted in I_1.

Bounding I_1. A Delaunay face in conflict with a vertex of the tour needs to be counted at most once for each tour segment adjacent to the vertex, i.e., at most twice for the vertex. By Theorem 3.9 we can bound the cost induced by these faces by a constant times the update cost, where the constant depends on the sampling parameter and the dimension. This, directly bounds the expected value of I_1 in terms of the expected update cost.

Remarks on I_1. Before bounding I_2 let us make two further remarks on I_1. First, in the case where points are inserted during the traversal the number of conflicting faces can be bounded directly by the update cost because the conflicting faces are destroyed when the endpoint of the tour segment is inserted. Depending on the update scheme it might actually be the case that the walk ends at the first d-dimensional conflicting face since the Bowyer-Watson update can start at any conflicting d-face and find the remaining conflicts from there.

Second, bounding I in terms of $I_1 + I_2$ can be applied to any traveling salesperson problem heuristic. In this case the bound on $\mathrm{E}[I_1]$ remains the same. More generally, the points can be located by using a tree on the points of the round and possibly on the points of the previous rounds. The points are located by traversing the tree. In this case, a conflict between a vertex of the tour and a simplex of the Delaunay tessellation might be counted as often as we visit the vertex during the traversal. Thus we can again bound $\mathrm{E}[I_1]$ in terms of the expected update cost but get the expected maximum degree of the tree as additional factor.

Bounding I_2. We bound I_2 by counting for each vertex of the Delaunay tessellation in a ball of a tour segment the total number of d-simplices at this vertex. That is,

$$\begin{aligned} I_2 &\leq \sum_{i=1}^{m-1}\sum_{j=1}^{n} \mathbf{1}_{B_{y_{\pi(i)},y_{\pi(i+1)}}}(x_j) f(x_j, DT(x_1,\dots,x_n)) \\ &= \sum_{j=1}^{n} b(x_j, T(y_1,\dots,y_m)) f(x_j, DT(x_1,\dots,x_n)), \end{aligned}$$

where $\mathbf{1}_A(x) = 1$ if $x \in A$ and $\mathbf{1}_A(x) = 0$ otherwise.

In the following proposition we bound $\mathrm{E}[I_2]$ for the case that the points are given by independent identically distributed random variables.

Proposition 4.2. *Let $X_1, \ldots, X_n, Y_1, \ldots, Y_m \in D \subseteq \mathbb{R}^d$ be independent identically distributed random variables which are in general position with probability 1. Let*

$$\begin{aligned} b_m &:= \sup_{x \in D} E[b(x, T(Y_1, \ldots, Y_m))], \\ F_n &:= E[F(DT(X_1, \ldots, X_n))]. \end{aligned}$$

Then $E[I_2] \leq (d+1) b_m F_n$.

Proof. In the following we abbreviate $T(Y_1, \ldots, Y_m)$ by T and $DT(X_1, \ldots, X_n)$ by DT. We have

$$\begin{aligned} \mathrm{E}[I_2] &\leq \mathrm{E}\left[\sum_{i=1}^{n} b(X_i, T) \cdot f(X_i, DT)\right] \\ &\leq n\,\mathrm{E}[b(X_1, T) \cdot f(X_1, DT)] \\ &\leq n\,\mathrm{E}[\mathrm{E}[b(X_1, T) \mid X_1] \cdot f(X_1, DT)]. \end{aligned}$$

Since X_1 is independent of $Y_1, \ldots, Y_m$, we have

$$\mathrm{E}[b(X_1, T) \mid X_1 = x_1] = \mathrm{E}[b(x_1, T)].$$

Since for all x_1

$$\mathrm{E}[b(x_1, T)] \leq \sup_{x \in D} \mathrm{E}[b(x, T(Y_1, \ldots, Y_m))] = b_m$$

we have

$$\mathrm{E}[b(X_1, T) \mid X_1 = x_1] \leq b_m.$$

Thus,

$$\begin{aligned} \mathrm{E}[I_2] &\leq b_m n \mathrm{E}[f(X_1, DT)] \\ &= b_m (d+1) \mathrm{E}[F(DT)] \\ &= (d+1) b_m F_n \end{aligned}$$

since $n\mathrm{E}[f(X_1, DT)] = \mathrm{E}[\sum_{i=1}^{n} f(X_i, DT)]$ counts each face once for each of its vertices, i.e., $(d+1)$ times. □

The previous proposition bounds $\mathrm{E}[I_2]$. Like the bound for $\mathrm{E}[I_1]$, it also holds for other traveling salesperson problem heuristics and tree traversals. Furthermore, it can be generalized to the case where the points do not come from the same distribution. In this case, we have as bound for $X_i \in D_i$ $(1 \leq i \leq n)$

$$\sup_{x \in \bigcup_{1 \leq i \leq n} D_i} \mathrm{E}[b(x, T(Y_1, \ldots, Y_m))]\, F_n.$$

In particular, the bound also holds for a non-random point set, although it does not yield an interesting bound in this case.

Let B_T be a ball chosen uniformly at random from the balls along the tour. To prove $\mathrm{E}[I_2] \in O(n)$ it suffices to prove that for all $x \in D$

$$\mathrm{P}[x \in B_T] \in O\left(\frac{1}{m}\right),$$

where D is the domain from which the X_i are drawn ($1 \leq i \leq n$). Now, $\mathrm{P}[x \in B_T]$ does not depend on properties of the Delaunay tessellation, thus we have reduced the problem to a problem on properties of the tour.

4.2.2 Analysis for Space-Filling Curve Orders

As noted before the analysis up to here also applies to other tour heuristics. To bound $\mathrm{P}[x \in B_T]$ we will now use the Hölder-continuity of space-filling curves. Therefore, the following analysis will only hold for space-filling curve orders.

The following argument suggests that $\mathrm{P}[x \in B_T] \in O(1/m)$ for space-filling curve orders if the point distribution considered is "sufficiently" smooth: if $\psi : [0,1] \to [0,1]^d$ is the space-filling curve used then the probability that the preimage $\psi^*(x)$ of a point x falls into the interval $[\psi^*(Y_{\pi(i)}), \psi^*(Y_{\pi(i+1)})]$ between two points defining a ball is $1/(m-1)$. Thus the probability that a point lies in the image of the interval under ψ is again $1/(m-1)$ and this image covers a similar area as the ball. This notion of "similar area" is difficult to capture and we will argue directly using the probability distribution over the space from which the points are drawn.

We want to bound $\mathrm{P}[x \in B_T]$. First it is important to consider how the space-filling curve was computed. If the points come from a certain region we can simply compute the space-filling curve based on a subdivision of this region. But for points from an unbounded region, like in the case of the normal distribution, the bounding cube for the space-filling curve depends on the actual points. For simplicity we will assume that the bounding cube is chosen as $[-u,u]^d$ where u is the largest occurring coordinate, i.e., the largest L_∞-norm of a point. For a space-filling curve $psi\colon [0,1] \to [0,1]^d$ we denote by $\hat{\psi}\colon [0,1] \to [-u,u]^d$ the scaled space-filling curve. The mapping $\hat{\psi}$ is Hölder continuous with exponent $1/d$ and Hölder constant $c_{\hat{\psi}} = 2u \cdot c_\psi$, i.e., for $t_1, t_2 \in [0,1]$ we have $\left\|\hat{\psi}(t_1) - \hat{\psi}(t_2)\right\| \leq c_{\hat{\psi}} \|t_1 - t_2\|^{1/d}$. We denote by $\hat{\psi}^*\colon [-u,u]^d \to [0,1]$ the selection of preimages according to ψ^*.

The following lemma provides a bound on the length of a tour edge in this setting.

Lemma 4.3. *Let $Y_1, \ldots, Y_m$ be independent identically distributed random variables in $\mathbb{R}^d$ with Lebesgue density function g_{Y_1}. Let $\psi : [0,1] \to [0,1]^d$ be a Hölder continuous and bi-measure preserving space-filling curve with Hölder constant c_ψ. Let L be a random tour segment of a space-filling curve tour through $Y_1, \ldots, Y_m$ based on $\hat{\psi}$. Then for all $\ell > 0$*

$$\begin{aligned} &P[|L| > \ell] \\ &\quad \leq \int_{\mathbb{R}^d} g_{Y_1}(y) \left(1 - \frac{d}{c_\psi^d} \int_{[0,\ell]} s^{d-1} \min\left\{g_{Y_1}(y') \mid \|y - y'\| < s\right\} ds\right)^{m-1} d\lambda^d(y). \end{aligned}$$

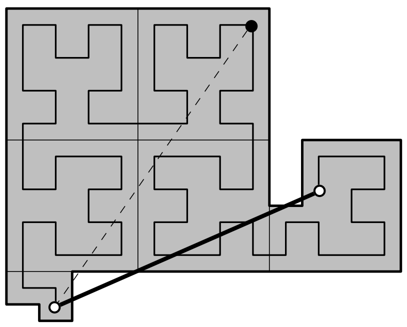


Figure 4.3: The length ℓ of a tour segment implies an empty area of size at least of order ℓ^d close to the segment.

Proof. Let Y (from $Y_1, \dots, Y_m$) be the first point of the random segment L. To bound $\mathrm{P}[|L| > \ell]$ we first consider $\mathrm{P}[|L| > \ell \,|\, Y = y]$. For this we use the Hölder continuity and bi-measure preserving property of the space-filling curve. The underlying idea is illustrated in Figure 4.3: A tour segment (solid line) of length ℓ yields that the empty region (grey area) has a volume of order at least ℓ^d. In turn, an area of ℓ'^d between two points (e.g., the end points of the dashed line) yields that the distance between the points is at most of order ℓ'.

In the case that $|L| > \ell'$ for an $\ell' > 0$ we get

$$\left\|\hat{\psi}^*(Y) - \hat{\psi}^*(Y')\right\| \;\geq\; \left(\frac{\ell'}{c_{\hat{\psi}}}\right)^d,$$

i.e., a tour segment longer than ℓ' implies an interval of length at least $(\ell'/c_{\hat{\psi}})^d$ in which no other preimage lies. By the bimeasure-preserving property of the space-filling curve we can consider the probabilities over [0,1]. Because of the scaling, a density of λ on the side of the points in $[-u, u]^d$ corresponds to a density of $(2u)^d\lambda = (c_{\hat{\psi}}/c_\psi)^d\lambda$ on the side of the preimages in $[0, 1]$.

If the measure corresponding to an interval is a then the probability for one point to be outside of the interval is $1 - a$ and for all $m - 1$ remaining points is $(1 - a)^{m-1}$.

Using the bimeasure-preserving property, now in the other direction, we can bound the change of the distribution for $t_0, t_0 + t \in [0, 1]$ by

$$g_{Y_1}(\hat{\psi}(t_0 + t)) \;\;\geq\;\; \min\left\{g_{Y_1}(\hat{\psi}(t')) \,|\, \left\|\hat{\psi}(t_0) - \hat{\psi}(t')\right\| \leq c_{\hat{\psi}} t^{1/d}\right\}.$$

This yields that

$$a \;\;\geq\;\; \int_{[0,(\ell/c_{\hat{\psi}} t)^d]} \left(\frac{c_{\hat{\psi}}}{c_\psi}\right)^d \min\left\{g_{Y_1}(y') \,|\, \|y' - y\| \leq c_{\hat{\psi}} t^{\frac{1}{d}}\right\} d\lambda(t).$$

Substituting the integration variable t by $s = c_{\hat{\psi}} \cdot t^{1/d}$ we get

$$a \;\;\geq\;\; \frac{d}{c_\psi^d} \int_{[0,\ell]} s^{d-1} \min\left\{g_{Y_1}(y') \,|\, \|y' - y\| \leq s\right\} d\lambda(s).$$

Thus we have a bound on a not depending on u. Now

$$\mathrm{P}\left[|L| > \ell \,|\, Y = y\right] \leq \left(1 - \frac{d}{c_\psi^d} \int_{[0,\ell]} s^{d-1} \min\left\{g_{Y_1}(y') \mid \left\|y' - y\right\| \leq s\right\} d\lambda(s)\right)^{m-1}$$

from which the claim of the lemma follows.

□

Using Lemma 4.3 we can bound $\mathrm{P}\left[x \in B_T\right]$ by

$$\int_{\mathbb{R}^d} g_{Y_1}(y) \left(1 - \frac{d}{c_\psi^d} \int_{[0,\|x-y\|]} s^{d-1} \min\left\{g_{Y_1}(y') \mid \left\|y - y'\right\| < s\right\} ds\right)^{m-1} d\lambda^d(y).$$

For the bound of the lemma to be useful, the density of Y should not vary too much over a small region. For instance, if the points come from a bounded convex region and the space-filling curve is defined on a bounding cube of the region then the density drops to zero when the space-filling curve leaves the region, i.e., the tour “jumps”. In this case we would need to handle the jumps separately.

4.3 Uniformly Distributed Points

So far, no assumption on the distribution of the points was made beyond being independent identically distributed. We will now use Lemma 4.3 for the multi-dimensional uniform distribution in the d-cube. We expect that the analysis of this chapter can be generalized to convex regions by further analyzing the boundary case.

Theorem 4.4. *The* incremental construction along space-filling curves *(using the history of a round) computes the Delaunay tessellation of points drawn independently and uniformly from a d-cube in linear expected time.*

Proof. We again analyze the cost of the last round. Assume $X_1, \ldots, X_n$ are the points of the previous round and $Y_1, \ldots, Y_m$ the points of the last round. Further assume that these points are drawn independently and uniformly from the unit d-cube. By Proposition 4.2 it suffices to prove for all $x \in [0,1]^d$ that the expected number of balls corresponding to a space-filling curve tour through $Y_1, \ldots, Y_m$ and containing x is constant.

Let L be the tour segment starting at Y, where Y is chosen uniformly at random from $Y_1, \ldots, Y_m$. Let B_T be the corresponding ball. If $x \in B_T$ then $|L| > \|x - Y\|$, thus by Lemma 4.3

$$\begin{aligned}
\mathrm{P}\left[x \in B_T\right] &\leq \mathrm{P}\left[|L| > \|x - Y\|\right] \\
&\leq \int_{[0,1]^d} \left[1 - \frac{d}{c_\psi^d} \int_{[0,\|x-y\|]} s^{d-1} ds\right]^{m-1} d\lambda^d(y) \\
&\leq \int_{[0,1]^d} \left[1 - \left(\frac{\|x-y\|}{c_\psi}\right)^d\right]^{m-1} d\lambda^d(y).
\end{aligned}$$

Let ω_d denote the surface area of the d-sphere and κ_d the volume of the d-ball. Using polar/spherical coordinates we get

$$\begin{aligned}
\mathrm{P}\left[x \in B_T\right] &\le \int_{\mathbb{R}^d} \left[1 - \left(\frac{x}{c_\psi}\right)^d\right]^{m-1} d\lambda^d(y) \\
&= \omega_{d-1} \int_{[0,\infty)} r^{d-1} \left[1 - \left(\frac{r}{c_\psi}\right)^d\right]^{m-1} d\lambda(r) \\
&= -\omega_{d-1} \frac{c_\psi^d}{m} \left[1 - \left(\frac{r}{c_\psi}\right)^d\right]^{m} \Bigg|_{r=0}^{\infty} \\
&\le \kappa_d \frac{c_\psi^d}{m}.
\end{aligned}$$

Thus the expected number of balls containing x is bounded by

$$\kappa_d c_\psi^d,$$

i.e., the volume of a d-ball of radius c_ψ. □

4.4 Normally Distributed Points

Now we use Lemma 4.3 to show that the incremental construction algorithm runs in linear expected time for normally distributed points in the plane. In the following we will handle the boundary case by assuming an additional point location data structure as a fallback.

Theorem 4.5. *The* incremental construction along space-filling curves *(using the history of a round and an* $O(\log n)$ *point location data structure) computes the Delaunay tessellation of points drawn independent identically normally distributed in the plane in linear expected time.*

Proof. As in the proof of Theorem 4.4 for uniformly distributed points, it suffices by Proposition 4.2 to bound $\mathrm{P}\left[x \in B_T\right]$ for a point $x \in \mathbb{R}^2$ and B_T the ball around a random tour segment L with starting point Y. Again we bound $\mathrm{P}\left[x \in B_T\right] \le \mathrm{P}\left[|L| > \|x - Y\|\right]$ using Lemma 4.3.

Without loss of generality we assume that the points are independently and normally distributed centered at the origin and with the identity matrix as covariance matrix. The density function of a point is

$$g(y) = \frac{1}{2\pi} e^{-\frac{1}{2}\|y\|^2}.$$

For a point Y_i with $1 \le i \le m$ the probability that it is farther than a distance r_0 from the origin is

$$\mathrm{P}\left[\|Y_i\| > r_0\right] = e^{-\frac{1}{2}r_0^2}.$$

Therefore, the probability that at least one of m points is farther away than r_0 from the origin is bounded from above by $me^{-\frac{1}{2}r_0^2}$. Thus, for

$$r_0 := 4\sqrt{\log m}$$

the probability that one of the points on the tour is farther than r_0 from the origin is bounded from above by

$$me^{-2\log m} = \frac{1}{m}.$$

The points farther away therefore contribute at most $\frac{1}{m}$ to $\mathrm{P}[x \in B_T]$ and can be ignored in the following.

It remains to bound the probability $\mathrm{P}[x \in B_T, \|Y\| < r_0]$, i.e., the case that the starting point of the segment is closer than r_0 to the origin. We first bound $\mathrm{P}[|L| > \ell_0, \|Y\| < r_0]$ for $\ell_0 := \frac{1}{\sqrt{\log m}}$. Using Lemma 4.3 and spherical coordinates we have

$$\begin{aligned} &\mathrm{P}[|L| > \ell_0, \|Y\| < r_0] \\ &\quad\leq \int_{[0,r_0]} re^{-\frac{1}{2}r^2}\left(1 - \frac{2}{c_\psi^2}\int_{[0,\ell_0]} se^{-\frac{1}{2}(r+s)^2}ds\right)^{m-1} dr. \end{aligned} \tag{4.1}$$

For $r < r_0$ we have $(r+\ell_0)^2 = r^2 + 2\ell_0 r + \ell_0^2 \leq r^2 + 8 + \frac{1}{\log m}$, and therefore

$$e^{-\frac{1}{2}(r+\ell_0)^2} \geq e^{-\frac{1}{2}r^2}c_1 \quad \text{for } c_1 = e^{-9}.$$

This yields

$$\begin{aligned} \left(1 - \frac{2}{c_\psi^2}\int_0^{\ell_0} se^{-\frac{1}{2}(r+s)^2}ds\right)^{m-1} &\leq \left(1 - \frac{2}{c_\psi^2}c_1 e^{-\frac{1}{2}r^2}\int_0^{\ell_0} sds\right)^{m-1} \\ &= \left(1 - \frac{c_1}{c_\psi^2}e^{-\frac{1}{2}r^2}\ell_0^2\right)^{m-1}. \end{aligned}$$

Inserting this bound into the bound (4.1) on $\mathrm{P}[|L| > \ell_0, \|Y\| < r_0]$ yields

$$\mathrm{P}[|L| > \ell_0, \|Y\| < r_0] \leq \int_{[0,r_0]} re^{-\frac{1}{2}r^2}\left(1 - \frac{c_1}{c_\psi^2}e^{-\frac{1}{2}r^2}\ell_0^2\right)^{m-1} dr.$$

We have

$$\frac{c_1}{c_\psi^2}m\ell_0^2 \cdot re^{-\frac{1}{2}r^2}\left(1 - \frac{c_1}{c_\psi^2}e^{-\frac{1}{2}r^2}\ell_0^2\right)^{m-1} = \frac{d}{dr}\left(1 - \frac{c_1}{c_\psi^2}e^{-\frac{1}{2}r^2}\ell_o^2\right)$$

and therefore

$$\begin{aligned} \mathrm{P}[|L| > \ell_0, \|Y\| < r_0] &\leq \frac{c_\psi^2}{c_1 m\ell_0^2}\left[\left(1 - \frac{c_1}{c_\psi^2}e^{-\frac{1}{2}r^2}\ell_0^2\right)^m\right]_0^{r_0} \\ &\leq \frac{c_\psi^2}{c_1 m\ell_0^2} = \frac{c_\psi^2}{c_1}\frac{\log m}{m}. \end{aligned}$$

Thus there are an expected number of $O(\log m)$ points for which we do not have a bound on $|L|$. As used for Proposition 3.16 we can handle these points by an additional point location data structure like Kirkpatrick's point location hierarchy. This avoids the boundary analysis for the space-filling curve order. The cost induced by the expected $O(\log m)$ points is in $O(n+\log m \log n)$ where the linear part comes from building the point location data structure and the remaining part from the point location queries.

Finally we bound $\mathrm{P}\left[|L| > \|x-Y\|, |L| < \ell_0, \|Y\| < r_0\right]$ for $x \in \mathbb{R}^2$. Since Y is distributed spherically symmetric we may assume that x lies on the positive x-axis, i.e., $x = (\chi, 0)$ with $\chi \leq 0$. Furthermore, the probability is 0 if $\chi > l_0 + r_0$, so we can assume

$$0 \leq \chi \leq l_0 + r_0.$$

Let $B(z, r)$ denote the ball of radius r around z. We have so far

$$\begin{aligned}
&\mathrm{P}\left[|L| > \|x-Y\|, |L| < \ell_0, \|Y\| < r_0\right] \\
\leq\ & \frac{1}{2\pi} \int_{B(x,l_0)} e^{-\frac{1}{2}\|y\|^2} \left(1 - \frac{2c_1}{c_\psi^2}\|x-y\|^2 e^{-\frac{1}{2}\|y\|^2}\right)^{m-1} d\lambda(y) \\
=\ & \frac{1}{2\pi} \int_{B(0,l_0)} e^{-\frac{1}{2}\|x-y'\|^2} \left(1 - \frac{2c_1}{c_\psi^2}\|y'\|^2 e^{-\frac{1}{2}\|x-y'\|^2}\right)^{m-1} d\lambda(y'). \qquad (4.2)
\end{aligned}$$

Now,

$$\begin{aligned}
\|x-y'\|^2 &\geq (\chi - l_0)^2 = \chi^2 + l_0^2 - 2\chi l_0 \\
&\geq \chi^2 + l_0^2 - 2(r_0 + l_0) l_0 \\
&\geq \chi^2 - 8
\end{aligned}$$

by the definition of r_0 and l_0, and analogously

$$\begin{aligned}
\|x-y'\|^2 &\leq (\chi + l_0)^2 = \chi^2 + l_0^2 + 2(r_0 + l_0) l_0 \\
&\leq \chi^2 + 8 + 2l_0^2.
\end{aligned}$$

Inserting these bounds on $\|x-y'\|^2$ in the bound (4.2) we obtain

$$\begin{aligned}
&\mathrm{P}\left[|L| > \|x-Y\|, |L| < \ell_0, \|Y\| < r_0\right] \\
\leq\ & \frac{1}{2\pi} \int_{B(0,l_0)} e^{8/2} e^{-\frac{1}{2}\chi^2} \left(1 - \frac{2c_1}{c_\psi^2}\|y'\|^2 e^{-9/2} e^{-\frac{1}{2}\chi^2}\right)^{m-1} d\lambda(y') \\
=\ & e^4 e^{-\frac{1}{2}\chi^2} \int_0^{l_0} r \left(1 - \frac{2c_1}{c_\psi^2} r^2 e^{-9/2} e^{-\frac{1}{2}\chi^2}\right)^{m-1} dr \\
=\ & -e^{17/2} \frac{c_\psi^2}{2c_1} \frac{1}{m} \left[\left(1 - \frac{2c_1}{c_\psi^2} r^2 e^{-9/2} e^{-\frac{1}{2}\chi^2}\right)^{m}\right]_{r=0}^{l_0} \\
\leq\ & e^{17/2} \frac{c_\psi^2}{2c_1} \frac{1}{m}
\end{aligned}$$

which proves the theorem.

$\square$

4.5 Experiments

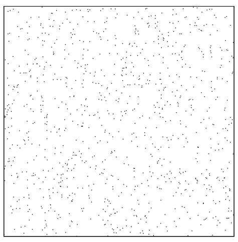
Related Experiments. The analyses in this and the previous chapter apply to several algorithms which have been thoroughly evaluated experimentally [3, 95, 153]. Before presenting our own experiments, we consider these. Besides these algorithms, there is a new CGAL* package [42] providing spatial orderings similar to space-filling curve orders which we describe as well.

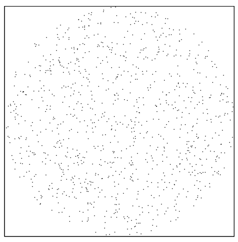
Amenta, Choi, and Rote [3] test incremental constructions con BRIO on surface data focussing on the performance in the case that the data structures are stored in virtual memory. As insertion order in a round they use oct-trees and kd-trees, with better results for oct-trees. An oct-tree traversal results in a space-filling curve order, however, the standard oct-tree traversal results in a Lebesque order which we have not considered since the Lebesque curve is neither Hölder continuous nor bi-measure preserving. As sampling parameter only 2 is considered. For instance Figure 6 in their paper shows the running time where points are located by walking in the Delaunay tessellation of the previously inserted points. This corresponds to incremental constructions along space-filling curves. The graph in their figure indicates a linear running time.

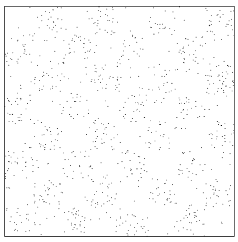
In their program *tess*3, Liu and Snoeyink [95] sample points to rounds, sort them along a Hilbert curve, and locate the points by walking along the Hilbert curve. The sampling is done according to the least significant bits. Assuming that these bits are sufficiently random, this corresponds to sampling with a sampling parameter of 2^b, where b is the length of the bit pattern used. The program *tess*3 performs a zigzag walk to find a sphere containing the new point. The algorithm is tested on uniformly distributed points in a cube and on protein data. It is compared to four popular codes for constructing the Delaunay tessellation in three dimensions. On both types of data, their algorithm performs best and has linear running time in the experiments (see Figures 3 and 4 in their paper). Additionally to the overall running time, for uniformly distributed points the number of created spheres/tetrahedra is counted and the running times for the following costs are measured: creating sphere equations, in-sphere tests, update, and point location. In their experiments, computing the Hilbert order takes 3.45% of the running time. Point location along the Hilbert curve takes only 0.43% of the running time.

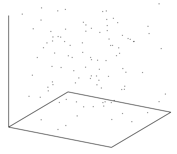
Zhou and Jones [153] construct Delaunay tessellations in two dimensions by sampling points to rounds, sorting points in a space-filling curve like order, and inserting points by walking along this order. They test their algorithm on uniform points, points on a parabola, and terrain data and observe that the running time of their algorithm is effectively linear. The algorithm is compared to insertion along a Lebesgue order (corresponding to the implementation of Amenta et al. of con BRIO), divide & conquer algorithms, jump & walk, and simple walking. Using the same data structures and primitives for the different algorithms, their algorithm performs best. The costs measured are the running times and the number of edge flips, in-circle tests, and CCW tests (i.e., orientation tests during the walk). The algorithm is tested for various sampling ratios.

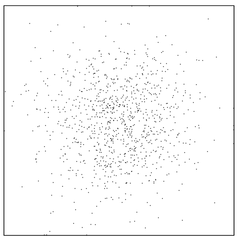
*Computational Geometry Algorithms Library, `http://www.cgal.org/`

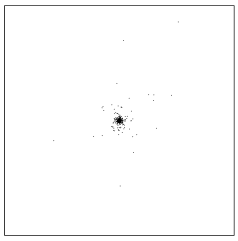

(a) uniform in square

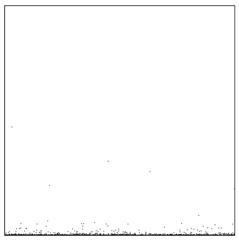

(b) uniform in disk


(c) chessboard


(d) uniform in cube


(e) normal


(f) Kuzmin


(g) line singularity

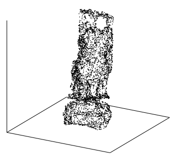

(h) Buddha

Figure 4.4: Point distributions depicted with (a)–(g) 100 and (h) 5000 points.

The number of edge flips and in-circle tests are minimized for sampling parameter α close to one. The CCW tests are minimized for α about 20 and their number drastically increases for $\alpha \to 1$. The number of flips and of in-circle tests are minimized for α close to 1. In their experiments, α about 10 balances the different costs best.

Delage [42] developed a CGAL package, *spatial sorting*, providing an order similar to a space-filling curve order and an assignment to rounds (without sampling). Pre-processing points with the provided order, significantly speeds up the construction of the Delaunay tessellation. Experimental results have not been published yet. The sorting is based on the Hilbert curve but splits the points at the medians of the coordinates. If only a small number of points remain, these are left unsorted.

Setting. The experiments described above yield an extensive evaluation of incremental constructions of Delaunay tessellations along space-filling curves. Nonetheless, we present our own experiments here. In particular, we are interested in the dependence of the running time of the algorithm on the point distribution and on the sampling parameter.

The algorithm is straightforward to implement using available software. Both CGAL [17] and Shewchuk's *Triangle*[†] [131] include an incremental construction algorithm using walking. For the experiments in this section we used CGAL.

We considered several uniform and non-uniform distributions which are depicted in Figure 4.5(a)–(g). Furthermore, we tested our algorithm for points on a surface, shown in Figure 4.5(h).

[†]Triangle, `http://www-2.cs.cmu.edu/~quake/triangle.html`

In two dimensions we tested the algorithm on two uniform distributions, points distributed uniformly in a square and point distributed uniformly in a disk. The analysis for uniformly distributed points directly generalizes to the case of an (β, γ)-measure over a bounded convex region C, i.e., a probability measure for which the density function $f(x)$ is bounded for all $x \in C$ between β and γ for fixed $(0 < \beta \leq \gamma)$. As an example of such a distribution we take points distributed over a chessboard where the density function $f(x)$ is $1/8$ on white squares and $7/8$ on black squares.

Further two-dimensional distributions we considered in our experiments are the standard normal distribution, the Kuzmin distribution, and the line singularity distribution. The Kuzmin distribution can be used to model clusters. It is a radially symmetric distribution where the distance r to the center has cumulative distribution function $M(r) = 1 - (1 + r^2)^{-1/2}$. For the line singularity distribution, points are very densely distributed along a line segment. It can be defined by a transformation from the uniform distribution: For $b \geq 0$ and $u, v \in [0, 1]$ independent uniform random variables, the transformation is $(x, y) = (u, b/(v - bv + b))$; we chose $b = 0.001$. Blelloch et al. [16] suggest these distributions as non-uniform representatives of real world problems and to defy techniques relying on uniform distributions.

The algorithm runs in higher dimensions and we have analyzed it for uniformly distributed points in a cube (using an additional point location data structure). In three dimensions we consider in the experiments uniformly distributed points drawn from a cube and points drawn from a surface.

In the experiments [3, 95, 153] discussed above, incremental constructions along space-filling curves have been extensively compared to other algorithms, and we will limit the comparison to one worst-case optimal algorithm, the *Delaunay Hierarchy* [46] which is included in the CGAL library.

We test incremental constructions along space-filling curves for different sampling parameters $\alpha > 1$. We denote our algorithm with a sampling parameter of α by SFC(α). For $\alpha \to 1$ our algorithm corresponds to the randomized construction, i.e., to simple walking if no further point location data structure is used. For $\alpha \to \infty$ the algorithm corresponds to an incremental construction along a space-filling curve without sampling. We denote this by SFC(∞).

All time measurements are done on a Intel(R) Pentium(R) 4 CPU 3.00GHz with 2.048KB cache size. We used CGAL 3.2.1 and the g++ 3.3.5 compiler with options −O2 and −NDEBUG.

Experimental Results. We split the running time into the time needed to construct the space-filling curve order and the time for the incremental construction. Table 4.1 and Table 4.2[‡] show for the different distributions the corresponding running times on 1 000 000 points in two dimensions and 500 000 uniform points in three dimensions and 543 652 points from the Happy Buddha data set[§].

[‡]The improved running times in comparison to our previous experiments in [22] result from compiling with −NDEBUG.

[§]Stanford 3D scanning repository, `http://www-graphics.stanford.edu/data/3Dscanrep`

Distribution	SFC(2)	SFC(4)	SFC(8)	SFC(16)	SFC(∞)
square	0.3336	0.3464	0.3659	0.3711	0.3813
disk	0.3379	0.3643	0.3876	0.3982	0.4118
chessboard	0.3809	0.3951	0.4398	0.4433	0.4608
normal	0.6166	0.9044	1.1554	1.2224	1.3570
Kuzmin	0.7550	0.6775	0.8144	0.7667	0.6238
line	0.8693	0.7888	0.6740	0.7117	0.6239
cube	0.2686	0.2982	0.2859	0.3347	0.3041
Buddha	0.4433	0.4404	0.4506	0.4903	0.4432

Table 4.1: Running times of SFC construction in seconds.

Distribution	SFC(2)	SFC(4)	SFC(8)	SFC(16)	SFC(∞)	Del. Hier.
square	3.1570	3.0798	3.0778	3.1342	3.8410	16.5918
disk	3.1910	3.0822	3.0770	3.1202	3.8386	16.4786
chessboard	3.1734	3.0874	3.0894	3.1242	3.9074	16.4686
normal	3.1846	3.2170	3.0954	3.1218	3.9235	16.7746
Kuzmin	3.2018	3.1178	3.1022	3.1302	3.7290	22.6614
line	3.1702	3.0798	3.1006	3.1358	3.8390	16.6742
cube	15.2210	15.1361	15.3974	15.7474	20.3061	32.6212
Buddha	17.3047	17.2695	17.8179	18.7144	31.9196	108.9810

Table 4.2: Running times of the incremental construction in seconds.

For all distributions in two dimensions, a sampling parameter of about 8 achieved best results. However the running time did not change much for different sampling parameters, as long as the sampling parameter was not close to 1. In three dimensions, a smaller sampling parameter of about 4 achieved best results. In this case for large sampling parameters, in particular for SFC(∞), the running time increased considerably.

For uniformly distributed points in two dimensions the dependency of the running time on the sampling parameter is shown in more detail in Figures 4.5 and 4.6. Figure 4.6 shows the running time for sampling parameters close to 1 while Figure 4.5 shows the running time for larger sampling parameters. If the sampling parameter tends to 1 the algorithm corresponds to simple walking and therefore Figure 4.6 shows the running time of walking as first value. This running time can be improved by sampling the starting point of the walk, i.e., using *jump & walk*. The running time we obtained for jump & walk was 167 seconds (compared to 251 seconds for walking). Both figures use non-linear scales for the x-axis, Figure 4.5 uses a logarithmic scale.

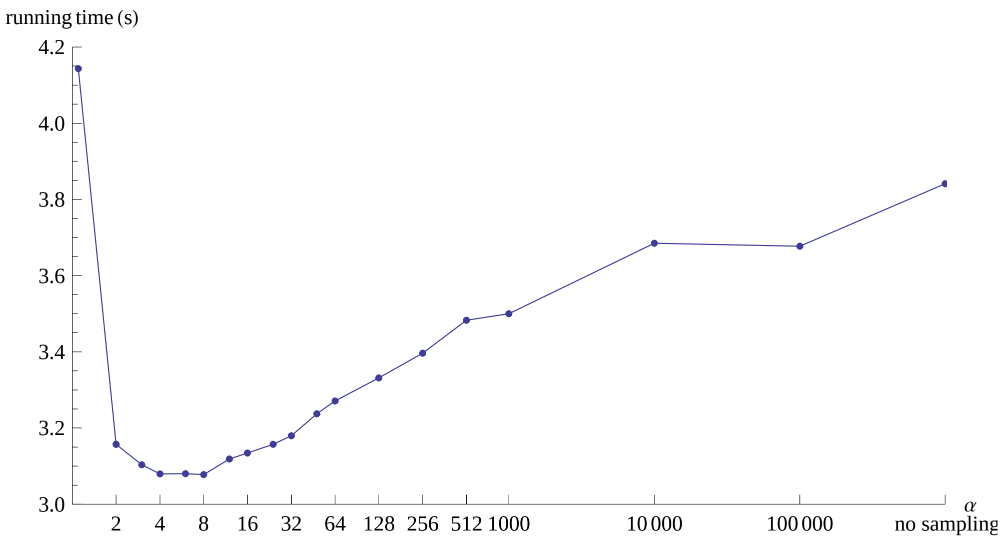


Figure 4.5: Running time for sampling parameter between 1 and ∞.

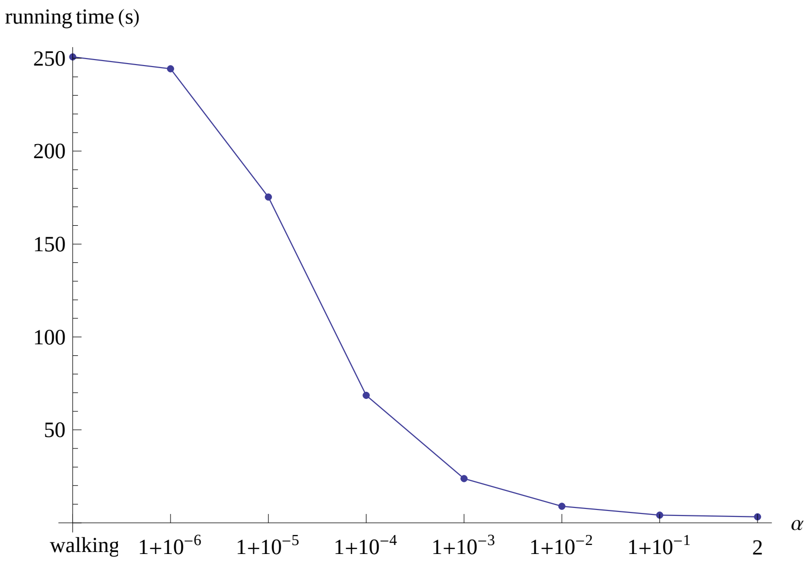


Figure 4.6: Running time for sampling parameter close to 1.

4.6 Constructing the Delaunay Tessellation using Nearest Neighbors

4.6.1 Introduction

A point can be located in a Delaunay tessellation by finding a nearby vertex of the tessellation and then walking from there. In the incremental construction using space-filling curves the nearby point was the previous point in the space-filling curve order. A randomized incremental construction using such a point location scheme has been proposed by Su and Drysdale [139]. For locating a new point a spiral search is performed on a grid data structure to find a point in the tessellation near to the new one. The spiral search starts with the cell containing the new point and then searches nearby cells. From this point a walk in the triangulation is used to find the triangle containing the new point. In this section we consider the performance of the algorithm on uniformly distributed points. We analyze a variant of the algorithm using a nearest neighbor as starting point, and present ideas for the analysis of the case that an *oriented walk* from a nearby point is used.

Su and Drysdale formulate the algorithm for points in the plane but it directly generalizes to higher dimensions. In the probabilistic analysis of the algorithm we will assume that the input points are drawn independently from a uniform distribution on a bounded convex region in the d-dimensional Euclidean space $\mathbb{R}^d$. For uniformly distributed points in a square Su and Drysdale showed that the grid data structure for the spiral search can be maintained in amortized constant time per point, and their proof directly generalizes to the case of a bounded convex region in $\mathbb{R}^d$. The spiral search in a convex bounded region in $\mathbb{R}^d$ needs expected constant time per point [12]. In our analysis we will concentrate on the expected running time of the point location starting from the point found by the spiral search.

The spiral search can be used to find the nearest neighbor in constant expected time but the number of cells examined can be reduced considerably (by a constant factor) if the neighbor encountered first is taken instead. Su and Drysdale therefore propose to start the walk at the first point found by the spiral search. Taking a nearest neighbor instead allows for a simpler analysis. Besides gaining further insight into the performance of the original algorithm the variant using nearest neighbors is of interest on its own because of the simple point location: walking is not necessary at all.

Besides stopping at the first point found, two further optimizations of the algorithm are the following: Only a single point per cell is stored since it is sufficient to know one point for any non-empty cell. Also the spiral search can be stopped after a certain number of cells (depending on n), and instead the walk is started at an arbitrary point of the tessellation.

For the nearest neighbor variant of the algorithm we prove that the expected time needed is linear in the number of points plus some extra cost bounded by the update cost. For the randomized incremental construction the expected update cost is linear in the complexity of the Delaunay tessellation. Overall this yields a linear expected time algorithm if the expected complexity of the

Delaunay tessellation is linear. This is the case in two dimensions. In higher dimensions it has been proved for points drawn independent, uniformly at random from a d-ball [57] and d-cubes [15] and it is assumed to be the case for uniformly distributed points from any “reasonably” smooth full dimensional bounded region [71].

Variants of the algorithm in two dimensions have been analyzed [49, 138]. However a subtle fact that has been overlooked is that a point close to a random query point is not a random point (i.e., uniformly drawn from the point set). Therefore its expected degree must be bounded differently than for a random point.

4.6.2 The Algorithm

The algorithm computes the Delaunay tessellation by a randomized incremental construction. The points are inserted one by one in a random order. In each insertion step the algorithm first finds the location of the new point in the present data structure and then updates the data structure including the point. The update can be done by flipping or by finding all conflicting simplices as discussed in Chapter 3.

Point location for a new point is done in two steps:

1. Find a nearby point using a dynamic grid data structure.

2. Find the new point by a walk starting at the nearby point.

If one finds a nearest neighbor in the first step, then the update can be started from there without needing to walk. It is still necessary to search for a conflicting simplex with the nearest neighbor as vertex.

Spiral Search on a Grid. Bentley, Weide and Yao [12] show how a variety of closest point problems can be solved efficiently for uniformly distributed points drawn from a bounded convex region in $\mathbb{R}^d$ using a grid. In particular they show how to perform *nearest neighbor searching* efficiently on uniformly distributed points: a set of points is assigned to a linear number of grid cells in linear time, such that for a new query point its nearest neighbor from the point set can be determined in expected constant time. For this, the grid cell of the new point is determined and the cells around it are searched for the nearest neighbor using a *spiral search*, i.e., the cells are searched ordered by distance. Ordering the cells by their L_∞-distance to the cell containing the query point simplifies the search. First the cell containing the new point is searched. If it is empty, the surrounding cells are searched according to some expanding pattern. In two dimensions this pattern can be a spiral as shown in Figure 4.7.

The first point found by the spiral search is not necessarily the nearest neighbor since some of the cells which have not yet been searched may contain closer points. To find the nearest neighbor the search must be continued until all cells have been searched for which the minimum distance to the query point is smaller than the distance between the point found first and the query point.

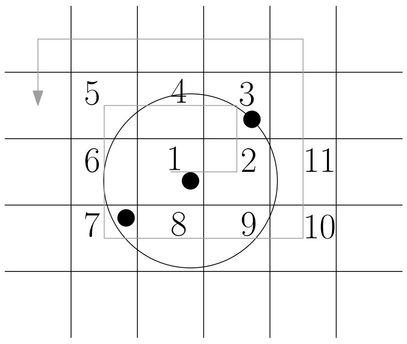


Figure 4.7: Finding a neighbor using spiral search.

In the figure the query point lies in cell 1 and the first point found lies in cell 3. In this case the search for the nearest neighbor can be stopped after cell 11.

Dynamic Grid Data Structure. To use spiral search in the construction of the Delaunay tessellation it is necessary to insert new points into the grid data structure. While this is not difficult, it has the effect that the number of grid cells and the number of points diverges. For this reason the grid data structure is recomputed each time the average number of points per cell exceeds $2^d c$ for a positive constant c. In this case each cell is subdivided into 2^d smaller cells and the points are reinserted. The overall amortized cost for maintaining the grid data structure is linear assuming linear time bucketing.

Walking Schemes. After finding a nearby point using spiral search it is necessary to find a simplex in conflict with the point, i.e., a simplex for which the circumsphere contains the point. This is done by walking in the Delaunay tessellation starting at the nearby point.

The variants of the algorithm differ in the walking scheme. The first variant using nearest neighbors does not need walking at all. It suffices to search the simplices at the nearest neighbor for a conflict with the new point. The tessellation is then updated using a Bowyer-Watson update (see Chapter 1) starting at a conflicting simplex. We analyze this variant in Section 4.6.3.

The second variant uses an *oriented walk*. This walking scheme is used by Su and Drysdale in their implementation. An oriented walk can be described as a traversal of simplices where one may pass through a $(d-1)$-dimensional face to a neighboring simplex if the new simplex and the query point are on the same side of the hyperplane containing the $(d-1)$-dimensional face. The walk ends as soon as a simplex traversed is in conflict with the new point or contains the point (in which case it is also in conflict with the point). An advantage of an oriented walk in comparison to a straight-line walk is that the orientation tests performed are simpler than the intersection tests in a straight-line walk.

4.6.3 Probabilistic Analysis of the Nearest Neighbor Variant

We now analyze the simplest variant of the algorithm where the spiral search is used to find a nearest neighbor of the query point and walking is not needed. The spiral search and the dynamic grid data structure have been analyzed by Bentley, Weide and Yao and by Su and Drysdale, respectively. It therefore suffices to bound the cost for finding a conflicting simplex from a nearest neighbor.

Theorem 4.6. *If spiral search is used to find a nearest neighbor, the algorithm constructs the Delaunay tessellation in linear expected time when the points are distributed independently and uniformly in a d-dimensional bounded convex region for which the expected complexity of the Delaunay tessellation is linear. In particular, this is the case in two-dimensions and for the unit d-ball and the unit d-cube.*

Proof. The expected time required for searching the nearest neighbor and for updating the tessellation is linear [12, 37]. It remains to bound the expected number of d-simplices of the tessellation containing the nearest neighbor of the point to be inserted.

The difficulty is that the nearest neighbor is not a random point of the tessellation. Still, a constant bound can be obtained. The analysis is similar to part of the analysis of the Delaunay hierarchy by Devillers [46].

Let $S := X \cup \{q\}$, where X is the set of points in the tessellation so far, and q the point to be inserted. The point q may be assumed to be a random point of S. Instead of considering the Delaunay tessellation of X, we may use the Delaunay tessellation of S: The difference in the complexity is bounded by the update cost.

Denote by $s(p)$ the number of d-simplices of the Delaunay tessellation containing p, by $n(p)$ the nearest neighbor of p in S, and by $\delta(p)$ the number of points in S having p as their nearest neighbor. We have

$$\begin{aligned} \mathrm{E}\left[s(n(q))\right] &= \frac{1}{|S|}\sum_{p\in S} s(n(q)) \\ &= \frac{1}{|S|}\sum_{p'\in S} s(p')\delta(p) \\ &\leq \mathrm{E}\left[s(q)\right]\cdot \tau_d, \end{aligned}$$

where τ_d denotes the *kissing number* [40], i.e., the number of d-spheres which can touch a d-sphere of same size without intersecting, or equivalently the number of points that can be placed on the surface of a d-sphere in such a way that the angular separation of two points is at least $\pi/3$. For instance the kissing numbers for $d = 2, 3$ are $\tau_2 = 6$ and $\tau_3 = 12$. By assumption $\mathrm{E}\left[s(q)\right]$ is constant.

The special cases of the d-ball and the d-cube follow directly from the linear expected complexity of the Delaunay tessellation [57, 15]. Golin and Na [71] note that the proof of the linear complexity of the d-ball can be extended to any "reasonably" smooth full dimensional bounded region. □

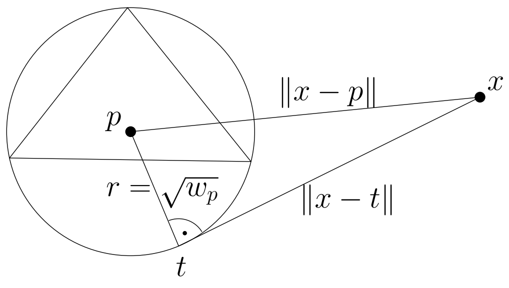


Figure 4.8: Illustration of power distance.

4.6.4 Notes on Oriented Walk Variant

The oriented walk variant is more difficult to analyze than the straight-line walk since the simplices visited by the oriented walk are not as easy to describe geometrically. However, an analysis may use the following monotonicity criterion of the oriented walk.

Monotonicity Criterion. The monotonicity criterion of the oriented walk captures the notion that traversed simplices are closer to the new point than the starting point of the walk. As monotonicity criterion one can take the decreasing power distance from the circumspheres of Delaunay simplices to the query point.

A different criterion has been proposed by Zhu [154]. He suggests to link the cost of the oriented walk to the cost of the straight line walk. But it seems difficult to prove such a link. One difficulty are the dependencies between the walking steps.

The power distance from a point $x \in \mathbb{R}^d$ to a point p with weight w_p is $p \in \mathbb{R}^d$ and $w_p \in \mathbb{R}$ is

$$\pi_p(x) = \|x-p\|^2 - w_p.$$

A geometric interpretation of the power distance is shown in Figure 4.8. Consider the sphere S of radius $\sqrt{w_p}$ around p. Let t be a point on this sphere which lies on a tangent line through x. Then we have $\pi_p(x) = \|x-t\|^2$. In the following the weighted point p and its weight will not be explicitly given but instead the sphere around p. We will call this distance the *power distance of the sphere S* to the point x.

The following basic observation is illustrated in Figure 4.9 (see also [59]).

Observation 4.7. *If an oriented walk in a Delaunay tessellation visits the simplex T_2 after the simplex T_1 then the power distance from the circumsphere of T_2 to q is less (or possibly equal if the points are not in general position) than the power distance from the circumsphere of T_1 to q.*

Proof. By induction it suffices to consider the case that T_2 is visited directly after T_1 by the $(d-1)$-face f they share. The hyperplane through f is the bisector of the two spheres. By the definition of the walk the point q lies on the side of T_2 and by the empty sphere property of the Delaunay tessellation this is the side closer to the circumsphere of T_2. □

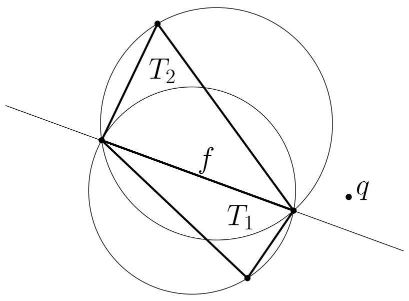


Figure 4.9: The power distance of the Delaunay circumspheres to the query point decreases along the oriented walk.

A different viewpoint on the above observation is the following proposition. The power diagram is the Voronoi diagram using as distance the power distance.

Proposition 4.8 (Weller [151]). *In the interior of the convex hull of a point set, the Delaunay tessellation of the point set is identical to the power diagram of the Delaunay circumspheres. Therefore the distance of the circumspheres to the query point monotonously decreases along the walk.*

Possible Analysis. Based on the monotonicity criterion, the oriented walk variant might be analysed as follows.

1. Bound the power distance to the new point for the first simplex.
2. Bound the number of simplices that can be closer with respect to the monotonicity criterion.

The first step may be done in two substeps.

1.a Bound the Euclidean distance between the new point and the first point found by a random variable R_1.

1.b Bound the diameter of circumspheres of simplices at this vertex by a random variable R_2.

This gives a sphere S around the new point with radius $R_1 + R_2$ in which the circumspheres of the starting simplex lies. The radius $R_1 + R_2$ of the sphere S is an upper bound on the square root of the power distance from the circumsphere of the starting simplex to the new point. For the second step one may bound the number of points for which the circumspheres of simplices at these points can intersect the sphere S.

The difficult case for the analysis is again the boundary case. Handling the boundary case seems to require further insight on the oriented walk and is an interesting open problem.

Conclusion

In this chapter we presented an alternative probabilistic analysis of incremental constructions con BRIO using space-filling curve orders. In contrast to the analysis in the previous chapter this analysis emphasizes the structure of random space-filling curve tours rather than the structure of random Delaunay tessellations. Space-filling curve tours seem easier to analyze and in Lemma 4.3 we give a general bound on the length of tour segments. With this bound we can easily prove that the incremental construction runs in linear expected time in higher dimensions for uniformly distributed points in a cube. In contrast an analysis in the spirit of the previous chapter would require a better understanding of random Delaunay tessellations already in three dimensions.

A drawback of considering the space-filling curve tour instead of the Delaunay tessellation is that we do not analyze the behaviour of the tessellation during the round. This forces us to use an additional point location data structure in the analysis. In an implementation an additional data structure might not be desirable, and it is an open problem to give the analysis without the data structure.

A further challenge which is not apparent in the case of uniformly distributed points but in the case of normally distributed points is the boundary behaviour of space-filling curve tours, i.e., its behaviour near to the convex hull of a point set. A better understanding of the boundary case for space-filling curves could make it easier to apply the analysis to other distributions. Furthermore, to prove a bound on the running time for more general distributions, e.g., normally distributed points in higher dimensions, it would be necessary to know the expected complexity of the Delaunay tessellations.

There are several fast implementations of incremental constructions con BRIO using space-filling curve orders available. Using our implementation we tested the algorithm on various distributions and for different sampling ratios. The experiments indicate a linear running time for all tested distribution. While our analysis applies to several of the tested point distributions, there are others to which it does not apply. Of these, points distributed on a surface are of particular interest in applications. Giving an analysis for this case remains an open problem. For this a different ordering might be more suitable since space-filling curves are typically bound to a certain dimension.

In the last part of this chapter we gave an analysis of an algorithm by Su and Drysdale. We proved that it runs in linear expected time for uniformly distributed points in arbitrary dimension. The analyzed case differs slightly from the implementation, and it would be interesting to extend the analysis to the version of the algorithm which uses an oriented walk. For this it would be sufficient to understand the behaviour of the walk for short distances. Beyond this, a more general analysis of the oriented walk would be of interest.