Computational Geometry

Chapter 2

Basic Techniques

On the Agenda

- Line Segment Intersection
- Plane Sweep
- Euler’s Formula
Triangle Area

\[ 2 \cdot \text{Area} = \left\| (P_2 - P_1) \times (P_3 - P_1) \right\| \]
\[ = \left\| P_2 - P_1 \right\| \left\| P_3 - P_1 \right\| \sin \alpha \]
\[ = \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix} \]
\[ = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \]

- The determinant is twice the area of the triangle whose vertices are the rows of the matrix.

Triangle Orientation

\[ \text{Area} = \frac{1}{2} \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \]

- The sign of the result indicates the orientation of the vertices.
- Positive triangle \( \equiv \) counter-clockwise direction \( \equiv \) left turn.
- Negative triangle \( \equiv \) clockwise direction \( \equiv \) right turn.
Line-Segment Intersection

- **Theorem:** Segments \((p_1, p_2)\) and \((p_3, p_4)\) intersect in their interiors if and only if
  - \(p_1\) and \(p_2\) are on different sides of the line \(p_3p_4\);
  - and
  - \(p_3\) and \(p_4\) are on different sides of the line \(p_1p_2\).

- This can be checked by computing the orientations of four triangles. Which?

- **Special cases:**
  - \(< < \)
  - \(\) ( )
  - \(\) ( )
  - \(\) ( )

Computing the Intersection

\[
p(t) = p_1 + (p_2 - p_1)t \quad 0 \leq t \leq 1
\]
\[
q(s) = q_1 + (q_2 - q_1)s \quad 0 \leq s \leq 1
\]

**Question:** What is the meaning of other values of \(s\) and \(t\)?

Solve (2D) linear vector equation for \(t\) and \(s\):

\[
p(t) = q(s)
\]

check that \(t \in [0,1]\) and \(s \in [0,1]\)
Point in Polygon

- Given a polygon $P$ with $n$ sides, and a point $q$, decide whether $q \in P$.

- Solution A: Count how many times a ray from $q$ to infinity intersects the polygon. $q \in P$ if and only if this number is odd.

- Time complexity: $\Theta(n)$

- Question: Are there any special cases?

Point in Polygon (cont.)

- Solution B: Sum up the angles $\alpha_i = \angle p_ip_{i+1}q$ for $i = 0, \ldots, n-1$ ($n \equiv 0 \mod n$)

- Sum = $2\pi$ iff $q \in P$ (otherwise Sum = 0)

- $\alpha_i = \sin^{-1}\left(\frac{\text{signed}_i \text{area}(p_i, q, p_{i+1})}{\| p_i - q \| \cdot \| p_{i+1} - q \|}\right)$

- Note: Some angles are negative.

- Time complexity: $\Theta(n)$

- Question: Can the problem be solved in less time if $P$ is convex?
Plane-Sweep Paradigm

- Problem: Given \( n \) line-segments in the plane, compute all their intersections.
- Variant: Report # of intersections.
- Another variant: Is there any pair of intersecting segments?
- Assumptions:
  - No line segment is vertical.
  - No two segments overlap in more than one point.
  - No three segments intersect at a common point.
- Naive algorithm: Check each pair of segments for intersection. Complexity: \( \Theta(n^2) \) time, \( \Theta(n) \) space.

Segment-Intersection Algorithm

- An event is any endpoint or intersection point.
- Sweep the plane from left to right using a vertical line.
- Maintain two data structures:
  - Event priority queue: sorted by \( x \) coordinate.
  - Sweep-line status: stores segments currently intersected by the sweep line, sorted by \( y \) coordinate.
Basic Idea

We are able to identify all intersections by looking only at adjacent segments in the sweep line status during the sweep.

Theorem: Just before an intersection occurs (infinitesimally-close to it), the two respective segments are adjacent to each other in the sweep-line status.

In practice: Look ahead: whenever two line segments become adjacent along the sweep line, check for their intersection to the right of the sweep line.

Detailed Algorithm

- Initialization:
  - Put all segment endpoints in the event queue, sorted according to $x$ coordinates. Time: $O(n \log n)$.
  - Sweep line status is empty.

- The algorithm proceeds by inserting, deleting, and handling discrete events from the queue until it is empty.
Detailed Algorithm (cont.)

- Event of type A: Beginning of a segment (left endpoint)
  - Locate segment position in the status.
  - Insert segment into sweep line status.
  - Test for intersection to the right of the sweep line with the segments immediately above and below. Insert intersection point(s) (if found) into the event queue.

- Time complexity:
  - $n$ events, $O(\log n)$ time for each
  - $\rightarrow O(n \log n)$ in total.

---

Detailed Algorithm (cont.)

- Event of type B: End of a segment (right endpoint)
  - Locate segment position in the status.
  - Delete segment from sweep line status.
  - Test for intersection to the right of the sweep line between the segments immediately above and below. Insert intersection point (if found, and if not already in the queue) into the event queue.

- Time complexity:
  - $n$ events, $O(\log n)$ time for each
  - $\rightarrow O(n \log n)$ in total.
Detailed Algorithm (cont.)

- **Event of type C: Intersection point**
  - Report/count the point.
  - Swap the two respective line segments in the sweep-line status.
  - For the new upper segment: Test it for intersection against the segment above it in the status (if exists). Insert intersection point (if found, and if not already in the queue) into the event queue.
  - Do a similar action for the new lower segment (check against the segment below it, if any).

- **Time complexity:**
  - \( k \) such events, \( O(\log n) \) each
  - \( \rightarrow O(k \log n) \) in total.

---

**Example**

![Diagram showing sweep line status and event queue example](image)
Example (cont.)

Insert s4 to status
Test s4-s3 and s4-s2. Add e1 to event queue

s0, s1, s2, s4, s3

b1, e1, b2, b0, b3, b4

Action
Sweep Line
Status
Event Queue

Delete s1 from status
Test s0-s2. Add e2 to event queue

s0, s2, s4, s3

e1, e2, b2, b0, b3, b4

Center for Graphics and Geometric Computing, Technion
Example (cont.)

Complexity Analysis

- Basic data structures:
  - Event queue: heap
  - Sweep line status: balanced binary tree
- Each heap/tree operation requires $O(\log n)$ time. (Why is $O(\log k) = O(\log n)$?)
- Total time complexity: $O((n+k) \log n)$.
  - If $k=\Omega(n^2)$ this is slightly worse than the naive algorithm!
  - But if $k=\omega(n^2/\log n)$ then the sweep algorithm is faster.
- Note: There exists a better algorithm whose running time is $\Theta(n \log n + k)$.
- Total space complexity: $O(n+k)$.
  - Question: How can this be improved to $O(n)$?
    (Hint: Which events are [temporarily] redundant in the queue?)
Graph Definitions

\[ G = \langle V, E \rangle \]
\[ V = \text{vertices} = \{A,B,C,D,E,F,G,H,I,J,K,L\} \]
\[ E = \text{edges} = \{(A,B),(B,C),(C,D),(D,E),(E,F),(F,G), \]
\[ (G,H),(H,A),(A,J),(A,G),(B,J),(K,F), \]
\[ (C,L),(C,I),(D,I),(D,F),(F,I),(G,K), \]
\[ (J,L),(J,K),(K,L),(L,I)\}\]

Vertex degree (valence) = number of edges incident on vertex.
\[ \text{deg}(J) = 4, \text{deg}(H) = 2 \]

\[ k-\text{regular graph} = \text{graph whose vertices all have degree } k \]

A face of a planar graph is an empty cycle of vertices/edges.

Connectivity

A graph is connected if there is a path of edges connecting every two vertices.
A graph is \( k \)-connected if between every two vertices there are \( k \) edge-disjoint paths.

A graph \( G' = \langle V', E' \rangle \) is a subgraph of a graph \( G = \langle V, E \rangle \) if \( V' \) is a subset of \( V \) and \( E' \) is the subset of \( E \) incident on \( V' \).

A connected component of a graph is a maximal connected subgraph.

A subset \( V' \) of \( V \) is an independent set in \( G \) if the subgraph it induces does not contain any edges of \( E \).
Graph Embedding

A graph is embedded in $\mathbb{R}^d$ if each vertex is assigned a position in $\mathbb{R}^d$.

Embedding in $\mathbb{R}^2$  
Embedding in $\mathbb{R}^3$

Planar Graphs

A planar graph is a graph whose vertices and edges can be embedded in $\mathbb{R}^2$ such that its edges do not intersect.

Theorem: Every planar graph can be drawn as a straight-line plane graph.
### Triangulation

A triangulation of a point set is a straight-line plane graph whose (finite) faces are all triangles. (Triangulation of the CH of the set.)

The Delaunay triangulation of a set of points is the unique set of triangles such that the circumcircle of any triangle does not contain any other point. The Delaunay triangulation avoids long and skinny triangles.

### Meshes

A mesh is a straight-line graph embedded in $\mathbb{R}^3$.

- **Boundary edge**: adjacent to exactly one face.
- **Regular edge**: adjacent to exactly two faces.
- **Singular edge**: adjacent to more than two faces.

- **Closed mesh**: mesh with no boundary edges.
- **Manifold mesh**: mesh with no singular edges.

**Corners** $\subseteq V \times F$

**Half-edges** $\subseteq E \times F$
Planar Graphs and Meshes

Every manifold mesh is planar!!

Center for Graphics and Geometric Computing, Technion

Topology

The genus of a graph is half of the maximal number of closed paths that do not disconnect the graph (the number of "holes").

Genus(sphere) = 0
Genus(torus) = 1

Euler-Poincaré Formula

For a planar graph:
\[ v + f - e = 2(c - g) - b \]

v = # vertices \hspace{1cm} c = # conn. comp.
f = # faces \hspace{1cm} g = genus
e = # edges \hspace{1cm} b = # boundaries

Center for Graphics and Geometric Computing, Technion
Examples

- Genus 0
- Genus 1
- Genus 2

Exercises

**Theorem:** In a closed manifold triangle mesh, the average vertex degree is ~6.

**Proof:** In such a mesh, \( f = 2e/3 \).

By Euler’s formula: \( v + 2e/3 - e = 2 - 2g \) hence \( e = 3(v - 2 + 2g) \) and \( f = 2(v - 2 + 2g) \).

So \( \text{Average}(\text{deg}) = 2e/v = 6(v - 2 + 2g)/v \)

\(~ 6 \) for large \( v \).

**Corollary:** Only a toroidal \((g=1)\) closed manifold triangle mesh can be regular (all vertex degrees are 6).

**Proof:** In a regular mesh the average degree is exactly 6. This can happen only if \( g = 1 \).

Does Euler’s theorem imply that any planar graph has an independent set of size at least \( \frac{1}{4} n \)?
Euler’s Formula

- For a connected planar graph with $E$ edges, $V$ vertices, and $F$ faces, the following relation holds:

$$V - E + F = 2$$

The Linearity Relation

- **Theorem**: In a planar graph, $E = O(V)$ and $F = O(V)$.
- **Proof**:
  - We may assume that the graph is maximally triangulated (this may only increase $E$ and $F$).
  - Every face is bounded by 3 half-edges $\Rightarrow 3F = 2E \Rightarrow E = 3F/2$
  - By Euler’s formula: $V - E + F = 2 \Rightarrow V - 3F/2 + F = 2 \Rightarrow F = 2(V - 2) = O(V)$
  - Similarly, $F = 2E/3 \Rightarrow V - E + 2E/3 = 2 \Rightarrow E = 3(V - 2) = O(V)$