TRIANGULATIONS IN $\mathbb{R}^2$

HANG SI

Contents

1. Introduction

Triangulations of topological and geometric objects are central topics in many different parts of mathematics and computer science. They are the natural way to represent a region of interest into smaller, easy-to-handle pieces. Many problems like collision detection, ray tracing, shortest path, and so on are efficiently solved by triangulations.

Triangulations of point sets are fundamental objects studied and used in computational geometry. They are the primitive types of unstructured meshes. Typically, given a set of points in space, we would like to connect them in a nice way. The meaning of "nice" depends on the applications in which the meshes to be used. Geometrically, one would like to connect the nearest neighbours and to avoid small angles. This leads to the introduction of the well-known Delaunay triangulations, which have many of these nice properties. We discuss efficient algorithms to compute Delaunay triangulations.

2. Triangulations

In this section, we formally define triangulations of point sets as a special type of simplical complexes. We then introduce the Euler's formula as the basic tool for understanding the complexity of triangulations. A line-sweep algorithm is given for efficiently constructing a triangulation for a pint set in the plane.

2.1. Simplical Complexes.

Recall the *convex hull* of a set X of points is the most smallest convex set that contains the set X.

Recall a set $X = \{\mathbf{u}_0, \ldots, \mathbf{u}_k\}$ of points is said to be *affinely independent* if the set $Y = \{\mathbf{u}_1 - \mathbf{u}_0, \ldots, \mathbf{u}_k - \mathbf{u}_0\}$ of vectors are linearly independent (the reverse is not true).

Recall the *dimension* of an affine space is the dimension of its associated vector space. This means that $\{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_n\}$ is an affine basis of an affine space if and only if $\{\mathbf{x}_1 - \mathbf{x}_0, ..., \mathbf{x}_n - \mathbf{x}_0\}$ is a linear basis of the associated line space.

Definition 1. *A k-simplex σ is the convex hull of $k+1$ affinely independent points, where k is the dimension of σ.*

For examples, a 0-, 1-, 2-, and 3-simplex is a point, line segment (edge), a triangle, and a tetrahedron, respectively, see Figure 2.1.

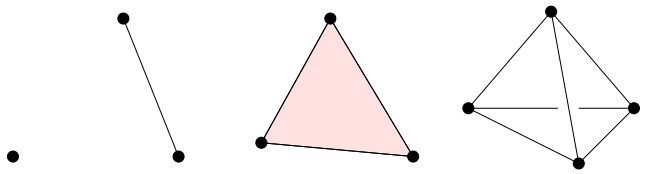

Figure 1. From left to right, a 0-, 1-, 2-, and 3-simplex.

Definition 2. *The $k+1$ points that define a k-simplex σ are called vertices of σ. The convex hull of any $i+1$ vertices of σ is called an i-face of σ.*

For examples, an 0-, 1-, and 2-faces are vertex, edge, and triangle, respectively.

Note that the empty set is also a face (of dimension -1) of σ.

A i-face is itself an i-simplex.

Exercise. Count the number of all faces of a triangle ($1 + 3 + 3 + 1 = 8$ faces).

A k-simplex has $\sum_{i=-1}^{k} \binom{k+1}{i+1} = 2^{k+1}$ faces.

Definition 3. *A simplicial complex $\mathcal{K}$ is a set of simplifies such that*

(1) $\sigma \in \mathcal{K} \longrightarrow \forall \tau \in \mathcal{K}$, *where τ is a face of σ; and*
(2) $\sigma, \tau \in \mathcal{K} \longrightarrow \sigma \cap \tau \in \mathcal{K}$.

The *dimension* of a simplicial complex $\mathcal{K}$ is the largest dimension of it simplices.

The union of all simplices of $\mathcal{K}$ is called the *underlying space* of $\mathcal{K}$, which is denoted as $|\mathcal{K}|$.

Definition 4. *Given a point set V in $\mathbb{R}^2$, a triangulation of V is a 2-dimensional simplicial complex $\mathcal{T}$ such that*

(1) *V is the vertex set of $\mathcal{T}$; and*
(2) *$|\mathcal{T}|$ is the convex hull of V.*

In this definition of triangulations, we do not require that a triangulation of V uses all vertices of V. However, in the following discussion (unless stated explicitly), we assume that all vertices of V are used.

2.2. **Euler's Formula.** This section gives some basic facts and formulas about the combinatorial structures of a triangulation using well developed theories in graph theory. The most important fact is the famous Euler's formula which bounds the number of vertices, edges, and faces of a triangulation. These results are necessary analysing triangulation algorithms.

Recall a graph $\mathcal{G} = (V, E)$ is a set V of vertices, and a set E of edges, each a pair of vertices of V. A graph is *simple* if every edge has two distinct vertices and no two edges have the same vertices. A simple graph is *connected* if there is a *path* (a sequence of edges) that connecting every pair of its vertices.

A triangulation in $\mathbb{R}^2$ is a *planar graph*, which is a graph that can be drawn in the plane without crossing edges. In other words, the graph is *planar* if it has an embedding in the plane without crossing edges. For example, the complete graphs K_4 is planar but K_5 is not planar.

Let $\mathcal{G} = (V, E)$ be a planar graph. It decomposes the plane into a set of regions, which are called *faces* of $\mathcal{G}$. Let v, e, and f denote the number of vertices, edges, and faces of $\mathcal{G}$. Euler's formula is a linear relation between these numbers.

Theorem 1. *Every connected planar graph $\mathcal{G} = (V, E)$ satisfies*

$$v - e + f = 2. \tag{1}$$

This formula is well-known as the Euler's formula for planar graphs and convex 3d polytopes. There are plenty of proofs, see a collecttion of different proofs of this formula by D. Eppstein's "Geometry Junkyard" [1]. Below we give one of the proofs based on dual graphs.

Proof. Let $\mathcal{G} = (V, E)$ be a connected planar graph. Define the *dual graph* $\mathcal{G}^* = (V^*, E^*)$ of $\mathcal{G}$, such that every vertex in V^* corresponds to a face of $\mathcal{G}$, and every edge in E^* corresponds to two adjacent faces of $\mathcal{G}$. Let F be the set of faces of $\mathcal{G}$, we have V^* and F are bijective, and E^* and E are bijective.

Choose any spanning tree $\mathcal{T}$ of $\mathcal{G}$. It has v vertices, and $v - 1$ edges. The dual edges of $(\mathcal{G} - \mathcal{T})^*$ is also a spanning tree of $\mathcal{G}^*$. The two spanning trees together have $v - 1 + f - 1$ edges. □

[1] `https://www.ics.uci.edu/~eppstein/junkyard/euler/`

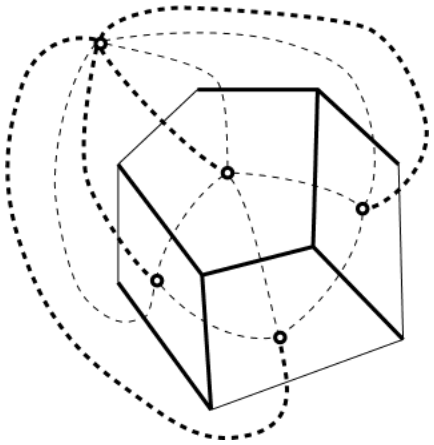

FIGURE 2. A proof of Euler's formula through the spanning trees of dual graphs.

A triangulation $\mathcal{T}$ of a set of vertices in $\mathbb{R}^2$ is a *maximumly connected* planar graph, which means by adding any one more edge to $\mathcal{T}$ will violate the planarity. Using the Euler's formula, we can get upper bounds on the number of edges and faces of $\mathcal{T}$ in terms of the number of vertices of it vertex set.

In a triangulation $\mathcal{T}$, every face have three edges, every interior edge is shared by two faces, and every convex hull edge is shared by one face, we have:

$$3f = 2e - h, \tag{2}$$

where h is the number of edges on the convex hull of $\mathcal{T}$. Since $h > 0$, then,

$$3f < 2e. \tag{3}$$

Using the Euler's formula and the above inequality, we can bound the total number of edges and faces of $\mathcal{T}$, which are

$$\begin{aligned} e &\leq 3v - 6, \\ f &\leq 2v - 4. \end{aligned} \tag{4}$$

2.3. **Line-sweep Algorithm.** In this section, we introduce a simple and efficient algorithm using line-sweeping [**?**] to construct triangulations from a set of points.

The basic idea is to sort the point set along a fixed direction (for example, the x-axis), then use a (vertical) line that sweeps over the plane from left to right. The triangulation is created online during the line sweeping. An invariant is: at any moment in time, the partial triangulation contains all points to the left of the line. When the line hits a new vertex (an event), the triangulation is augmented by creating new triangles connecting to this new vertex. The algorithm is given in Figure 3.

In line 1, the vertices in L are ordered in such a way, that no conflict will occur during the algorithm. A simple order is the lexicographic (dictionary) order along

Algorithm: LineSweep(V, $\mathbf{s}$)
Input: A set V of n points in the plane, $\mathbf{s}$ is normal of sweep line;
Output: A triangulation $\mathcal{T}$ of V;
1 sort the points in V into a sequence $L := \{\mathbf{v}_1, \dots \mathbf{v}_n\}$ along $\mathbf{s}$;
2 initialize $\mathcal{T}$ with only one triangle $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$;
3 **for** $i = 4$ to n **do**
4 let Q be the set of all hull edges that sees $\mathbf{v}_i$;
5 create new triangles to $\mathcal{T}$ by each edge in Q and $\mathbf{v}_i$;
6 **endfor**

FIGURE 3. The sweep line triangulation algorithm.

the x- or y-axis. To efficiently obtain the set Q (in line 4), one could start from the last newly created triangle, which must contain a hull edge e. Then the set of all hull edges which are visible by $\mathbf{v}_i$ can be collected by a breadth search from e.

The sort of a set of vertices along the sweep line can be done in time $O(n \log n)$. The number of hull edges which are visible by each $\mathbf{v}_i$ is a constant independent of n. The total number of newly created triangles is less than $2n - 4$. Thus this line sweep algorithm constructs a triangulation in $O(n \log n)$ time.

Once the triangulation of V is constructed, we also obtain the convex hull of the point set V by outputting the set of edges which only shared by one triangle. An algorithm to output convex hull is described in the next section.

3. Delaunay Triangulations

This section focus on Delaunay triangulations for finite point sets in the plane. The Delaunay triangulation of a point set is introduced by the Russian mathematician Boris Nikolaevich Delone (1890–1980) in 1934. It is a triangulation with many nice properties. There are many ways to define Delaunay triangulations. We first introduce them as duals of Voronoi diagrams, then introduce other equivalent definitions while showing their properties.

3.1. **Voronoi Diagrams.**

Voronoi diagrams are named after the Russian and Ukrainian mathematician Georgy Feodosevich Voronoy (1868–1908) in 1907. Voronoi diagrams arise in nature in various situations, see an example in Figure 3.1. They are the one of the most fundamental data structures in computational geometry.

Voronoi diagram divides the plane according to the *nearest-neighbour rule*: Each point is associated with region of the plane closet to it.

Consider the simplest case of two points $\mathbf{p}$ and $\mathbf{q}$ in the plane. The set of points that are at least as close to $\mathbf{p}$ as to $\mathbf{q}$ is the *half-space*:

$$H_{pq} = \{\mathbf{x} \in \mathbb{R}^2 \,:\, \|\mathbf{x} - \mathbf{p}\| \leq \|\mathbf{x} - \mathbf{q}\|\},$$

where $\|\cdot\|$ means Euclidean distance.

FIGURE 4. At the moment when a bullet hits a glass, the smashed glass forms a structure resembles a Voronoi diagram.

Definition 5. *Let S be a set of n points (called sites) in $\mathbb{R}^2$. the Voronoi region of a site $\mathbf{p} \in S$ is the set of points $\mathbf{x} \in \mathbb{R}^2$ that are as close to $\mathbf{p}$ as to any other site in S, that is*

$$V_p = \{\mathbf{x} \in \mathbb{R}^2 \, : \, \|\mathbf{x} - \mathbf{p}\| \leq \|\mathbf{x} - \mathbf{q}\|, \forall \mathbf{q} \in S\},$$

The Voronoi region of $\mathbf{p}$ is the intersection of a set of half-spaces. It follows that it is a convex polygonal region, possibly unbounded, with at most $n-1$ edges.

Each point $\mathbf{x} \in \mathbb{R}^2$ has at least one nearest site in S, so it belongs to at least one Voronoi region. It follows that the Voronoi regions cover the entire plane. Two Voronoi regions lie on opposite sides of the perpendicular bisector separating the two generating points.

Definition 6. *The Voronoi regions together with their edges and vertices form the Voronoi diagram of S.*

Voronoi diagram is a planar graph with n regions and minimum vertex degree 3. Each of the e edges has two vertices, and each of the v vertices belongs to at least three edges. Hence $2e \geq 3v$. Euler's formula $n + v - e = 2$ implies $e \leq 3n - 6$ and $v \leq 2n - 4$. In average, the number of edges of each Voronoi regions is less than 6.

Voronoi diagrams are important and useful in a wide variety of fields inside and outside computer science, such as the post-office location problem, collision detections, and so on, see e.g., [**?**].

We get a dual diagram if we draw a straight line connecting $\mathbf{p}$ and $\mathbf{q}$ in S if their Voronoi regions share a common line segment. In general, if no four sites of S share a common circle, i.e., S is in *general position*, the dual diagram is a 2-dimensional simplicial complex which decompose the convex hull of S. It is called the *Delaunay triangulation* of S, see Figure 3.1 for an example.

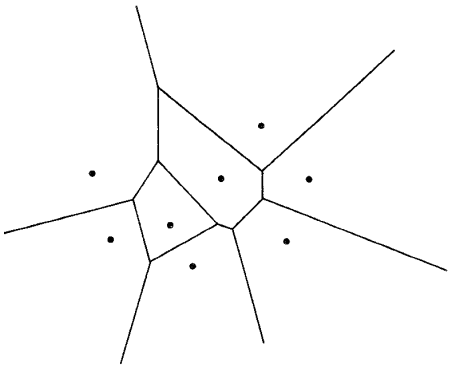

FIGURE 5. Voronoi diagram for eight points in the plane.

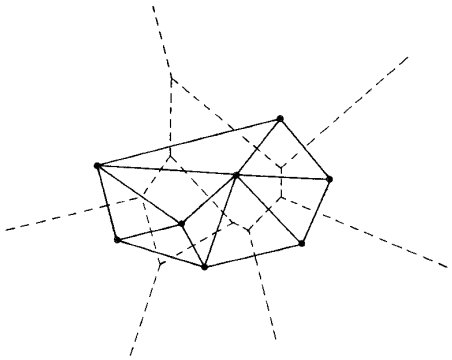

FIGURE 6. The dual of Voronoi diagram is the Delaunay triangulation.

There is an ambiguity in the definition of Delaunay triangulation if four or more Voronoi regions meet at a common point, i.e., four sites of S share a common circle. Probabilistically, the chance of picking four points on the circle is zero because the circle defined by first three points has zero measure in $\mathbb{R}^2$. A common way to say the same thing is that four points in a common circle for a *degeneracy* or a *special case*. An arbitrary small perturbation suffices to remove the degeneracy and to reduce the special to general case.

We will often assume *general position*, which is the absence of any degeneracy. This really means that we delay the treatment of degenerate cases to later.

Exercise: Show that If S is in general position, the Delaunay triangulation of S is unique.

3.2. **The Empty Circumcircle Property.** Alternatively, Delaunay triangulations can be defined through the so-called *empty circumcircle property*.

The *circumcircle*, or *circumscribing circle* of a simplex σ is the circle that passes through all vertices of σ. In plane, a triangle has a unique circumcircle, while an edge has infinitely many circumcircles.

Definition 7. *Let S be a set of n points (called sites) in $\mathbb{R}^2$. A simplex σ whose vertices are in S is Delaunay if it has a circumcircle that encloses no site in S. We say, that σ has an empty circumcircle.*

Definition 8. *If S is in general position, then the set of all Delaunay simplices of S form a simplical complex whose union is the convex hull of S, it is the Delaunay triangulaiton of S.*

Note that if S is not in general position, then the set of all Delaunay simplices of S is not a simplical complex. There are simplicies overlapping each other. In this case, one could still obtain a Delaunay subduevsion by deleting all Delaunay simplifies which are overlapping each other.

3.2.1. *A trivial algorithm.* One immediately application of this empty circumcircle property is that it gives you a trivial way to find the Delaunay triangulation of a point set. Assuming the point set is in general position. One can find all the Delaunay triangles from the set of all $\binom{n}{3}$ triangles by checking the empty circumcircle property. However, this way needs $O(n^4)$ time. We will discuss more efficient algorithms later.

3.2.2. *Minimum spanning trees.* Delaunay triangulation tends to connect the nearest neighbours. However, it is not the minimum distance triangulation of the point set. A counter example is given by Lawson [**1972**]. Instead, the problem to find minimum weighted triangulation is NP-hard [**?**].

As an important fact, Delaunay triangulation is a supergraph of the minimum spanning tree that spanned by the set of vertices in the plane.

Recall a *tree* is an undirected graph in which any two vertices are connected by exactly one path. In other words, any connected graph without simple cycles is a tree. A *complete graph* is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge. A *spanning tree* of that graph is a subgraph that is a tree and connects all the vertices together.

Take the complete graph of the set of points and define the length of an edge as the Euclidean distance between its endpoints. A *Euclidean minimum spanning tree* of that graph is then a spanning tree with edge length less than or equal to the edge length of every other spanning tree.

Theorem 2. *All edges of every Euclidean minimum spanning tree belong to the Delaunay triangulation of the same point set.*

We leave the proof as an exercise.

3.3. **The Lifting Map and Convex Hulls.** The *convex hull* of a finite set S of points is the smallest convex set that contains S. It is also the intersection of all half-spaces containing S. In this section, we introduce the relation of Delaunay triangulations and the convex hulls.

Let S be a set of n points (called *sites*) in $\mathbb{R}^2$. We now consider, for each site $\mathbf{p} = (p_x, p_y) \in S$, a point $\hat{\mathbf{p}} = (p_x, p_y, p_z) \in \mathbb{R}^3$, where $p_z := p_x^2 + p_y^2$, i.e., $\hat{\mathbf{p}}$ is a point on the paraboloid $z = x^2 + y^2$ in $\mathbb{R}^3$, and $\mathbf{p}$ is the projection of $\hat{\mathbf{p}}$ into the plane by removing its z-coordinate, see Figure 3.3. We call this map $f : \mathbf{p} \in \mathbb{R}^2 \to \hat{\mathbf{p}} \in \mathbb{R}^3$ the *lifting map*.

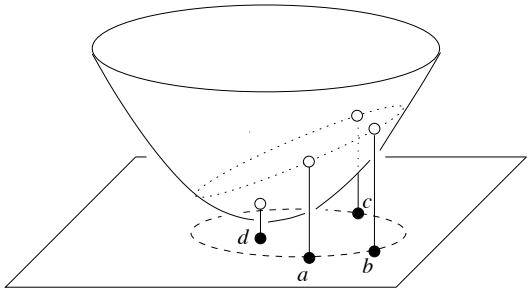

FIGURE 7. The lifting map that takes a point in the plane to a paraboloid in $\mathbb{R}^3$.

We have the following fact which gives the relation between circles in $\mathbb{R}^2$ and plane in $\mathbb{R}^3$.

Proposition 1. *Point* $\mathbf{d}$ *lies inside the circumcircle of the triangle with vertices* $\mathbf{a}$, $\mathbf{b}$, *and* $\mathbf{c}$ *if and only if point* $\hat{\mathbf{d}}$ *lies vertically below the plane passing through the points* $\hat{\mathbf{a}}$, $\hat{\mathbf{b}}$, *and* $\hat{\mathbf{c}}$.

Proof. ... □

Due to the above fact, we can easily show the nice relation between Delaunay triangulation and convex hulls.

Let $\hat{S}$ be the set of all sites resulting obtained by the lifting map on S. The convex hull of $\hat{S}$ is a 3d convex polytope, denoted as $\text{conv}(\hat{S})$. A *lower face* of $\text{conv}(\hat{S})$ if there is no point of $\text{conv}(\hat{S})$ below it with respect to the z-axis. The projection of the set of lower faces of $\text{conv}(\hat{S})$ into the xy-plane gives a subdivision of the convex hull of S. If S is in general position, then this subdivision is a *simplical complex* $\mathcal{T}$ and the circumcircle of every triangle in $\mathcal{T}$ is empty (due to Proposition), hence $\mathcal{T}$ is the *Delaunay triangulation* of S. Figure 8 shows an example.

3.4. **Primitives for Delaunay Triangulation Algorithms.** This section describes two geometric primitives, the "orientation test" and the "incircle test", that are sufficient for implementing many Delaunay triangulation algorithms.

The *orientation* of a sequence of $d+1$ points $(\mathbf{p}_1, \ldots, \mathbf{p}_{d+1})$ in $\mathbb{R}^d$ is either -1, 0, or $+1$. It is 0 if the set of $d+1$ points lies on a common k-flat for $k < d$. In $\mathbb{R}^2$, three points $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ has positive orientation if $\mathbf{p}_3$ lies to the left of the line

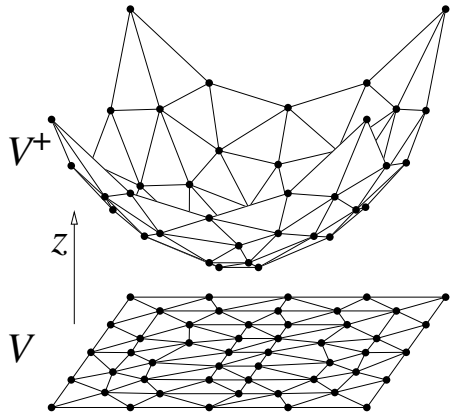

FIGURE 8. The parabolic lifting map. In this example, a two-dimensional vertex set V is lifted to a paraboloid in $\mathbb{R}^3$. The underside of the convex hull of the lifted vertices projects down to a Delaunay triangulation of V.

directed from $\mathbf{p}_1$ to $\mathbf{p}_2$, or equivalently, if $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ are in counterclockwise order. Orientation is preserved under an even permutation of the points and negated under an odd permutation. The orientation of $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ is given by the sign of the determinant

$$\begin{vmatrix} p_{11} & p_{12} & 1 \\ p_{21} & p_{32} & 1 \\ p_{31} & p_{32} & 1 \end{vmatrix}$$

The value of this determinant is twice of the signed area of the triangle with vertices $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$.

Suppose the sequence of 3 points $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ in $\mathbb{R}^2$ has positive orientation. By the above Proposition, a point $\mathbf{p}_4$ lies outside the circumcircle of $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ exactly if $\hat{\mathbf{p}}_4$ lies above the plane passing though $\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \hat{\mathbf{p}}_3$ in $\mathbb{R}^3$. In this case, $\hat{\mathbf{p}}_4$ sees $\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \hat{\mathbf{p}}_3$ form a counterclockwise order in the plane. It is exactly the orientation test of 4 points $(\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \hat{\mathbf{p}}_3, \hat{\mathbf{p}}_4)$ in $\mathbb{R}^3$, which is

$$\begin{vmatrix} p_{11} & p_{12} & p_{11}^2 + p_{12}^2 & 1 \\ p_{21} & p_{32} & p_{21}^2 + p_{22}^2 & 1 \\ p_{31} & p_{32} & p_{31}^2 + p_{32}^2 & 1 \end{vmatrix}$$

The evaluation of the sign of this determinant is the *incircle test* for points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$. If the determinant is 0, then $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ are co-circular.

3.5. **Lawson's Edge Flip Algorithm.** This section introduces a local condition for edges, shows it implies a triangulation is Delaunay, and derives an algorithm based on edge flipping. The correctness of this algorithm implies two important results of planar triangulations, (1) among all triangulations of the same point set,

the Delaunay triangulation maximises the minimum angle; and (2) the set of all triangulations of the point set is connected by edge flips.

Definition 9. *Let $\mathcal{K}$ be a triangulation of a point set S in $\mathbb{R}^2$. An edge $\mathbf{ab} \in \mathcal{K}$ is locally Delaunay if*

(i) *it belongs to only one triangle and therefore bounds the convex hull, or*
(ii) *it belongs to triangles,* **abc** *and* **abd**, *and* **d** *lies outside the circumcircle of* **abc**.

This definition is illustrated in Figure 3.5. A locally Delaunay edge is not necessary a Delaunay edge. However, if every edge is locally Delaunay, then we can show that all are Delaunay edges.

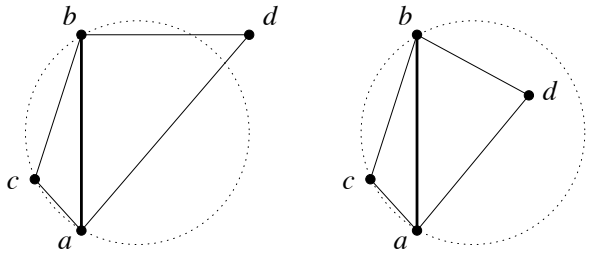

FIGURE 9. To the left **ab** is locally Delaunay and to the right it is not.

Theorem 3 (Delaunay Lemma)**.** *If every edge of $\mathcal{K}$ is locally Delaunay, then $\mathcal{K}$ is the Delaunay triangulation of S.*

Proof. ... □

3.5.1. *The Edge-flip Algorithm.* If an edge **ab** is not locally Delaunay, let the two triangles sharing at **ab** are **abc** and **abd**. The union of these two triangles must be a convex quadrangle. We can *flip* **ab** to **cd**. Formally, this means we remove **ab**, **abc**, and **abd** from the triangulation, and add **cd**, **cda**, and **cdb** to the triangulation. The resulting edge **cd** must be locally Delaunay.

We can use edge flips as elementary operations to convert an arbitrary triangulation $\mathcal{K}$ to the Delaunay triangulation. The algorithm uses a stack to maintains all edges which may be locally non-Delaunay. Initially, all edges of $\mathcal{K}$ are pushed on the stack.

3.5.2. *Termination and Running Time.* Flipping **ab** to **cd** is likely gluing a tetrahedron $\hat{\mathbf{a}}\hat{\mathbf{b}}\hat{\mathbf{c}}\hat{\mathbf{d}}$ from below to $\hat{\mathbf{a}}\hat{\mathbf{b}}\hat{\mathbf{c}}$ and $\hat{\mathbf{a}}\hat{\mathbf{b}}\hat{\mathbf{d}}$. The algorithm can be understood as gluing a sequence of tetrahedra. Once we glue $\hat{\mathbf{a}}\hat{\mathbf{b}}\hat{\mathbf{c}}\hat{\mathbf{d}}$ we cannot glue another tetrahedron right below $\hat{\mathbf{a}}\hat{\mathbf{b}}$. In other words, once we flip **ab** we cannot introduce **ab** again by some other flip. This implies that the Lawson's edge-flip algorithm will

Algorithm: LawsonFlip(L)
Input: a stack L of edges of a triangulation $\mathcal{K}$;
Output: the modified triangulation $\mathcal{K}$;
1 **while** $L \neq \emptyset$ **do**
2 pop an edge **ab** from L;
3 **if ab** is not locally Delaunay **then**;
4 flip **ab** to **cd**;
5 push edges **ac**, **cb**, **db**, **da** on L;
6 **endif**
7 **endwhile**

FIGURE 10. The Lawson edge-flip algorithm.

eventually terminate when all locally non-Delaunay edges are flipped. By the Delaunay lemma, the triangulation is Delaunay. This also implies there are at most as many flips as there are edges connecting n points, namely $\binom{n}{2}$. Each flip takes constant time, hence the total running time is $O(n^2)$.

3.5.3. *Maxmin Angle Property.* We illustrate an optimal property of the Delaunay triangulation.

Theorem 4. *Among all triangulation of a finite point set $S \subset \mathbb{R}^2$, the Delaunay triangulation maximises the minimum angle.*

Proof. A flip substitute two new triangles for two old triangles. It therefore changes six of the angles, see Figure 3.5.3. The six old angles are:

$$\alpha_1, \beta_1, \gamma_1 + \gamma_2, \alpha_2, \beta_2, \delta_1 + \delta_2,$$

and the six new angles are

$$\gamma_1, \delta_1, \beta_1 + \beta_2, \gamma_2, \delta_2, \alpha_1 + \alpha_2.$$

We show that for each of the six new angles there is an old angle that is at least as small. At first, $\gamma_1 \geq \alpha_2$, since both angles are opposite the same edge **bd**, and **a** lies outside the circumcircle of **bdc**. Similarly, $\delta_1 \geq \alpha_1$, $\gamma_2 \geq \beta_2$, $\delta_2 \geq \beta_1$, and for trivial $\alpha_1 + \alpha_2 > \alpha_1$, and $\beta_1 + \beta_2 \geq \beta_1$.

It follows that a flip does not decrease the smallest angle in a triangulation. Since we can transform any triangulation $\mathcal{K}$ of S to the Delaunay triangulation, this implies that the smallest angle in $\mathcal{K}$ is no larger than the smallest angle in the Delaunay triangulation. □

3.5.4. *The Flip Graph.* One can use flips to traverse the set of all triangulations of S. We can form a *flip-graph* $\mathcal{G}$ of S. Each triangulation is a node of $\mathcal{G}$, and each edge of $\mathcal{G}$ between two nodes u and v means there is a flip that changes the triangulation u to v. The termination of Lawson's flip algorithm implies that the

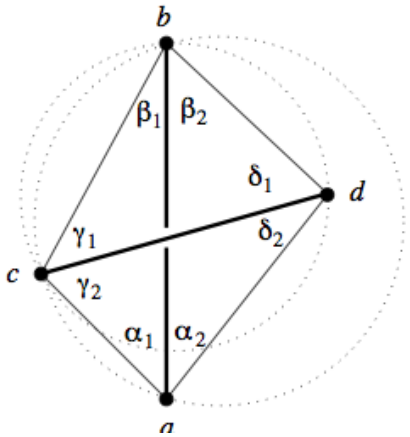

FIGURE 11. Flipping **ab** to **cd** changes six of the angles.

flip-graph is connected. Moreover, one can go from any triangulation of S to any other triangulation in less than $O(n^2)$ flips.

3.6. **Randomized Incremental Algorithm.** In this section, we introduce an algorithm that construct Delaunay triangulations incrementally, using edge flips and randomisation. After explain the algorithm, we present a detailed analysis of the expected running time.

The basic step of this algorithm is to interleave flipping edges and adding points. Denote the points in $S \subset \mathbb{R}^2$ as $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$, and assume general position. To reduce the outside to the inside case, we start with a triangulation $\mathcal{D}_0$ that consists of a single and sufficiently large triangle **xyz**. The algorithm is a `for-loop` adding the points in sequence. After adding a point, it uses edge flips to satisfy the Delaunay lemma before the next point is added. The algorithm is given in Figure 12.

Algorithm: IncrementalFlip($S = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$)
Input: a sequence S of n points in $\mathbb{R}^2$;
Output: the Delaunay triangulation $\mathcal{D}_n$ of S;
1 initialize $\mathcal{D}_0$ with only one larger triangle **xyz**;
2 **for** $i = 1$ to n **do**
3 find the triangle $\tau \in \mathcal{D}_{i-1}$ containing $\mathbf{p}_i$;
4 split τ into three triangles containing $\mathbf{p}_i$;
5 initial the stack L with three link edges of $\mathbf{p}_i$;
6 LawsonFlip(L);
7 **endfor**
8 remove all triangles containing **x**, **y**, and **z** from $\mathcal{D}_n$;

FIGURE 12. The incremental-flip algorithm.

3.6.1. *Number of Flips.* The running time of this algorithm consists of two parts: (1) the time to locate the triangle τ containing the vertex $\mathbf{p}_i$; and (2) the time to perform flips. In this section, we study the total number of flips that will be done in this algorithm. We consider two cases: the worst case and the expected case when vertices are inserted in a random order.

Note that every new triangle in $\mathcal{D}_i$ has $\mathbf{p}_i$ as vertex. Indeed, if $\mathbf{abc}$ is a triangle in $\mathcal{D}_{i-1}$ and its circumcircle does not contain $\mathbf{p}_i$, then it must be also a triangle in $\mathcal{D}_i$. This implies that all flips during the insertion of $\mathbf{p}_i$ occur right around $\mathbf{p}_i$. This implies the number of edges flips is related to the degree of $\mathbf{p}_i$, i.e. the number of edges which have $\mathbf{p}_i$ as a vertex. It is denoted as $\deg(\mathbf{p}_i)$. In the incremental flip algorithm, each edge flip increases the degree of $\mathbf{p}_i$ by 1. Since the initial degree of $\mathbf{p}_i$ is 3 (creating by splitting the triangle τ), then the number of edge flips to add $\mathbf{p}_i$ in $\mathcal{D}_i$ is equal to $\deg(\mathbf{p}_i) - 3$.

We first consider the worst case. Figure 3.6.1 shows such an example. Assuming the sequence of vertices is ordered first from left to right, then from bottom to top. The degree of each successive vertices can be $\Theta(n)$, hence the total number of flips is $\Theta(n^2)$. This shows that if the insertion order of the vertices are chosen badly, the incremental flip algorithm can take $\Theta(n^2)$ time.

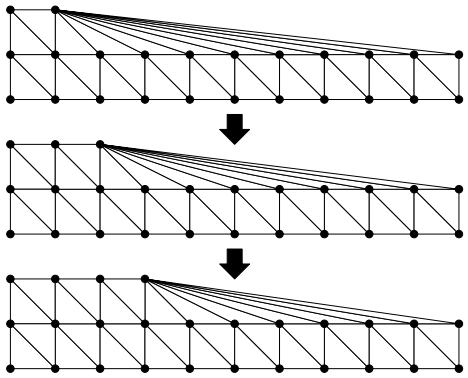

FIGURE 13. Each vertex insertion can case $\theta(n)$ flips.

3.6.2. *Basic Definitions of Probability.* Here let's review the basic definitions of a *probability space*. What is an "outcome", what is the "probability of an outcome", what is an "event", etc. The following figure from MIT open course is very helpful.
Sample space, outcome, event. . A countable *sample space*, $\mathcal{S}$, is a nonempty countable set. An element $w \in \mathcal{S}$ is called an *outcome*. A subset of $\mathcal{S}$ is called an *event*.
Probability space. . A *probability function* on a sample space $\mathcal{S}$, is a total function $\Pr\{\} : \mathcal{S} \to \mathbb{R}$ such that

- $\Pr\{w\} \geq 0$ for all $w \in \mathcal{S}$, and

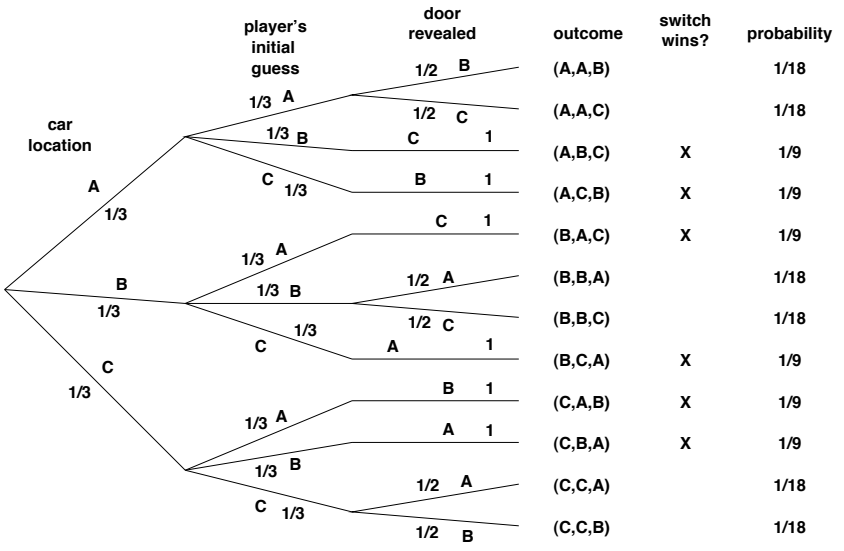

FIGURE 14. The tree diagram of the four-step method. Step (1), find the sample space, Step (2) define the events of interest, Step (3) Determine the outcome probablities, Step (4) Compute event probabilities.

- $\sum_{w \in \mathcal{S}} \Pr\{w\} = 1$.

The sample space together with a probability function is called a *probability space*.

Conditional probability. (Optional). A *conditional probability* is the probability that one event happens, given that some other event definitely happens. For example, the question like: what is the probability that two rolled dice sum to 10, given that both are odd?

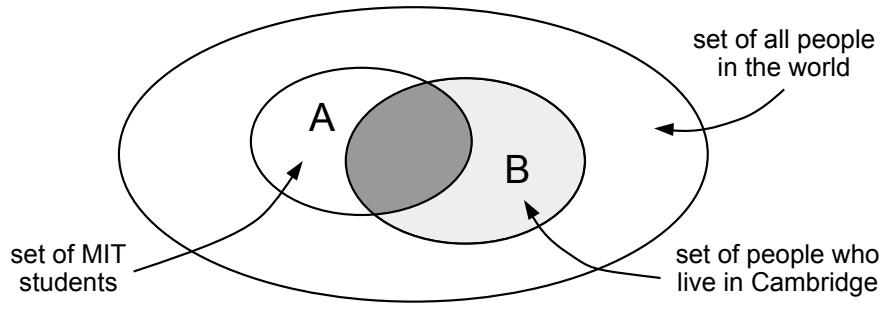

FIGURE 15. Conditional Probability.

In general, $\Pr\{A|B\}$ denotes the probability of event A, given that event B happens, where

$$\Pr\{A|B\} := \frac{\Pr\{A \cap B\}}{\Pr\{B\}}.$$

Note that if $\Pr\{B\} = 0$ that the conditional probability $\Pr\{A|B\}$ is undefined.

Random variable. . A *random variable*, R, on a probability space is a total function whose domain is the sample space.

The codomain (range) of R can be anything, but will usually be a subset of the real numbers.

3.6.3. *The Expected Number of Flips.* We next show if the vertices are inserted in a random order, the expected total number of flips is only $O(n)$. Random does not mean arbitrary but rather that every permutation of the n points is equally likely. There are total $n!$ permutations. Let $S_1, \ldots, S_m$ be the $m = n!$ permutations of n points. Let F_i be the total number of flips resulted by the incremental-flip algorithm on S_i. Let F is the *random variable* which is a function from the finite probability space $P = \{S_1, \ldots, S_m\}$ to the total number of flips. Since we assume every permutation is equally likely, then the probability of any particularly sequence is $\Pr\{S_i\} = \frac{1}{n!}$. Hence the expected number of flips of the incremental-flip algorithm is

$$\mathrm{E}[F] := \sum_{i=1}^{m} F_i \Pr\{S_i\} = (F_1 + F_2 + \cdots + F_m)\frac{1}{n!}. \tag{5}$$

For an example, let R (a random variable) be the number comes up with a fair, six sided die. The expected value of R is:

$$\mathrm{E}[R] := \sum_{i=1}^{6} i \Pr\{R_i = i\} = 1\frac{1}{6} + 2\frac{1}{6} + \cdots + 6\frac{1}{6} = \frac{7}{2}.$$

The technique we use to analyse the algorithm is called the backward analysis.

Consider inserting the last point $\mathbf{p}_n$. The sum of degrees of all possible last points is the same as the sum of degrees of all points in $\mathcal{D}_n$ (due to the uniqueness of the Delaunay triangulation). The latter is equal to twice the number of edges, which is

$$\sum_{i=1}^{n} \deg(\mathbf{p}_i) \leq 2(3n - 6) \leq 6n.$$

Note that each of the last point appears $(n-1)!$ times in all the $n!$ permutations. Therefore the number of flips for adding all last points is at most:

$$f_n \leq (6n - 3n)(n-1)! \leq 3n(n-1)!,$$

Then the total number of flips is:

$$\begin{aligned}\textstyle\sum_{i=1}^{n} f_i &= f_n + f_{n-1} + \cdots + f_1 \\ &\leq 3n \cdot (n-1)! + 3(n-1)(n-2)! + \cdots + 0 \\ &\leq 3n \cdot ((n-1)! + (n-2)! + \cdots + 0) \\ &\leq 3n \cdot n \cdot (n-1)! \\ &= 3n \cdot n!\end{aligned}$$

The expected number of edge flips for adding n points is

$$\mathrm{E}[R] \leq 3n \cdot n! \cdot \frac{1}{n!} = 3n.$$

There is a simple way to say the same thing. If points are inserted in a random order, the expected number of flips for the last point is at most 3. Hence the total number of edge flips for adding n points is $O(n)$.

3.6.4. *Point location.* The point location is another important fact for running time. A simple point location scheme is called "straight walk". More precisely, the algorithm starting from an arbitrary triangle $\sigma \in \mathcal{D}_i$, and search the triangle τ that containing $\mathbf{p}_i$ by walking along the ray starting from an interior point of σ toward to $\mathbf{p}_i$, see Figure 3.6.4 Left.

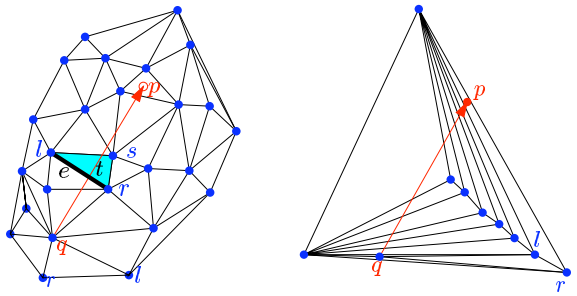

FIGURE 16. Locating a point by straight line walk. (Images from [**Devillers, Pion & Teillaud 2001**])

How much time will this searching algorithm uses? We assume that each triangle is visited by only once. Then each point location will visit at most the number of triangles of the triangulation which is less than $2n$. One can show that this is possible by the triangulation shown in Figure 3.6.4 Right. Hence the location of n points by this algorithm may take $O(n^2)$ time. It is too slow.

A way to improve the point location is to build a history graph point-location data structure (as a helper). A simple approach is based on the idea of maintaining the uninsured points in a set of *buckets*. Think of each triangle of the current triangulation as a *bucket* that holds all the uninsured points that lie inside this triangle. Whenever a triangle is split or an edge is flipped, some old triangles are destroyed and replaced by new triangles. When this happens, lump together all

the points in the deleted triangles and re-distributed them into the new triangles. Since the number of new triangles is $O(1)$ (3 for split and 2 for flip). This process requires $O(1)$ time for each point that is re-bucketed.

To analyse the expected running time of this algorithm we need to bound two quantities: (1) how many structural changes are made in the triangulation on average with the addition of each new point, which is $O(1)$ by randomised insertion. (2) How many effort is spent in re-bucketing points. Next we argue that the total expected time spent in re-bucketing points is $O(n \log n)$.

To do this, we will show that the expected number if times that any point is re-bucketed is $O(\log n)$. Again this is done by backward analysis. Let's fix a point $\mathbf{q} \in S$. Consider the case just after the insertion of $\mathbf{p}_i$, i.e., the i-th point in the sequence. If $\mathbf{q}$ is already inserted, then it is not involved in the re-bucketing process. Assume $\mathbf{q}$ is not yet inserted. Also, we assume that any of the existing points is equally likely to be the last point inserted.

We claim that the probability that $\mathbf{q}$ was re-bucketed as a result of the last insertion is at most $3/i$. To see this, let σ be the triangle containing $\mathbf{q}$ after the insertion of $\mathbf{p}_i$. All of newly created triangles are now incident to $\mathbf{p}_i$. σ will come into existence as a result of the last insertion if and only if one of its three vertices is $\mathbf{p}_i$. Since every three vertices of σ is equally likely to be $\mathbf{p}_i$, it follows that the probability that σ came into existence if $3/i$. Thus the probability that $\mathbf{q}$ required re0bucketing after inserting $\mathbf{p}_i$ is $3/i$.

After inserting of ith points $\mathbf{p}_i$, there are still $n - i$ points to be inserted. Hence $\mathbf{q}$ might be re-bucketed at most $n - i$ times, and each has probability $3/i$ of being re-bucketed. Thus the expected number of points that require re-bucketing as part of the last insertion is at most $(n-i)3/i$. By the linearity of expectation, the total number of re-bucketing is

$$\sum_{i=1}^{n} \frac{3}{i}(n-i) \leq \sum_{i=1}^{n} \frac{3}{i} n = 3n \sum_{i=1}^{n} \frac{1}{i} = 3n \ln n + O(1).$$

Thus the total expected time spent in re-bucketing n points $O(n \log n)$, as desired.

Hence the randomised incremental flip algorithm constructs Delaunay triangulation in expected $O(n \log n)$ time.

3.7. **Divide-and-Conquer Algorithm.** There are many Delaunay triangulation algorithms, some of which are surveyed and evaluated by Fortune [7] and Su and Drysdale [18]. Their results indicate a rough parity in speed among the incremental insertion algorithm of Lawson [11], the divide-and-conquer algorithm of Lee and Schachter [12], and the plane-sweep algorithm of Fortune [6]; As Su and Drysdale [18] also found, the divide-and-conquer algorithm is fastest, with the sweepline algorithm second. The incremental algorithm performs poorly, spending most of its time in point location. (Su and Drysdale produced a better incremental

insertion implementation by using bucketing to perform point location, but it still ranks third.

An important optimization to the divide-and-conquer algorithm, adapted from Dwyer [5], is to partition the vertices with alternating horizontal and vertical cuts (Lee and Schachter?s algorithm uses only vertical cuts). Alternating cuts speed the algorithm and, when exact arithmetic is disabled, reduce its likelihood of failure.

[5] Rex A. Dwyer. A Faster Divide-and-Conquer Algo- rithm for Constructing Delaunay Triangulations. Algo- rithmica 2(2):137?151, 1987.

[12] D. T. Lee and B. J. Schachter. Two Algorithms for Con- structing a Delaunay Triangulation. International Jour- nal of Computer and Information Sciences 9(3):219? 242, 1980.

[18] Peter Su and Robert L. Scot Drysdale. A Compari- son of Sequential Delaunay Triangulation Algorithms. Proceedings of the Eleventh Annual Symposium on Computational Geometry, pages 61?70. Association for Computing Machinery, June 1995.