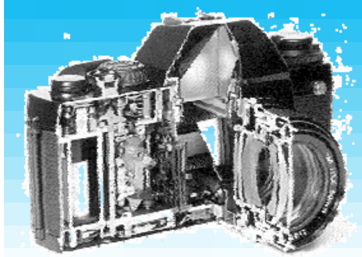


Computational Geometry
Lecture 7

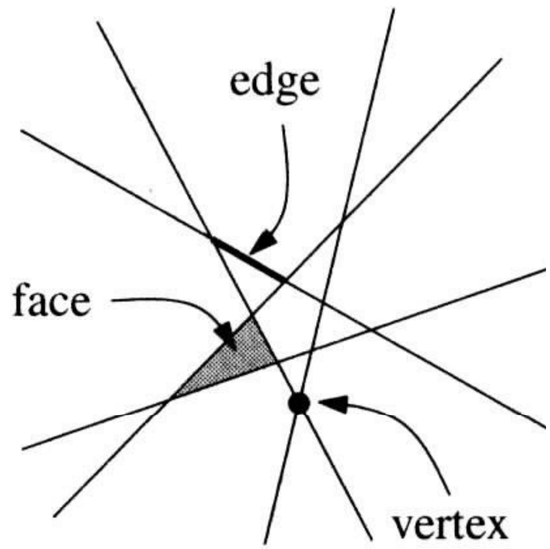
Jingyi Yu

Computer and Information Science

University of Delaware

New Concept: Arrangements of Lines

- L is a set of n lines in the plane.
- L induces a subdivision of the plane that consists of vertices, edges, and faces.
- **This is called the *arrangement* induced by L, denoted $A(L)$**



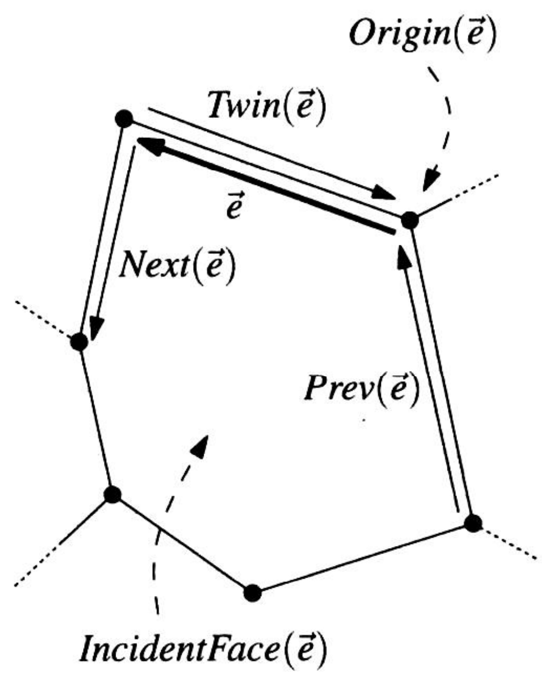
- The *complexity* of an arrangement is the total number of vertices, edges, and faces.

Arrangments

- Number of vertices of $A(L) \leq \binom{n}{2}$
  - Vertices of $A(L)$ are intersections of $l_i, l_j \in L$
- Number of edges of $A(L) \leq n^2$
  - Number of edges on a single line in $A(L)$ is one more than number of vertices on that line.
- Number of faces of $A(L) \leq \frac{n^2}{2} + \frac{n}{2} + 1$
- Inductive reasoning: add lines one by one
  Each edge of new line splits a face. → $1 + \sum_{i=1}^{n} i$
- Total complexity of an arrangement is $O(n^2)$

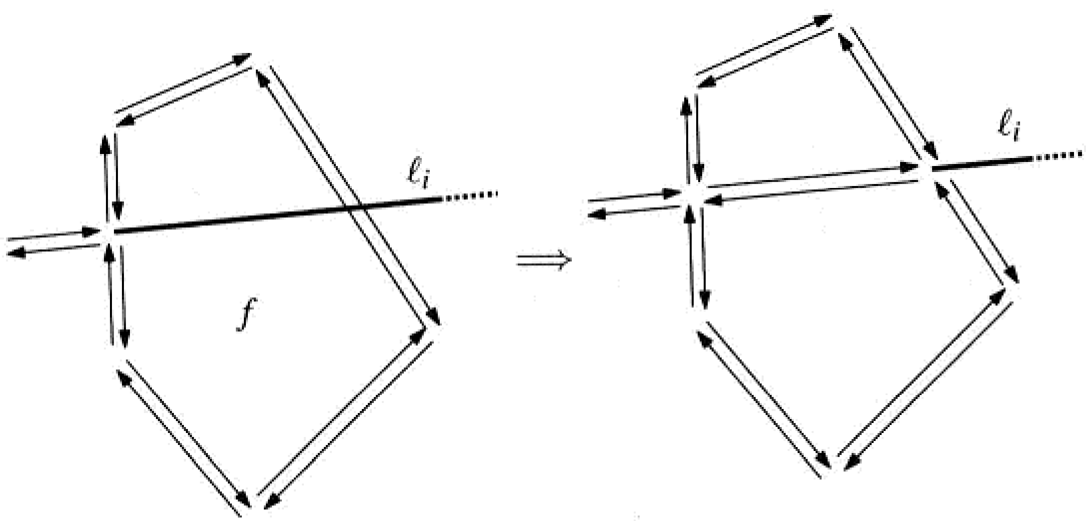
How Do We Store an Arrangement?

- Data Type: doubly-connected edge-list (DCEL)
  - Vertex:
    - Coordinates, Incident Edge
  - Face:
    - an Edge
  - Half-Edges
    - Origin Vertex
    - Twin Edge
    - Incident Face
    - Next Edge, Prev Edge



Building the Arrangement

- Iterative algorithm: put one line in at a time.
- Start with the first edge e that l_i intersects.
- Split that edge, and move to *Twin*(e)



ConstructArrangement Algorithm

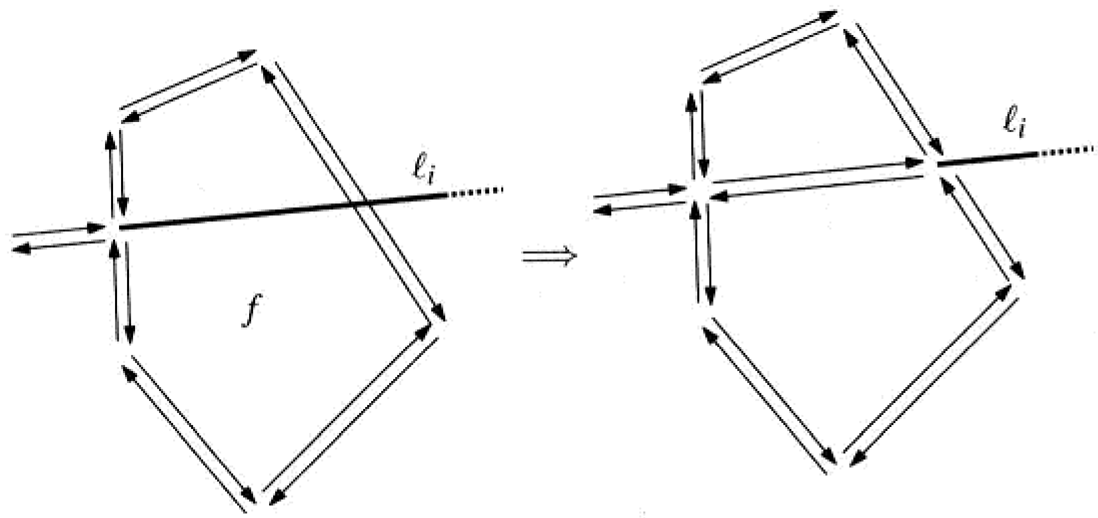
Input: A set L of n lines in the plane

Output: DCEL for the subdivision induced by the part of $A(L)$ inside a bounding box

1. Compute a bounding box $B(L)$ that contains all vertices of $A(L)$ in its interior
2. Construct the DCEL for the subdivision induced by $B(L)$
3. **for** i=1 to n **do**
4. Find the edge e on $B(L)$ that contains the leftmost intersection point of l_i and A_i
5. f = the bounded face incident to e
6. **while** f is not the face outside $B(L)$ **do**
7. Split f, and set f to be the next intersected face

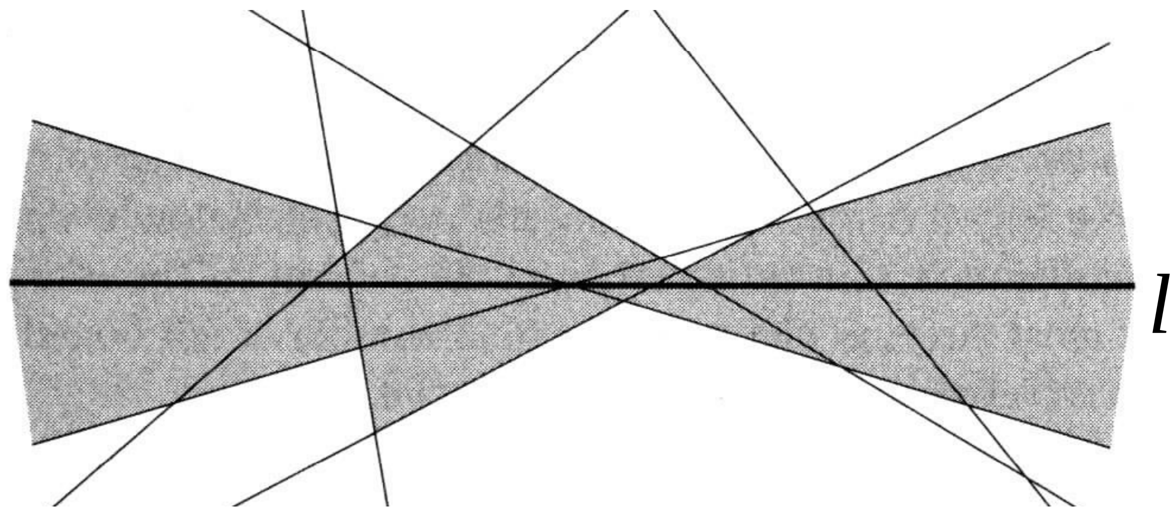
ConstructArrangement Algorithm -Running Time-

- We need to insert n lines.
- Each line splits $O(n)$ edges.
- We may need to traverse $O(n)$ $Next(e)$ pointers to find the next edge to split.



Zones

- The *zone* of a line l in an arrangement $A(L)$ is the set of faces of $A(L)$ whose closure intersects l.



- Note how this relates to the complexity of inserting a line into a DCEL…

Zone Complexity

- The complexity of a zone is defined as the total complexity of all the faces it consists of, i.e. the sum of the number of edges and vertices of those faces.
- The time it takes to insert line l_i into a DCEL is linear in the complexity of the zone of l_i in $A(\{l_1,\ldots,l_{i-1}\})$.

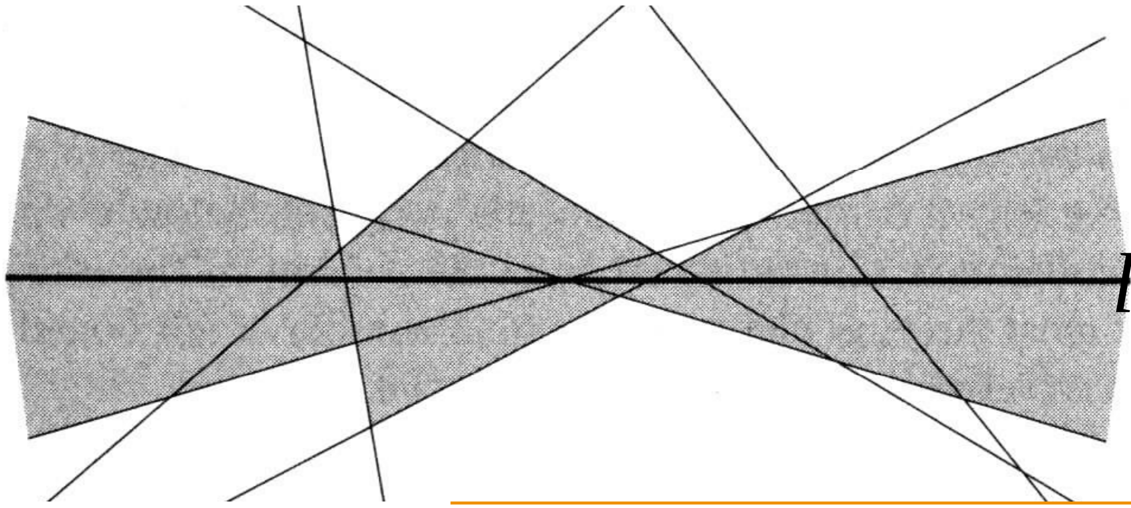
Zone Theorem

- The complexity of the zone of a line in an arrangement of m lines on the plane is $O(m)$

- We can insert a line into an arrangement in linear time.
- **We can build an arrangement in $O(n^2)$ time.**

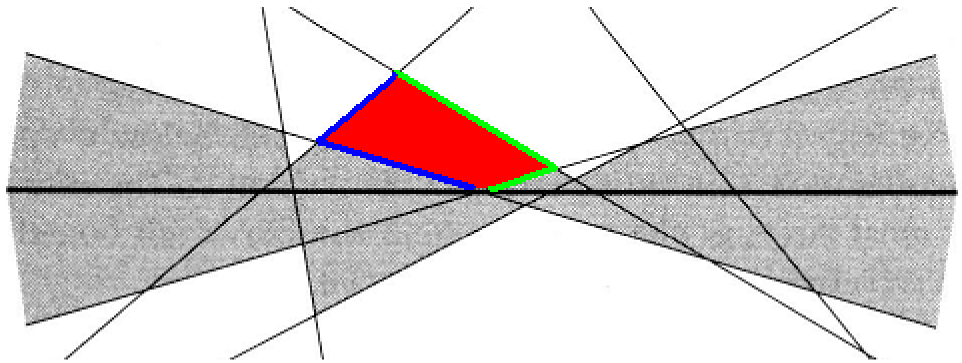
Proof of Zone Theorem

- Given an arrangement of *m* lines, *A*(*L*), and a line *l*.
- Change coordinate system so *l* is the x-axis.
- Assume (for now) no horizontal lines



Proof of Zone Theorem

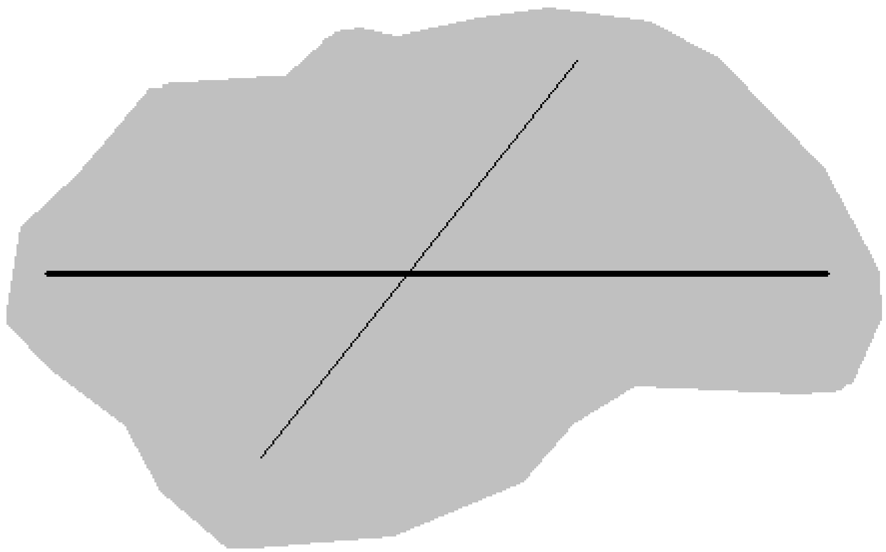
- Each edge in the zone of *l* is a *left bounding edge* and a *right bounding edge*.



- Claim: number of left bounding edges ≤ 5*m*
- Same for number of right bounding edges
  → Total complexity of *zone*(*l*) is linear

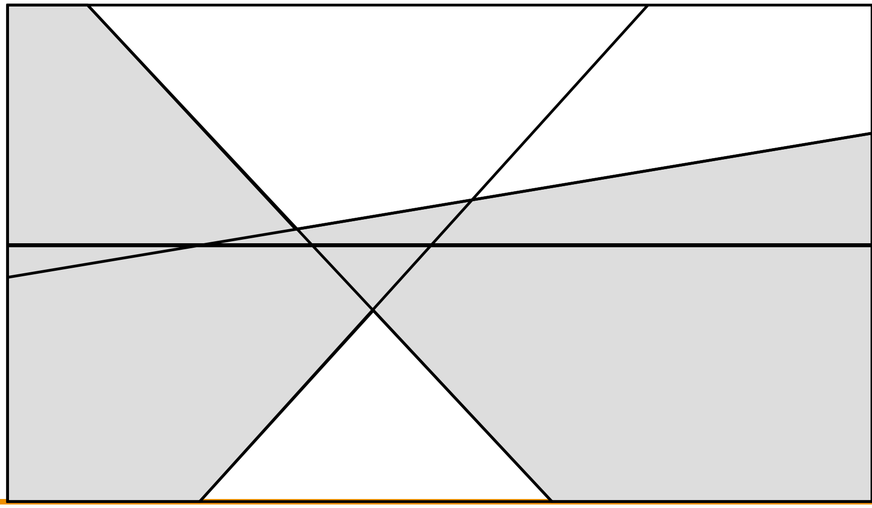
Proof of Zone Theorem -Base Case-

- When m=1, this is trivially true.
  (1 left bounding edge $\leq$ 5)



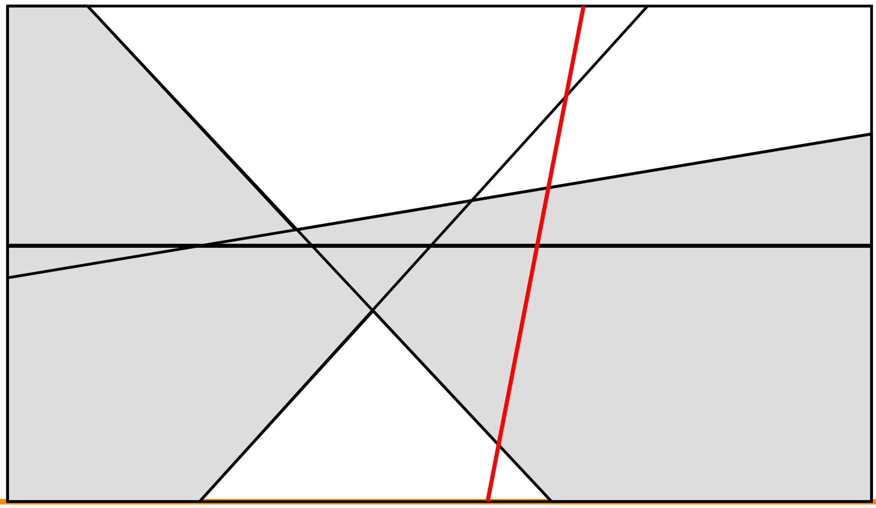
Proof of Zone Theorem -Inductive Case-

- Assume true for all but the rightmost line l_r: i.e. Zone of l in $A(L\text{-}\{l_r\})$ has at most $5(m\text{-}1)$ left bounding edges
- Assuming no other line intersects l at the same point as l_r , add l_r



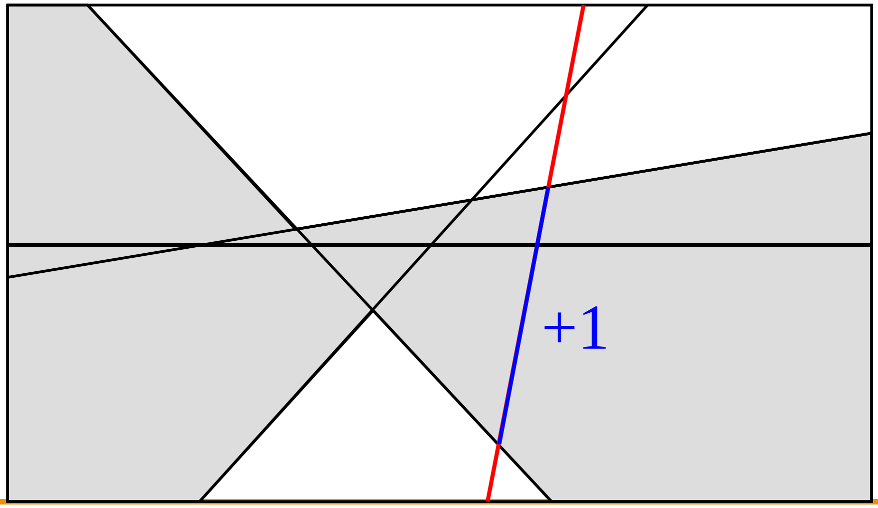
Proof of Zone Theorem -Inductive Case-

- Assume true for all but the rightmost line l_r: i.e. Zone of l in $A(L\text{-}\{l_r\})$ has at most $5(m\text{-}1)$ left bounding edges
- Assuming no other line intersects l at the same point as l_r , add l_r



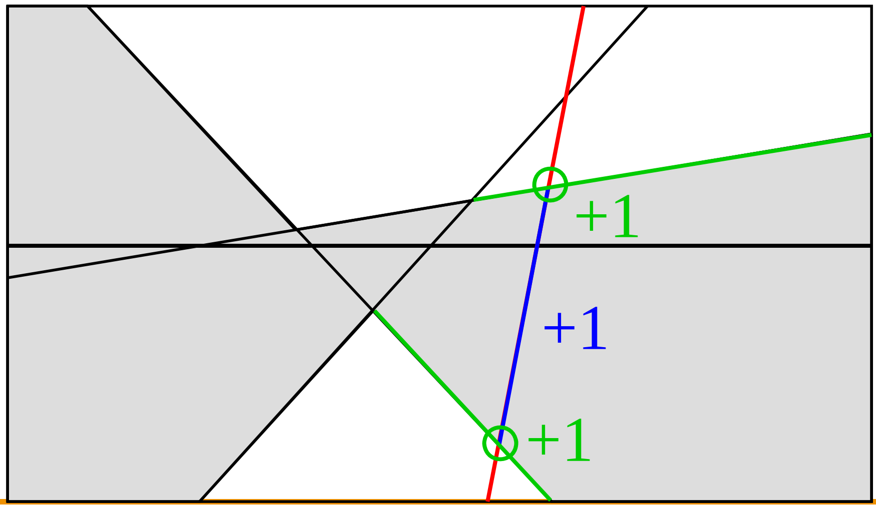
Proof of Zone Theorem -Inductive Case-

- Assume true for all but the rightmost line l_r: i.e. Zone of l in $A(L\text{-}\{l_r\})$ has at most $5(m\text{-}1)$ left bounding edges
- Assuming no other line intersects l at the same point as l_r , add l_r
  - l_r has one left bounding edge with l (+1)



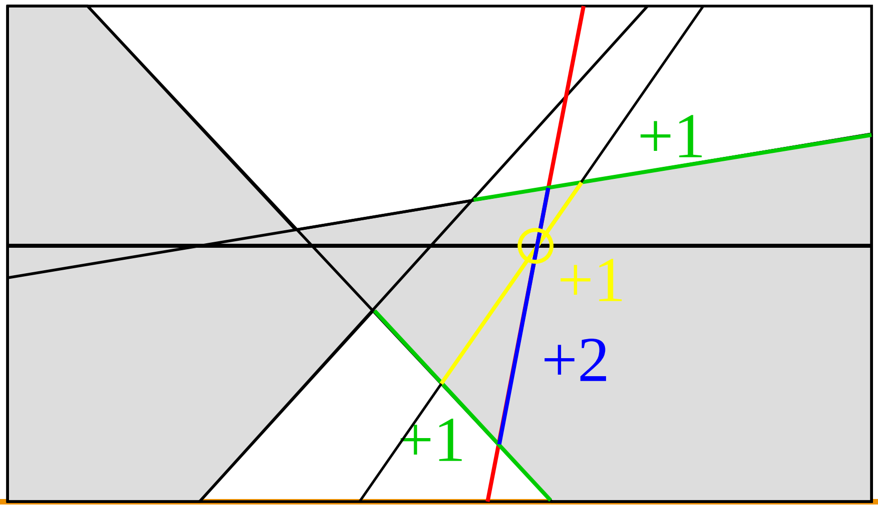
Proof of Zone Theorem -Inductive Case-

- Assume true for all but the rightmost line l_r: i.e. Zone of l in $A(L\text{-}\{l_r\})$ has at most $5(m\text{-}1)$ left bounding edges
- Assuming no other line intersects l at the same point as l_r , add l_r
  - l_r has one left bounding edge with l (+1)
  - l_r splits at most two left bounding edges (+2)



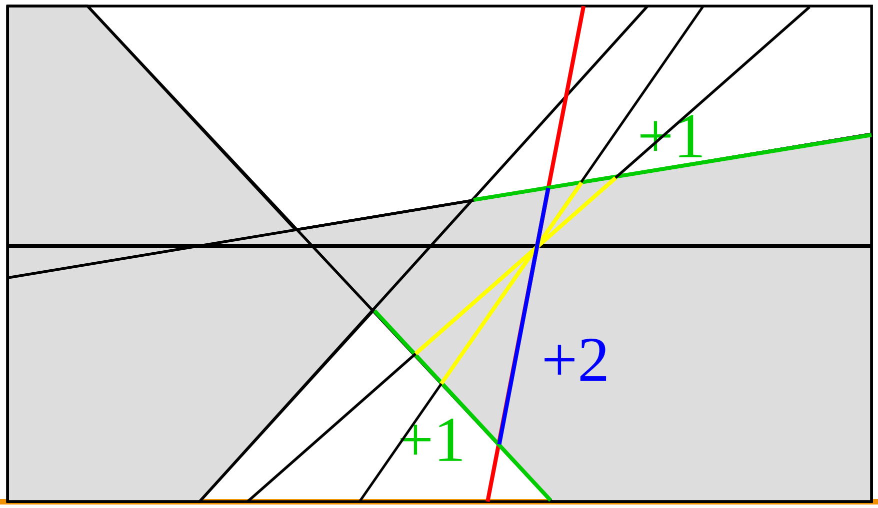
Proof of Zone Theorem
Loosening Assumptions

- What if l_r intersects l at the same point as another line, l_i does?
  - l_r has two left bounding edges (+2)
  - l_i is split into two left bounding edges (+1)
  - As in simpler case, l_r splits two other left bounding edges (+2)



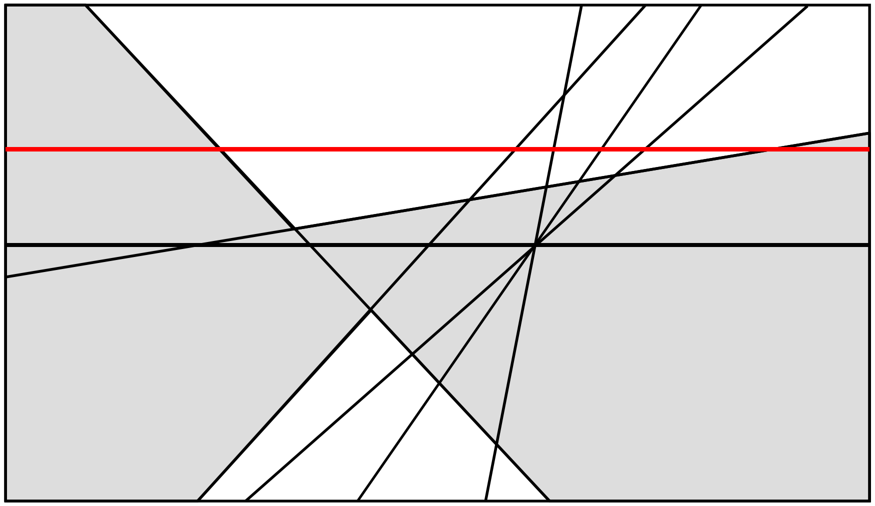
Proof of Zone Theorem
Loosening Assumptions

- What if l_r intersects l at the same point as another line, l_i does? (+5)
- What if >2 lines (l_i, l_j, …) intersect l at the same point?
  - Like above, but l_i, l_j, … are already split in two (+4)



Proof of Zone Theorem -Loosening Assumptions-

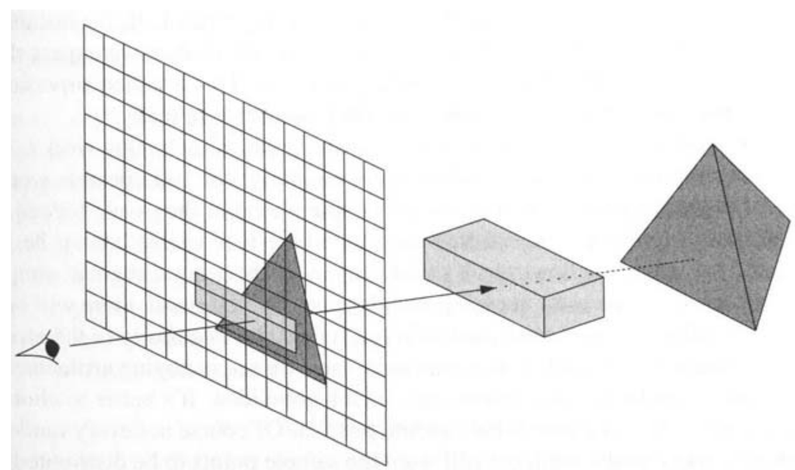
- What if there are horizontal lines in *L*?
- A horizontal line introduces *less* complexity into A(L) than a non-horizontal line.



Done proving the Zone Theorem

Ray-Tracing

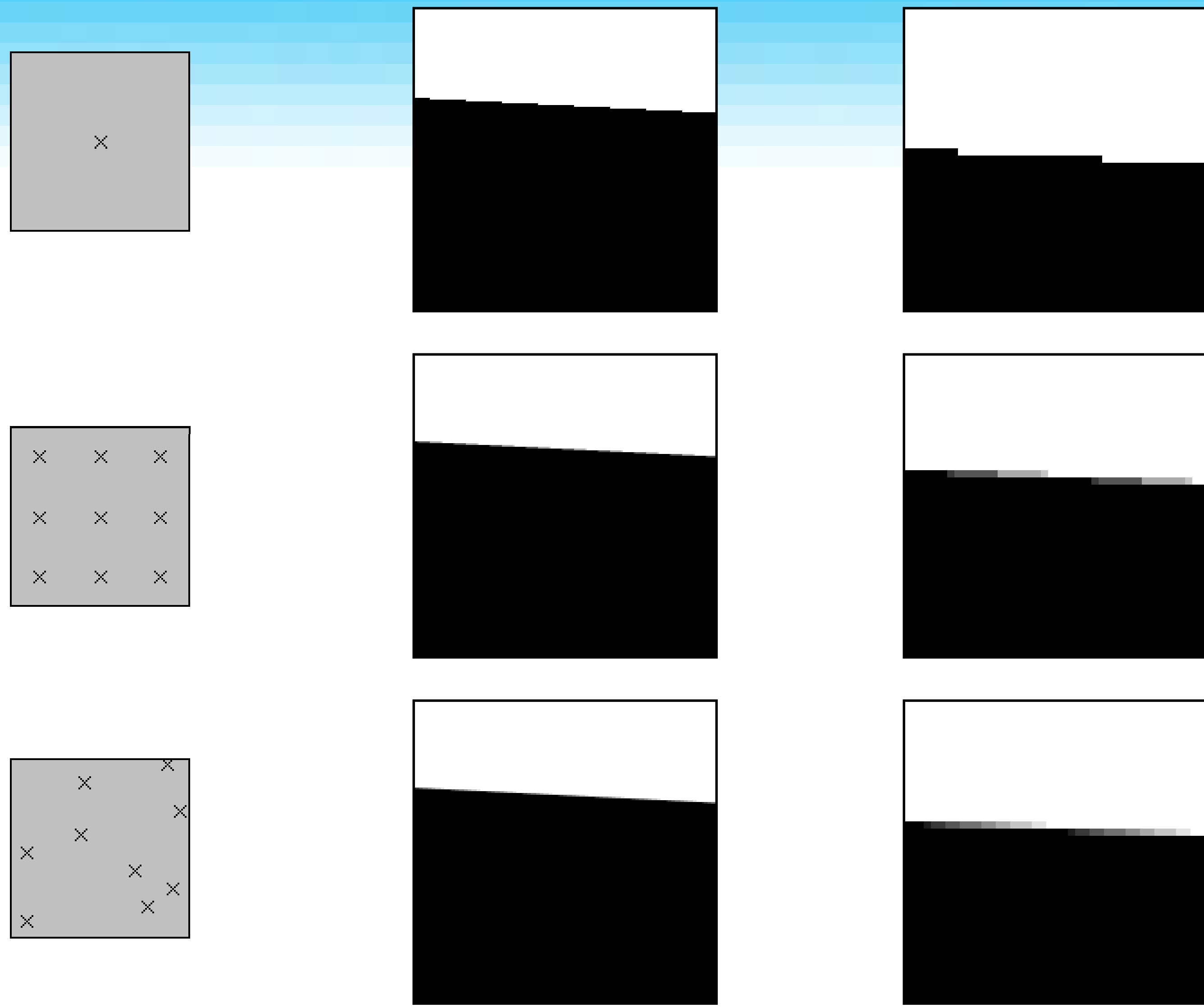
- Render a scene by shooting a ray from the viewer through each pixel in the scene, and determining what object it hits.
- Straight lines will have visible jaggies.
- We need to supersample



Supersampling

- We shoot many rays through each pixel and average the results.
- How should we distribute the rays over the pixel? Regularly?
- Distributing rays regularly isn't such a good idea. Small per-pixel error, but regularity in error across rows and columns. (Human vision is sensitive to this.)

Sample Point Set | Rendered Half-Plane | (4x zoom)



Wow…that really makes a difference!

Supersampling

- We need to choose our sample points in a somewhat random fashion.
- Finding the ideal distribution of *n* sample points in the pixel is a very difficult mathematical problem.
- Instead we'll generate several random samplings and measure which one is best.
- How do we measure how good a distribution is?

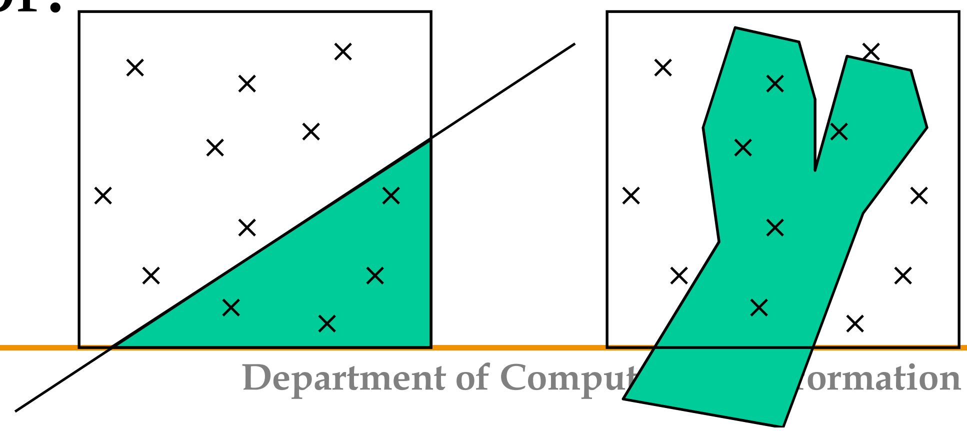
Big Picture

- To ray-trace a pixel realistically, we need pick a good distribution of sample points in the pixel.
- We need to be able to determine how good a distribution of sample points is.

How do we do this?

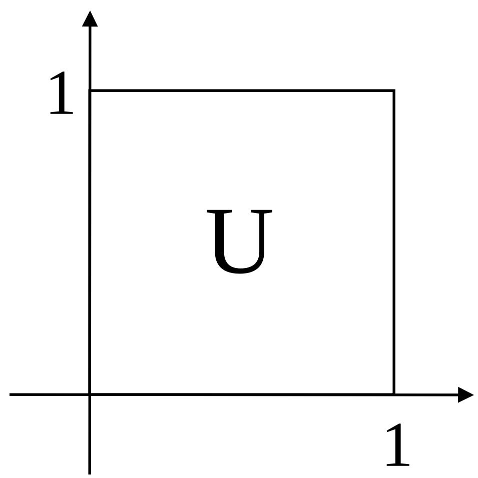
Discrepancy

- We want to calculate the discrepancy of a distribution of sample points relative to possible scenes.
- Assume all objects project onto our screen as polygons.
- We're really only interested in the simplest case: more complex cases don't exhibit regularity of error.



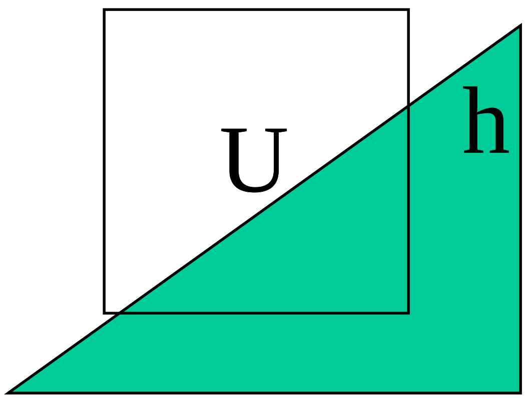
Discrepancy

- Pixel: Unit square $U = [0:1] \times [0:1]$



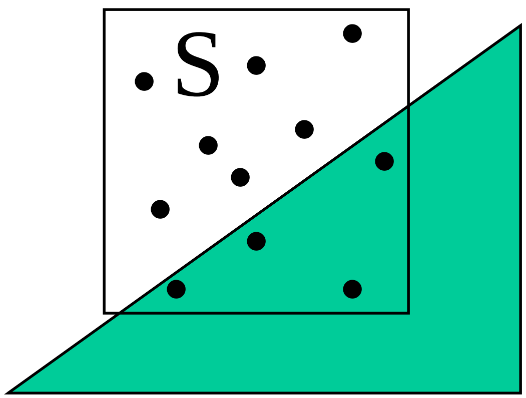
Discrepancy

- Pixel: Unit square U = [0:1] x [0:1]
- Scene: H = (infinite) set of all possible half-planes h.



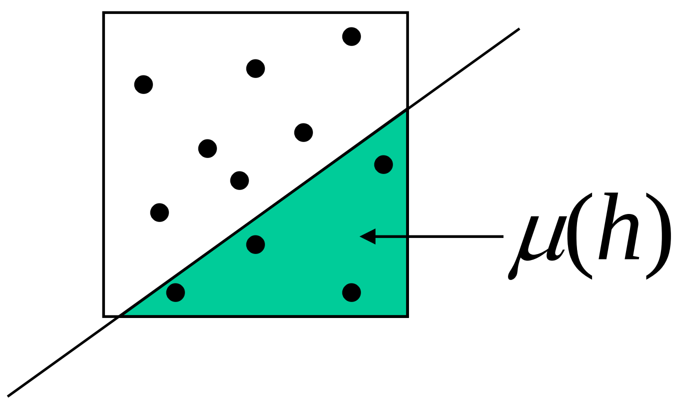
Discrepancy

- Pixel: Unit square U = [0:1] x [0:1]
- Scene: H = set of all possible half-planes h.
- Distribution of sample points: set S



Discrepancy

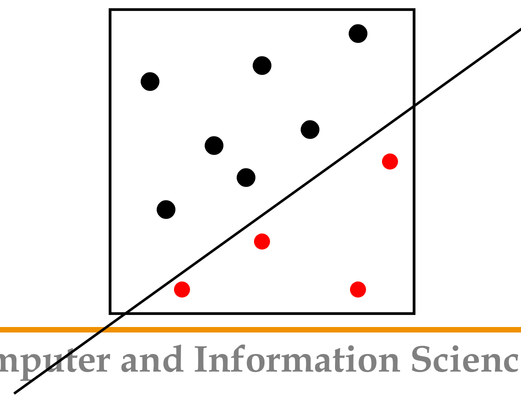
- Pixel: Unit square U = [0:1] x [0:1]
- Scene: H = set of all possible half-planes h.
- Distribution of sample points: set S
- Continuous Measure: $\mu(h)$ = area of $h \cap U$



Discrepancy

- Pixel: Unit square U = [0:1] x [0:1]
- Scene: H = set of all possible half-planes h.
- Distribution of sample points: set S
- Continuous Measure: $\mu(h)$ = area of $h \cap U$
- Discrete Measure:

  $\mu_S(h) = \text{card}(S \cap h) / \text{card}(S)$



Discrepancy

- Pixel: Unit square U = [0:1] x [0:1]
- Scene: H = set of all possible half-planes h.
- Distribution of sample points: set S
- Continuous Measure: $\mu(h)$ = area of $h \cap U$
- Discrete Measure:
  $\mu_S(h) = \text{card}(S \cap h) / \text{card}(S)$
- Discrepancy of h wrt S:
  $\Delta_S(h) = | \mu(h) - \mu_S(h) |$

Discrepancy

- Pixel: Unit square U = [0:1] x [0:1]
- Scene: H = set of all possible half-planes h.
- Distribution of sample points: set S
- Continuous Measure: $\mu(h)$ = area of $h \cap U$
- Discrete Measure: $\mu_S(h) = \mathrm{card}(S \cap h) / \mathrm{card}(S)$
- Discrepancy of h wrt S: $\Delta_S(h) = | \mu(h) - \mu_S(h) |$
- Half-plane discrepancy of S :

$$\Delta_H(S) = \max_{\text{all } h} \Delta_S(h)$$

Big Picture

We've defined the discrepancy of a chosen set of sample points with respect to all possible scenes as:

$$\Delta_H(S) = \max_{\text{all } h} \Delta_S(h)$$

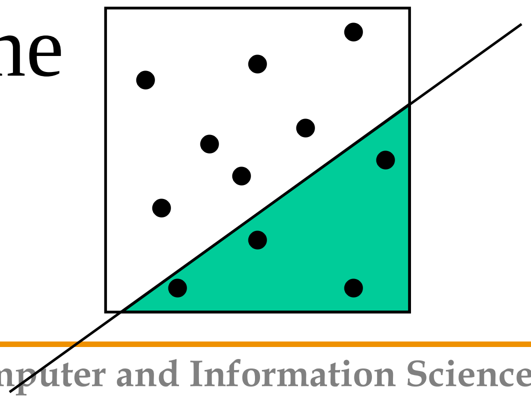
We want to pick S to minimize $\Delta_H(S)$

Computing the Discrepancy

- $\Delta_H(S) = \max_{\text{all } h} \Delta_S(h)$
- There are an infinite number of possible half-planes…We can't just loop over all of them.

Computing the Discrepancy

- $\Delta_H(S) = \max_{\text{all } h} \Delta_S(h)$
- There are an infinite number of possible half-planes…We can't just loop over all of them.
- But…the half-plane of maximum discrepancy must pass through one of the sample points.



Computing the Discrepancy

- The half-plane of maximum discrepancy must pass through at least one sample point.
- It may pass through exactly one point
  - The maximum discrepancy must be at a local extremum of the continuous measure.
  - There are an infinite number of h through point p, but only $O(1)$ of them are local extrema.
  - We can calculate the discrepancies of all n points vs $O(1)$ h each, in $O(n^2)$ time.

Computing the Discrepancy

- The half-plane of maximum discrepancy must pass through at least one sample point.
- It may pass through exactly one point
- Or it may pass through two points
  - There are $O(n^2)$ possible point pairs.
  - We need some new techniques if we want to be able to compute the discrepancy in $O(n^2)$ time.

Big Picture

We've defined the discrepancy of a chosen set of sample points with respect to all possible scenes, $\Delta_H(S)$.

We want to pick S to minimize $\Delta_H(S)$.

We need a way to compute $O(n^2)$ discrete measures to find values of $\Delta_S(h)$.

We want to do this in $O(n^2)$ time.

New Concept: Duality

- The concept: we can map between different ways of interpreting 2D values.
- Points (x,y) can be mapped in a one-to-one manner to lines (slope,intercept) in a different space.
- There are different ways to do this, called *duality transforms*.

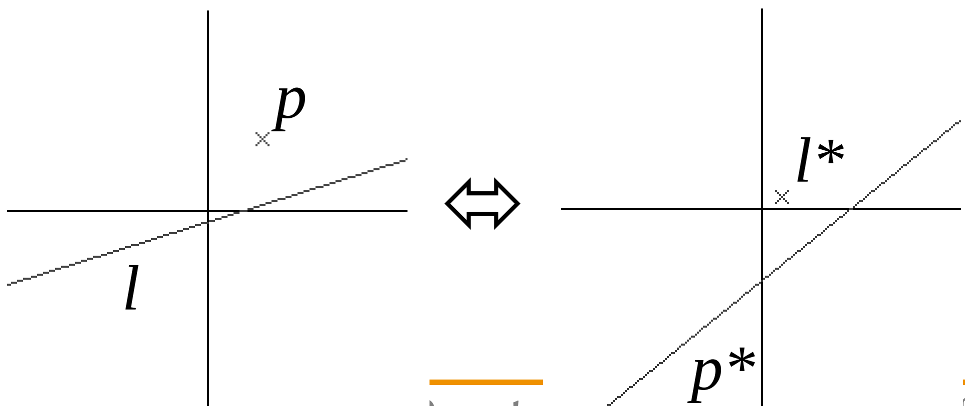
Duality Transforms

- **A *duality transform* is a mapping which takes an element *e* in the *primal plane* to element *e** in the *dual plane*.**

- One possible duality transform:
  point p: (p_x,p_y) ⇔ line p^*: $y = p_x x - p_y$
  line l: $y = mx + b$ ⇔ point l^*: $(m, -b)$

Duality Transforms

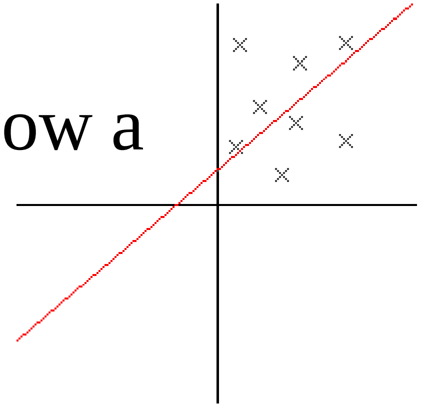
- This duality transform takes
  - points to lines, lines to points
  - line segments to double wedges
- This duality transform preserves order
  - Point *p* lies above line *l* ⇔ point *l** lies above line *p**



Back to the Discrepancy problem

To determine our discrete measure, we need to:

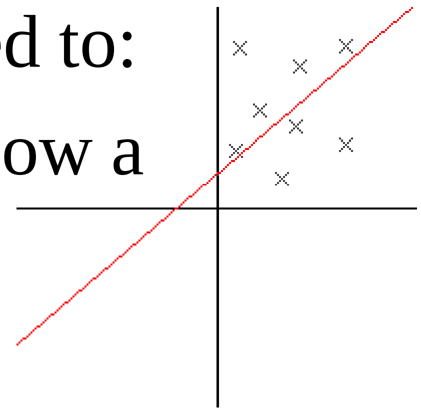
Determine how many sample points lie below a given line (in the primal plane).



Back to the Discrepancy problem

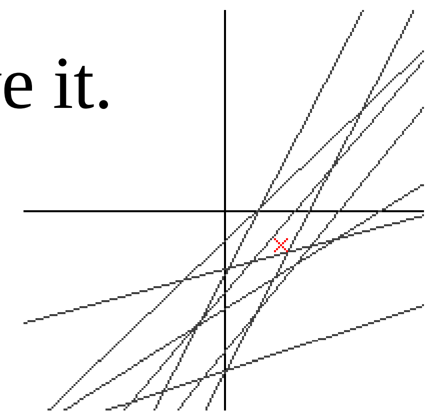
To determine our discrete measure, we need to:

Determine how many sample points lie below a given line (in the primal plane).



⇕ *dualizes to* ⇕

Given a point in the dual plane we want to determine how many sample lines lie above it.



Is this easier to compute?

Duality

- The dualized version of a problem is no easier or harder to compute than the original problem.

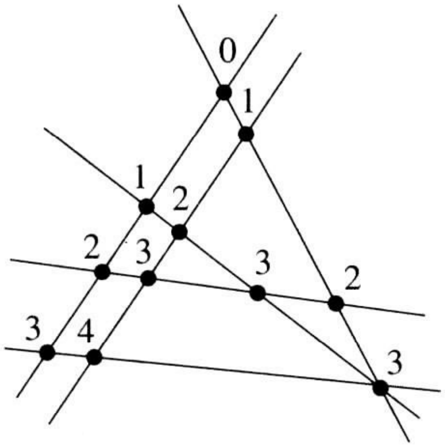
- But the dualized version may be easier to think about.

Back to Discrepancy (Again)

- For every line between two sample points, we want to determine how many sample points lie below that line.

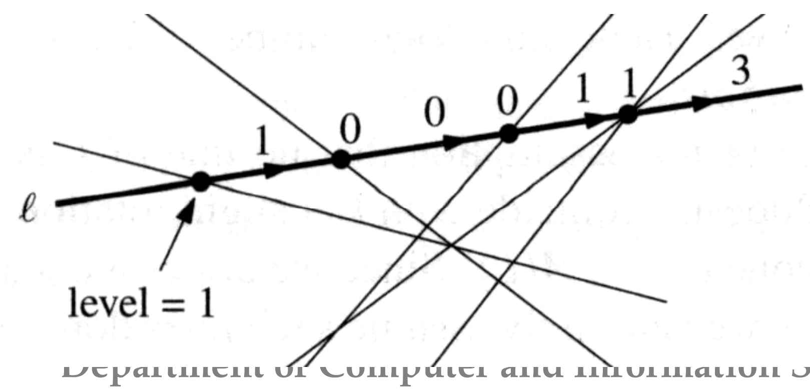
  -or-

- For every vertex in the dual plane, we want to determine how many sample lines lie above it.
- We build the arrangement $A(S^*)$ and use that to determine, for each vertex, how many lines lie above it.
  Call this the *level* of a vertex.



Levels and Discrepancy

- For each line l in S^*
  - Compute the level of the leftmost vertex. $O(n)$
    - Check, for all other lines l_i, whether l_i is above that vertex
  - Walk along l from left to right to visit the other vertices on l, using the DCEL.
    - Walk along l, maintaining the level as we go (by inspecting the edges incident to each vertex we encounter).
  - $O(n)$ per line



What did we just do?

- Given the level of a vertex in the (dualized) arrangement, we can compute the discrete measure of S wrt the h that vertex corresponds to in $O(1)$ time.
- **We can compute all the interesting discrete measures in $O(n^2)$ time.**
- Thus we can compute all $\Delta_S(h)$, and hence $\Delta_H(S)$, in $O(n^2)$ time.

Summary

- Problem regarding points S in ray-tracing
- Dualize to a problem of lines L.
- Compute arrangement of lines $A(L)$.
- Compute level of each vertex in $A(L)$.
- Use this to compute discrete measures in primal space.
- We can determine how good a distribution of sample points is in $O(n^2)$ time.

Further

- Zone Theorem has an analog in higher dimensions
  - Zone of a hyperplane in an arrangement of n hyperplanes in d-dimensional space has complexity $O(n^{d-1})$