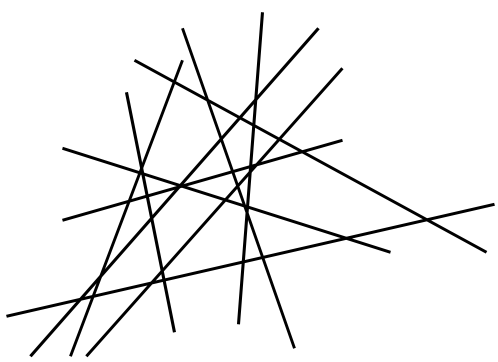
Computational Geometry

Chapter 9

Line Arrangements

On the Agenda

- ❑ Line Arrangements
- ❑ Applications



Complexity of a Line Arrangement

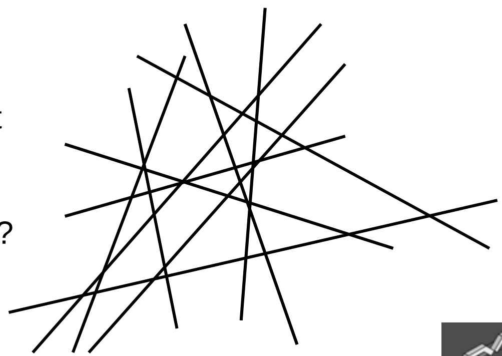
- Given a set L of n lines in the plane, their *arrangement* $A(L)$ is the plane subdivision induced by L.
- **Theorem**: The combinatorial complexity of the arrangement of n lines is $\Theta(n^2)$ in the worst case.
- Proof:
  - Number of vertices $\leq \binom{n}{2} = \frac{n^2}{2} - \frac{n}{2}$ (each pair of different lines may intersect at most once).
  - Number of edges $\leq n^2$ (each line is cut into at most n pieces by at most n-1 other lines).
  - Number of faces $\leq \frac{n^2}{2} + \frac{n}{2} + 1$ (by Euler's formula and connecting all rays to a point at infinity).

  Equalities hold for lines in general position. (Show!)

Computing a Line Arrangement

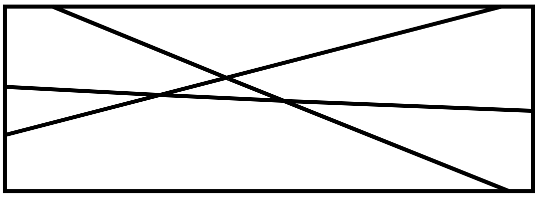
- Goal: Compute this planar map (as a DCEL).
- A plane-sweep algorithm would require $\Theta(n^2 \log n)$ time (after finding the leftmost event*): $\Theta(n^2)$ events, $\Theta(\log n)$ time each.

(*) Question:
How can the leftmost event be found in $O(n \log n)$ time instead of $O(n^2)$ time?



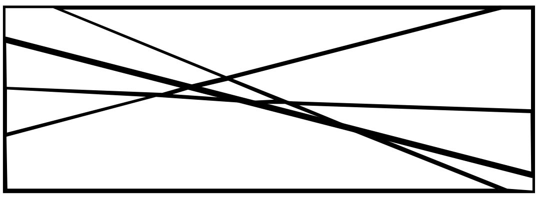
An Incremental Algorithm

- **Input:** A set *L* of *n* lines in the plane.
- **Output:** The DCEL structure for the arrangement *A*(*L*), i.e., the subdivision induced by *L* in a bounding box *B*(*L*) that contains all the intersections of lines in *L*.
- The algorithm:
  - Compute a bounding box *B*(*L*), and initialize the DCEL.
  - Insert one line after another. For each line, locate the entry face, and update the arrangement, face by face, along the path of faces ("zone") traversed by the line.



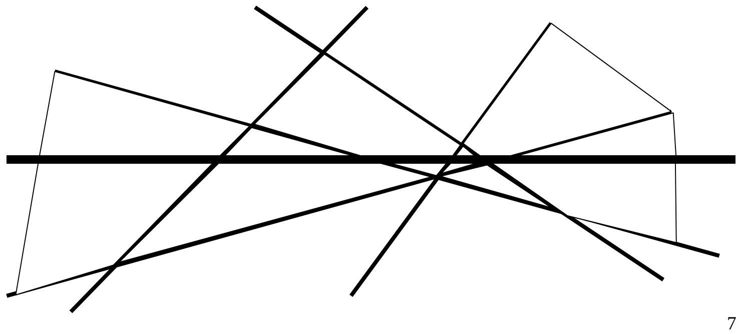
Line Arrangement Algorithm (cont.)

- After inserting the *i*th line, the complexity of the map is $O(i^2)$. ($\Theta(i^2)$ in the worst case—general position.)
- The time complexity of each insertion of a line depends on the complexity of its *zone.*



Zone of a Line

- The *zone* of a line *ℓ* in the arrangement *A*(*L*) is the set of faces of *A*(*L*) intersected by *ℓ*.
- The complexity of a zone is the total complexity of all its faces: the total number of edges (or vertices) of these faces.

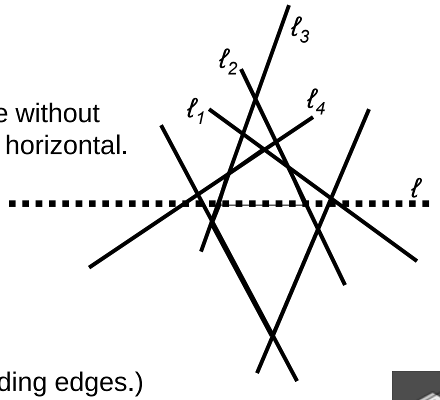


Center for Graphics and Geometric Computing, Technion

7

The Zone Theorem

- **Theorem:** In an arrangement of *n* lines, the complexity of the zone of a line is O(*n*).
- **Proof (sketch):**
  - Consider a line *ℓ*. Assume without loss of generality that *ℓ* is horizontal.
  - Assume first that there are no horizontal lines.
  - Count the number of *left bounding edges* in the zone, and prove that this is at most 4*n*. (Same idea for right bounding edges.)

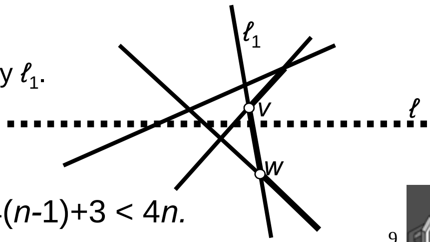


Center for Graphics and Geometric Computing, Technion

8

Zone Complexity: Proof

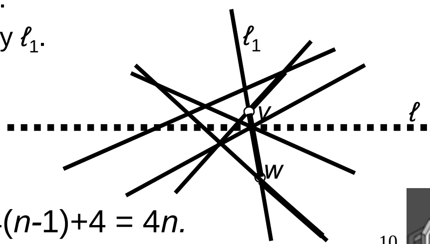
- By induction on n.
- For n=1: Trivial.
- For n>1:
  - Let ℓ_1 be the rightmost line intersecting ℓ (assume it's unique).
  - By the induction hypothesis, the zone of ℓ in $A(L \setminus \{\ell_1\})$ has at most $4(n-1)$ left bounding edges.
  - When adding ℓ_1, the number of such edges increases:
    - One new edge on ℓ_1.
    - Two old edges split by ℓ_1.



Hence, the new zone complexity is at most $4(n-1)+3 < 4n$.

Zone Complexity: Proof (cont.)

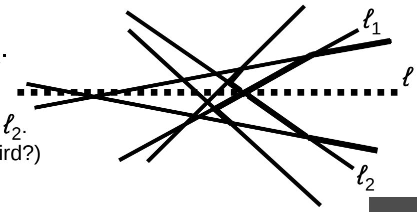
- What happens if several (>2) lines intersect ℓ in the rightmost intersection points (i.e., if ℓ_1 is not unique)?
  - Pick ℓ_1 randomly out of these lines.
  - By the induction hypothesis, the zone of ℓ in $A(L \setminus \{\ell_1\})$ has at most $4(n-1)$ left bounding edges.
  - When adding ℓ_1, the number of such edges increases:
    - Two new edges on ℓ_1.
    - Two old edges split by ℓ_1.



Hence, the new zone complexity is at most $4(n-1)+4 = 4n$.

Zone Complexity: Proof (cont.)

- And what happens if exactly 2 lines, ℓ_1 and ℓ_2, intersect ℓ in the rightmost intersection points?
  - Discard both lines.
  - By the induction hypothesis, the zone of ℓ in $A(L \setminus \{\ell_1,\ell_2\})$ has at most $4(n-2)$ left bounding edges.
  - When adding ℓ_1, the number of such edges increases by 3.
  - When adding ℓ_2, the number of such edges increases by 5.
    - One new edge on ℓ_1.
    - Two old edges split by ℓ_1.
    - Two new edges on ℓ_2.
    - Three old edges split by ℓ_2.
      (Two are seen; where is the third?)



Hence, the new zone complexity is at most $4(n-2)+8 = 4n$.

Center for Graphics and Geometric Computing, Technion 11

Zone Complexity: Proof (cont.)

- And what if there are horizontal lines?
- If these lines are parallel to ℓ, then just (imaginarily) rotate them; this will only **increase** the complexity of the zone of ℓ.
- If there is a line ℓ_0 identical to ℓ, then the complexity of the zone of ℓ is equal to that of the zone of ℓ_0.
- If there are several lines identical to ℓ, their multiplicity does not increase the complexity of the zone of ℓ.

Center for Graphics and Geometric Computing, Technion 12

Constructing the Arrangement

- The time required to insert a line ℓ_i is linear in the complexity of its zone, which is linear in the number of the already existing lines. Therefore, the total time is

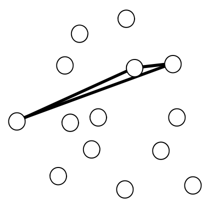
$$O(n^2) \quad + \sum_{i=1}^{n}\big(O(\log i) \quad + O(i)\big) \quad = O(n^2).$$

Finding a bounding box (can be done in O(*n* log *n*))	Finding the entry point (bin. search)	According to the Zone Theorem

- Note: The bound does not depend on the line-insertion order! (All orders are good.)

Application 1: Minimum-Area Triangle

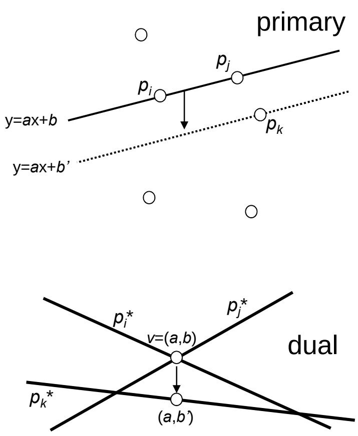
- Given a set of *n* points, determine the three points that form the triangle of minimum area.*
- Easy to solve in $\Theta(n^3)$ time, but not so easy to solve in $O(n^2)$ time.
- May be solved in $\Theta(n^2)$ time using the line arrangement in the dual plane.



(*) Finding the specific set of *n* points that **maximizes** the area of the minimum-area triangle, or, at least, determining what this area is, is the famous *Heilbronn's triangle problem*.

An $\Theta(n^2)$-Time Algorithm

- Construct the arrangement of dual lines in $\Theta(n^2)$ time.
- For each pair of points p_i and p_j (assume $p_i p_j$ is the triangle base):
  - Identify the vertex *v* in the dual arrangement, corresponding to the line through these points.
  - Find the line of the arrangement that is vertically closest to *v*.
  - Remember the best line so far.
- Output point corresponding to the best dual line.
- Questions:
  Why is it easier to find p_k^* than p_k?
  Why is it OK to look vertically?
  Why is the total running time only $\Theta(n^2)$?

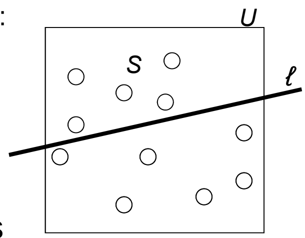


Application 2: Discrepancy

- Given a set *S* of *n* points in the unit square $U=[0,1]^2$.
- For a given line ℓ, how many points lie below ℓ?
  - Let *h* be the halfplane below ℓ.
  - If the points are well distributed, this number should be close to $\mu(h)\cdot n$, where $\mu(h) = |U \cap h|$. Define $\mu_S(h) = |S \cap h|/|S|$.
  - The *discrepancy* of *S* with respect to *h* is:

    $$\Delta_S(h) = |\mu(h) - \mu_S(h)|$$
- The *halfplane discrepancy* of *S* is

  $$\Delta(S) = \sup_h \Delta_S(h)$$

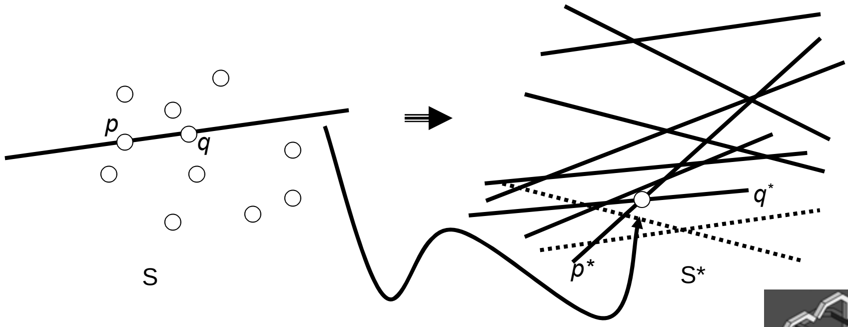


Observation: To compute $\Delta(S)$, it suffices to consider halfplanes that pass through pairs of points.

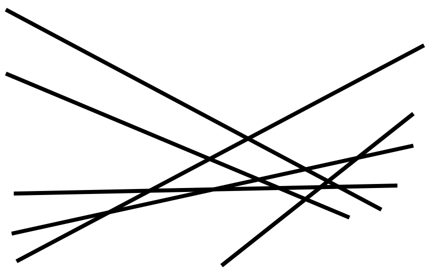
Naive algorithm (all pairs): $\Theta(n^3)$ time.

Computing Discrepancy

- ❑ In the dual plane, this is equivalent to counting the number of dual lines *above* the dual point.



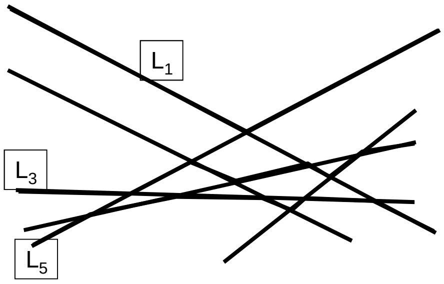
Computing Discrepancy (cont.)



- ❑ For every vertex in $A(S^*)$, compute the number of lines above it, passing through it (2 in general position), or lying below it.
- ❑ These three numbers sum up to n, so it suffices to compute only two of them.
- ❑ From the DCEL structure we know how many lines pass through each vertex.

Levels of an Arrangement

- A point is at *level k* in an arrangement of *n* lines if there are at most *k*-1 lines above this point and at most *n*-*k* lines below this point.
- There are *n* levels in an arrangement of *n* lines.
- A vertex can be on multiple levels, depending on the number of lines passing through it.
- (Sometimes levels are counted from 0 instead of 1.)



An $\Theta(n^2)$-Time Algorithm

- Construct the dual arrangement.
- For each line, compute the levels of all its vertices:
  1. Compute the levels of the left infinite rays by sorting slopes. O(*n* log *n*) time.
  2. Traverse all the lines from left to right, incrementing or decrementing the level, depending on the direction (slope) of the crossing line. $\Theta(n)$ time for each line.
- Total: $\Theta(n^2)$ time.