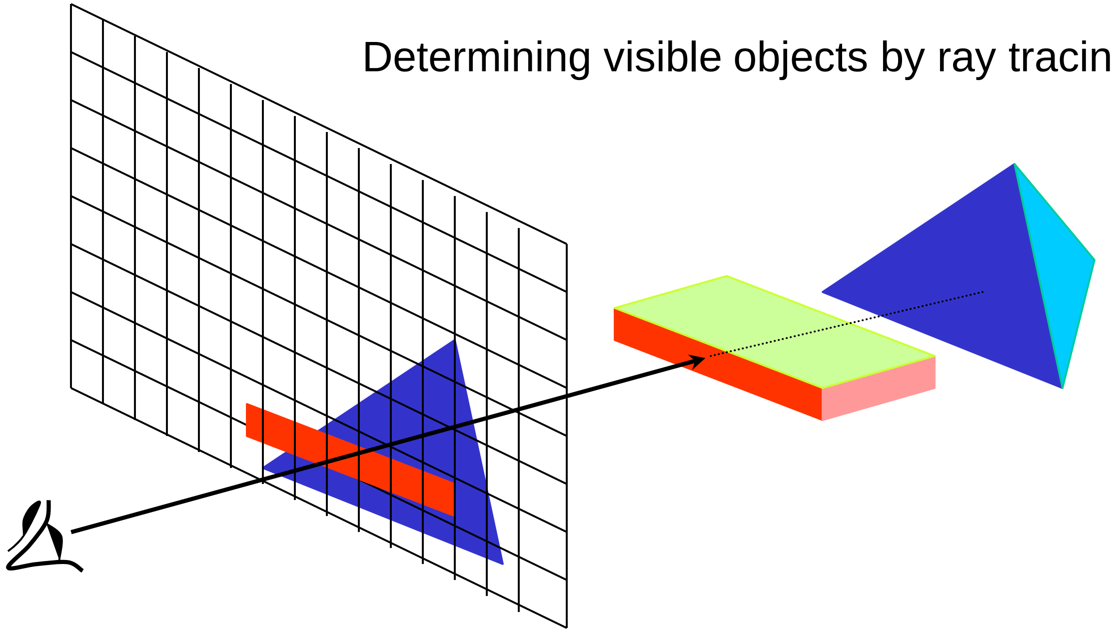
Duality and Line Arrangements

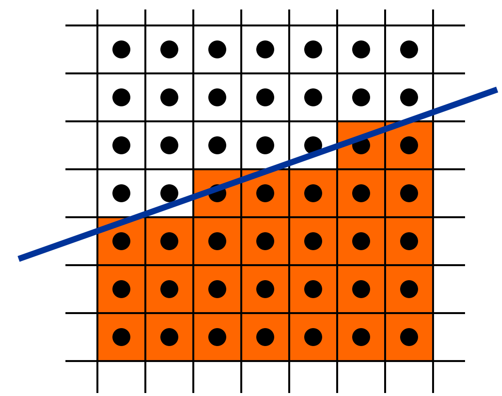
slides by Andy Mirzaian

(a subset of the original slides are used here)

Super-sampling in Ray Tracing

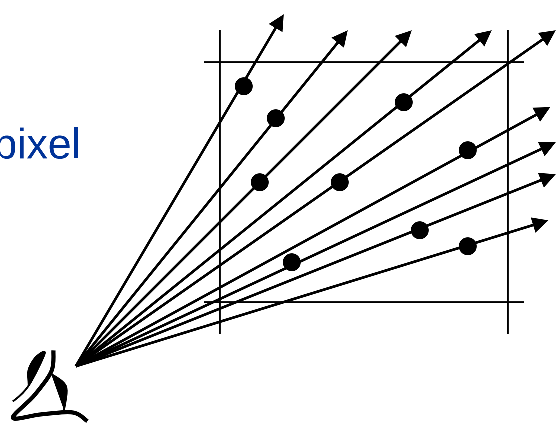
Determining visible objects by ray tracing.





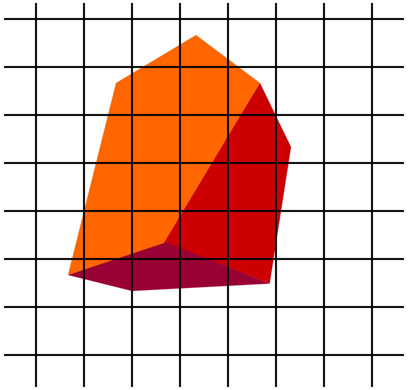
- A ray through each pixel center.
- Problems: jagged edges, false hit/miss.

- A solution: super-sampling. Shoot many rays per pixel (usually at random)
  if 100 rays shot at pixel, and 43 hit the same object, we say object visible in roughly 43% pixel area.



Computing the Discrepancy

Pixel U = $[0:1] \times [0:1]$
S = a set of n sample points in U
$\mathcal{H}$ = set of all half-planes



For h $\in \mathcal{H}$ define:

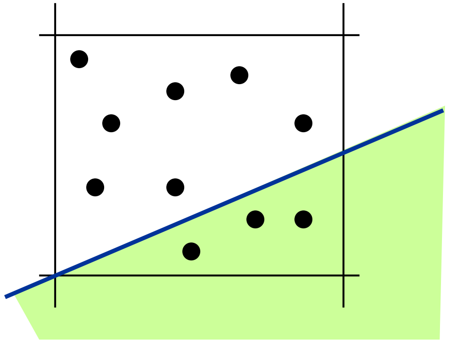
$\mu(h) = \text{area}(h \cap U)$ continuous measure

$\mu_S(h) = |S \cap h| / |S|$ discrete measure

$\Delta_S(h) = | \mu(h) - \mu_S(h) |$ discrepancy of h

$\Delta_{\mathcal{H}}(S) = \sup_{h \in \mathcal{H}} \Delta_S(h)$ half-plane discrepancy of S

The bounding line of this worst half-plane h passes through either only 1, or at least 2, sample points.



$\Delta_S(h) = |1/4 - 3/10| = 0.05$

FACT: **$\Delta_{\mathcal{H}}(S)$** can be computed in $O(n^2)$ time, using
Geometric Duality &
Arrangement of lines in the plane.

Geometric Duality

Point -to- hyperplane Transformations

Some Applications:

- Intersection of half-spaces $\Leftrightarrow$ Convex Hull of point sets

- Whenever the problem becomes intuitively “easier” in the dual space.

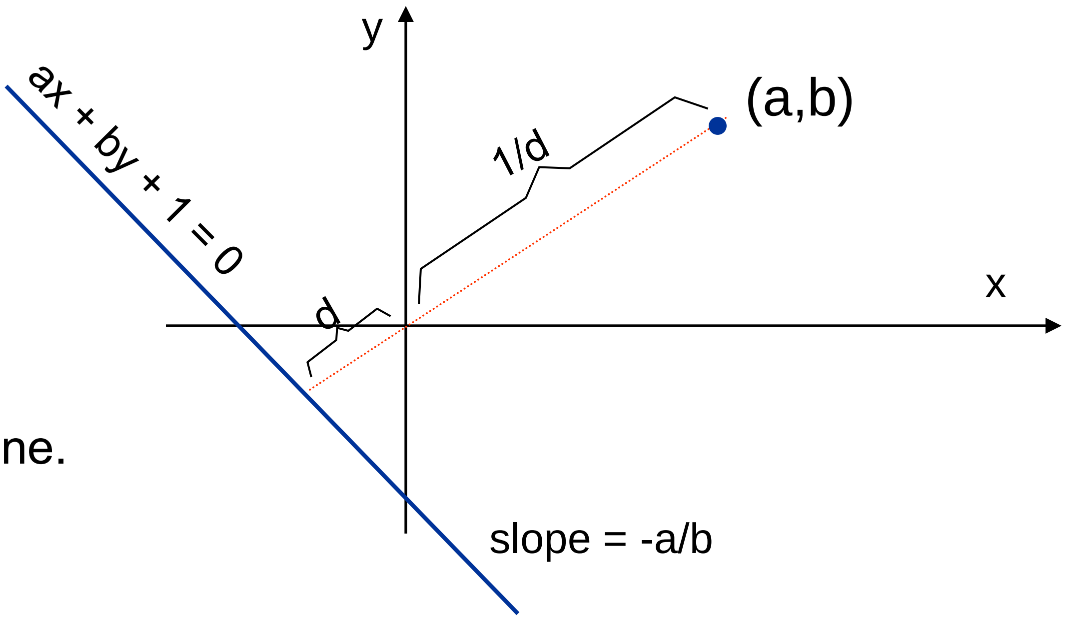
1. Hough (or Reciprocal) Transform (1969)

$\Re^d$	Point: $(a_1, a_2, \ldots, a_d)$ $\neq$ origin	Hyper-plane: $a_1x_1+a_2x_2+\ldots+a_dx_d+1=0$ not passing through the origin
$\Re^2$	Point: $(a,b) \neq (0,0)$	Line: $ax + by + 1 = 0$

p* = dual of p

$d(0,p) \times d(0,p^*) = 1$

The origin is "above" the line.

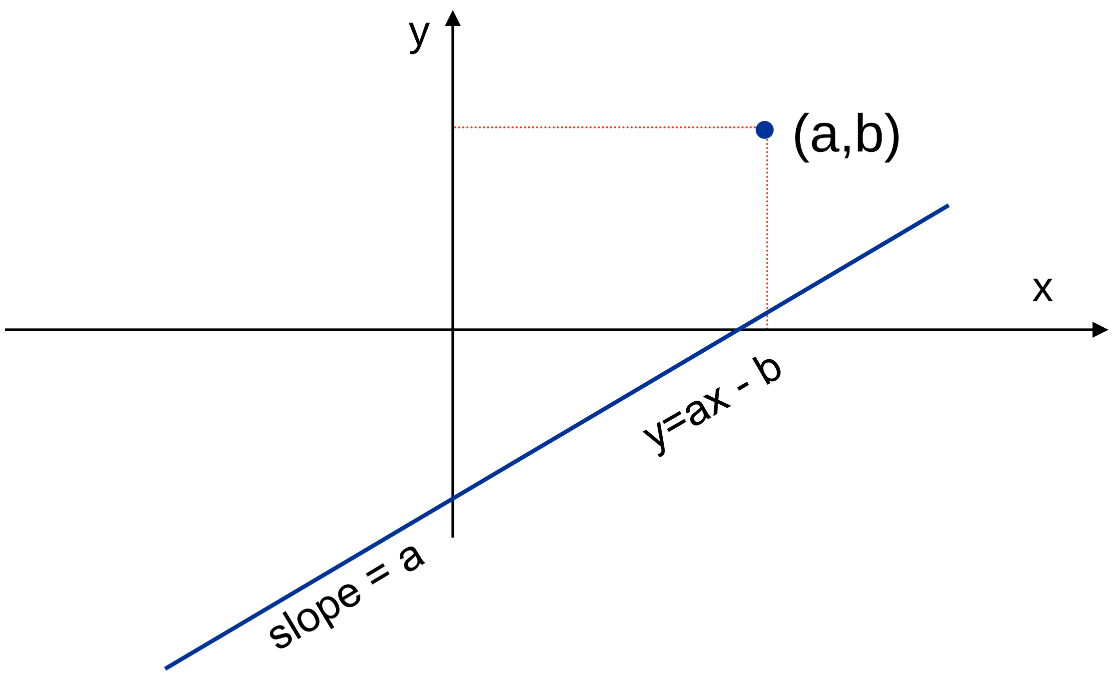


2. Another Point-Line Transform

Point p: (a, b) $\longrightarrow$ *line* p^*: $y = ax - b$ *(non-vertical line)*

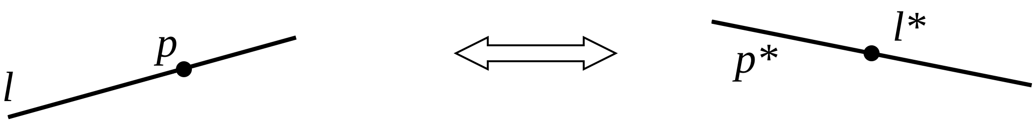
Line l: $y = ax + b$ $\longrightarrow$ *point* l^*: $(a, -b)$

symmetric: $p^{**} = p$

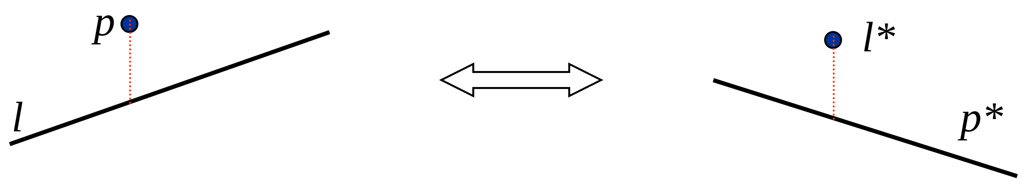


Duality Transforms Preserve Incidence

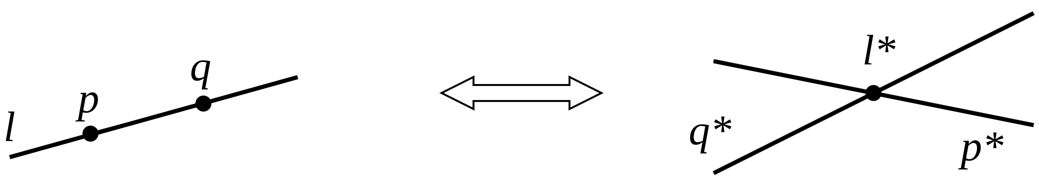
1. *Point p is mapped to line p*.*
2. *Line l is mapped to point l*.*
3. *Point p and line l are incident ⇔ line p* and point l* are incident*



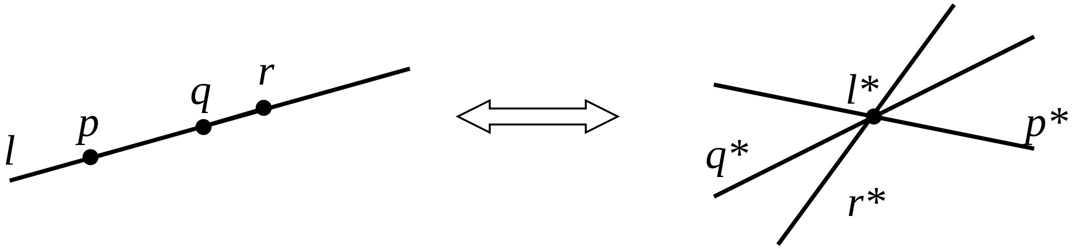
4. *Above/Below relation is also preserved (or reversed).*



5. *Line l passes through points p & q ⇔ Lines p* & q* intersect at point l*.*



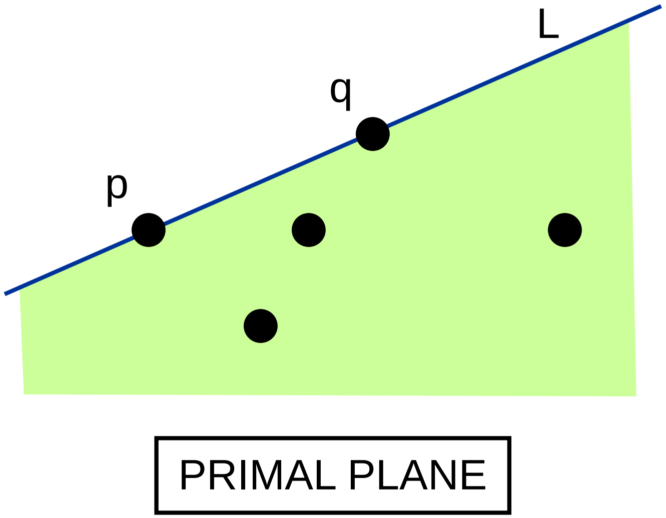
6. *Points p, q, r are collinear ⇔ Lines p*, q*, r* are concurrent.*

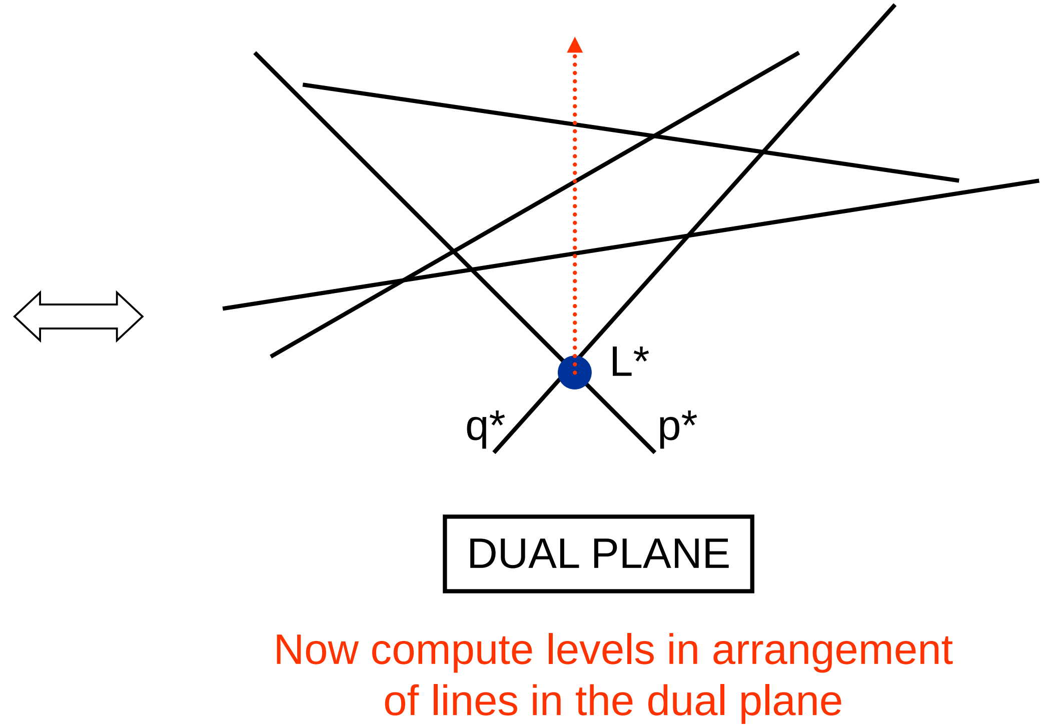


They preserve incidence since they are affine transforms

Problem Transformation by Duality

Problem 1: Compute the discrepancy:

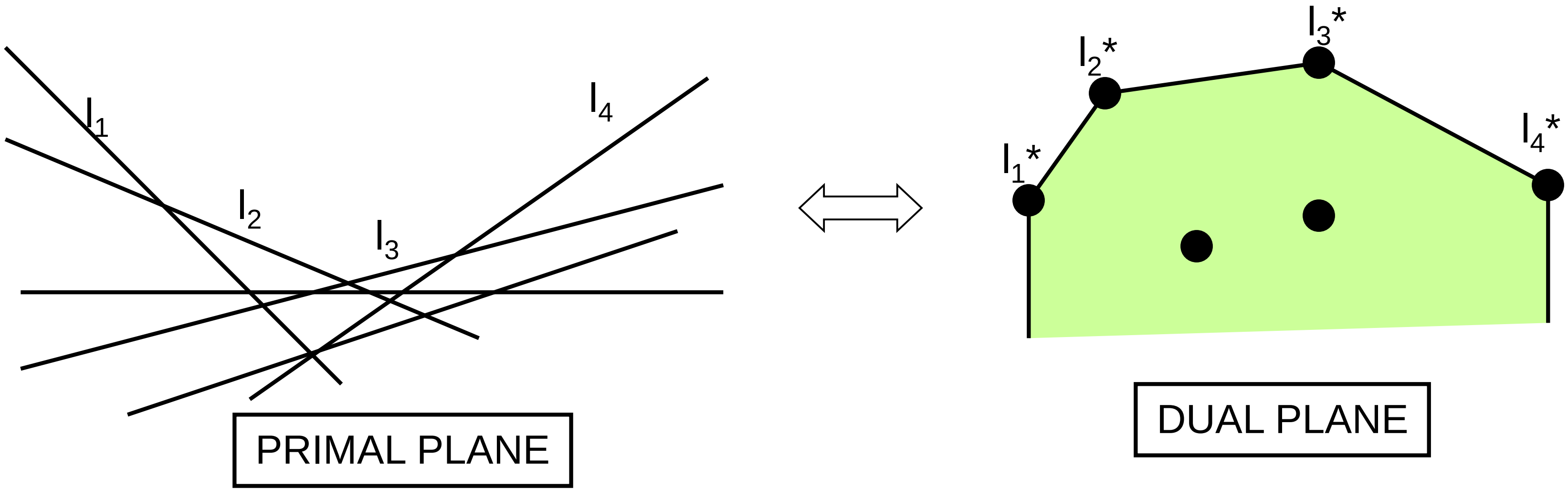




Problem 2: [Degeneracy]: are there any 3 collinear points among n given points?
Solution: Duality: are there 3 concurrent lines among n given lines?
(Use arrangement of lines)

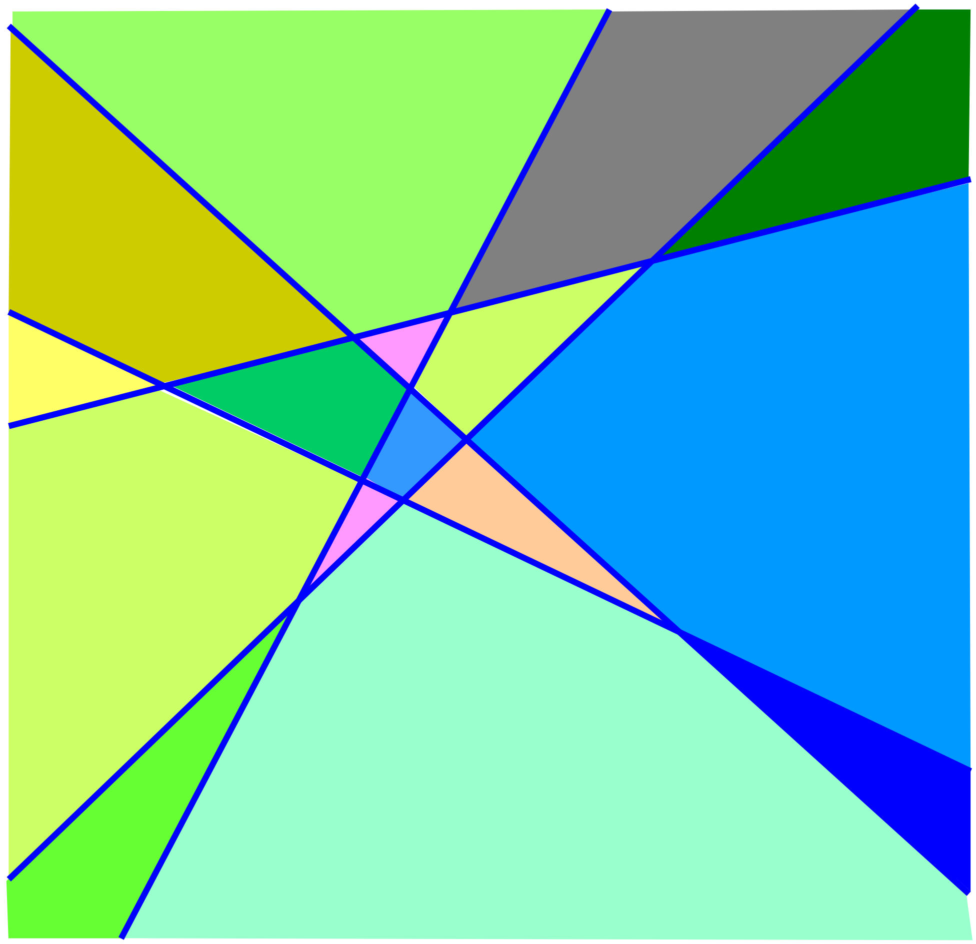
Problem Transformation by Duality

Problem 3: Find upper (respectively, lower) envelope of n given lines
Solution: Compute upper (respectively, lower) convex hull of the n dual points



Problem 4: Compute intersection of n half-spaces
Solution: Compute Convex Hull of n dual points
Caution: What happens if the origin is not in the intersection?

Arrangements



Arrangements of lines and hyper-planes

$\mathcal{L} = \{ l_1, l_2, \ldots, l_n \}$ n lines in $\Re^2$ (in general: n hyper-planes in $\Re^d$)

$\mathcal{A}(\mathcal{L})$ = arrangement of $\mathcal{L}$, i.e., subdivision of $\Re^2$ (resp., $\Re^d$) induced by $\mathcal{L}$.

Assume: $\mathcal{L}$ is in general position, i.e., no 2 lines in $\mathcal{L}$ are parallel,
& no 3 are concurrent.

Combinatorial complexity of $\mathcal{A}(\mathcal{L})$ in $\Re^2$:

V = # vertices = $\binom{n}{2}$

E = # edges = n^2 (each line is cut into n edges)

F = # regions = $\Theta(n^2)$

Euler: (V+1) – E + F = 2 vertex at ∞

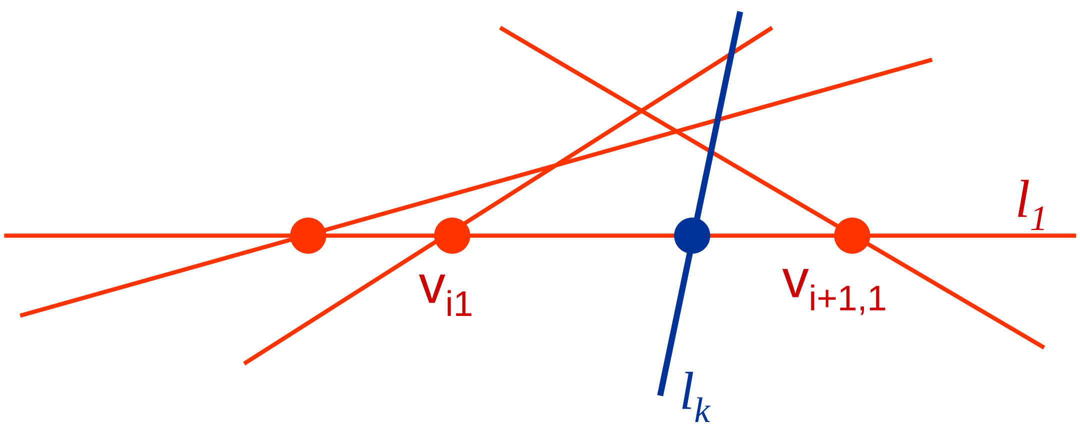
$\therefore$ F = $(n^2 + n + 2) / 2 = \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{d}$

Arrangement Construction

Will take at least $\Omega(n^2)$ time & space, due to combinatorial size of $\mathcal{A}(\mathcal{L})$.

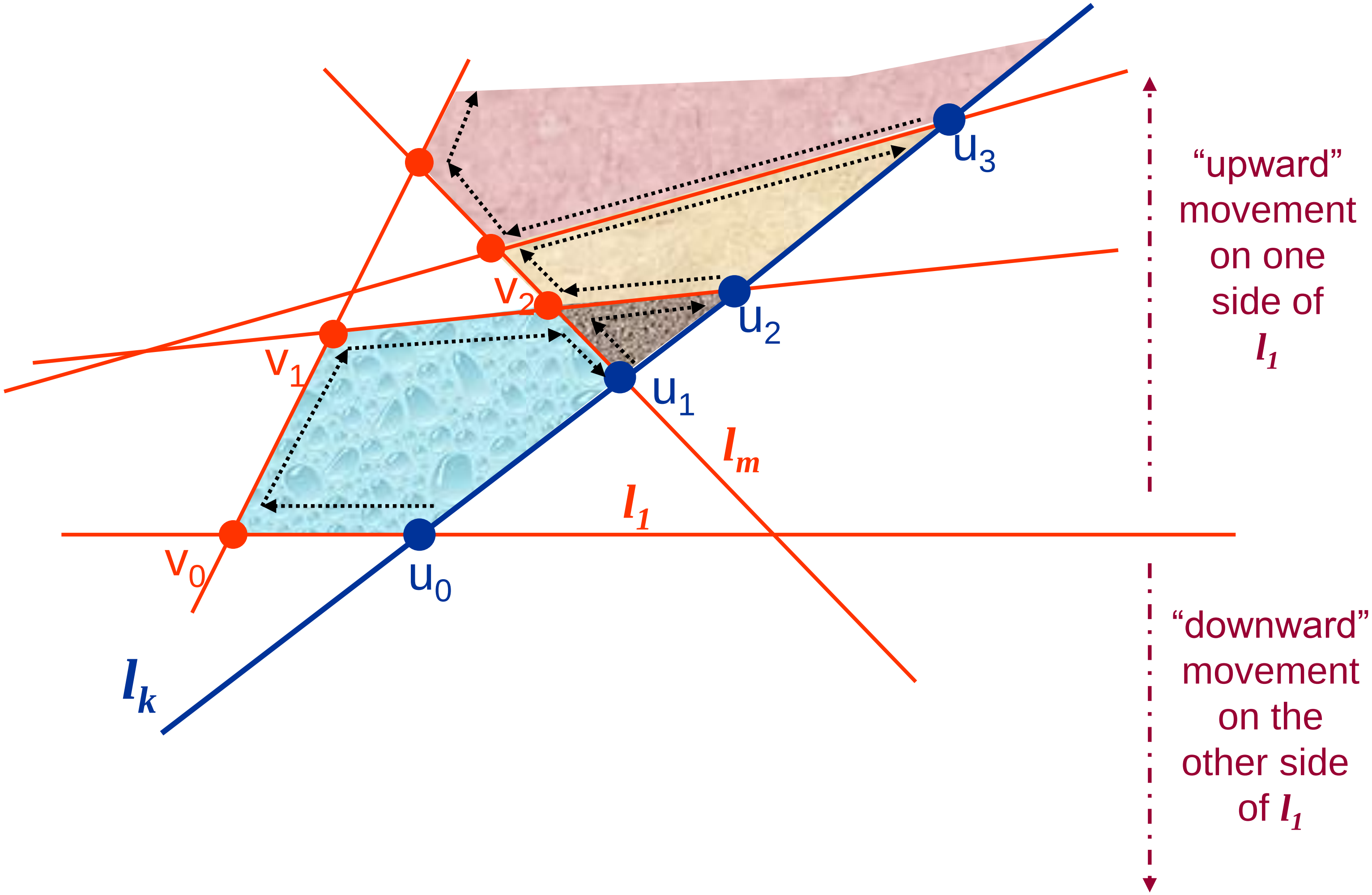
1. <u>Plane Sweep</u>:
   At least $\Omega(n^2 \log n)$ time, since it will "sort" the arrangement vertices.

2. <u>Naïve Incremental Algorithm</u>
   $\mathcal{A}(\{ l_1, l_2, \ldots, l_{k-1} \}) \rightarrow \mathcal{A}(\{ l_1, l_2, \ldots, l_k \}),\ \text{for } k=2..n$
   - Binary Search to find v_{i1}, $v_{i+1,1}$.
   - O(log k) for each of $l_1, l_2, \ldots, l_{k-1}$.
   - O(k log k) to insert l_k.
   - Total $O(\sum_k k \log k) = O(n^2 \log n)$ time, and $O(n^2)$ space.



Refined Incremental Algorithm

How to insert l_k in $\mathcal{A}(\{\, l_1 , l_2 , \ldots , l_{k-1} \})$?



Refined Incremental Algorithm

How to insert l_k in $\mathcal{A}(\{ l_1 , l_2 , ... , l_{k-1} \})$:

1. Find $u_0 = l_1 \cap l_k$ and rightmost vertex v_0 on l_1 to the left of u_0, in O(k) time.
   Let v_0v_1 be CCW from v_0u_0 on vertex v_0 .

2. If segment v_0v_1 intersects l_k , then we have closed a polygonal line that starts from a previous intersection point, namely u_0, and ends in another intersection point, namely u_1. Therefore, we can insert u_1 properly in the adjacency list.
   We now go to u_1 and repeat steps 1,2,3 with u_1 as the new u_0 and l_m as the new l_1.

3. If v_0v_1 does not intersect l_k , then we take the next vertex v_2 CCW on v_1 from v_1v_0 and repeat the same procedure.

4. When we encounter a vertex that has a leftmost edge which is a ray diverging from l_k, we have finished the "upward" movement.

5. We do a similar "downward" movement, starting from v_0, on the other side of l_1.

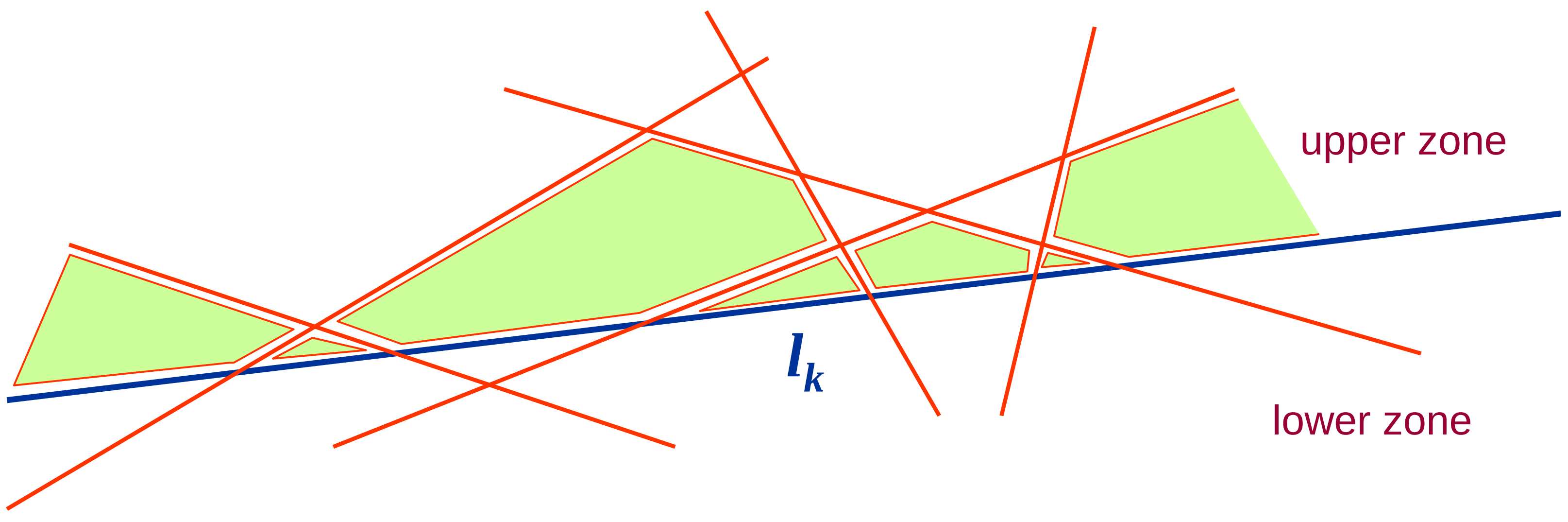
ZONE of a line in the arrangement

Zone of l_k

= collection of the polygonal regions of the arrangement that have an edge on l_k.

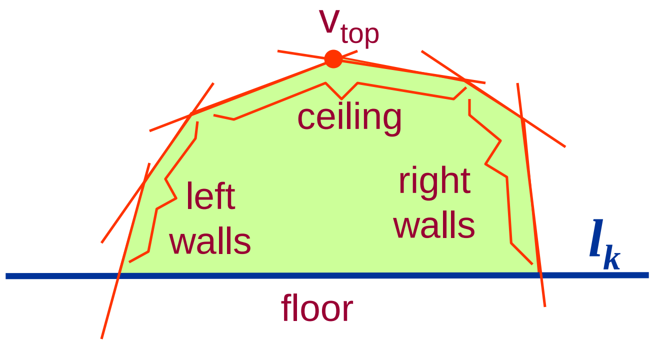
Combinatorial complexity of a zone

= total number of vertices of the polygonal regions in the zone (counting multiplicities).



ZONE Complexity

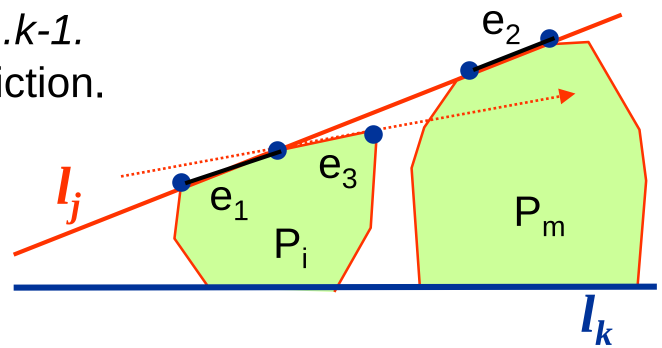
THEOREM: Combinatorial complexity of zone of l_k in $\mathcal{A}(\{ l_1 , l_2 , \ldots , l_{k-1} \})$ is $\leq$ **5k-6** on each side ($\leq$ **10k–12** on both sides).



Proof 1: ∃k convex polygonal regions incident to in the "upper" zone of l_k (similarly in the "lower" zone).

Define: ceiling, left/right wall, & floor edges for each polygon as the figure on the right.

Claim: ∃ ≤ 1 left wall & ≤ 1 right wall on each line l_j , $j = 1..k-1$.
Proof: $e_1 \neq$ ceiling ⇒ e_3's extension cuts P_m ⇒ a contradiction.



Corollary: ∃ ≤ 2(k-1) -2 = 2k-4 wall edges.
(First poly has no left wall, last poly has no right wall.)

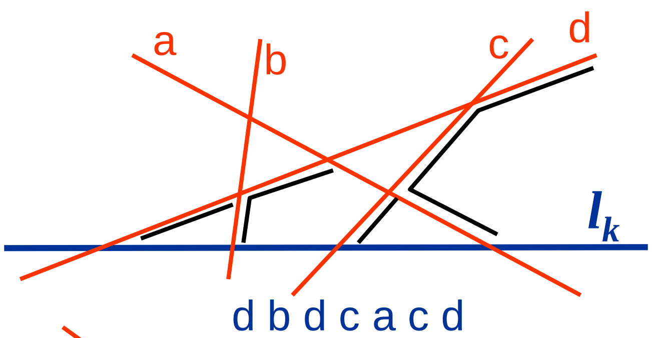
Total count:

k	floors
2k-4	walls
2k-2	ceilings (1s & last have only one ceiling edge)
5k-6	total

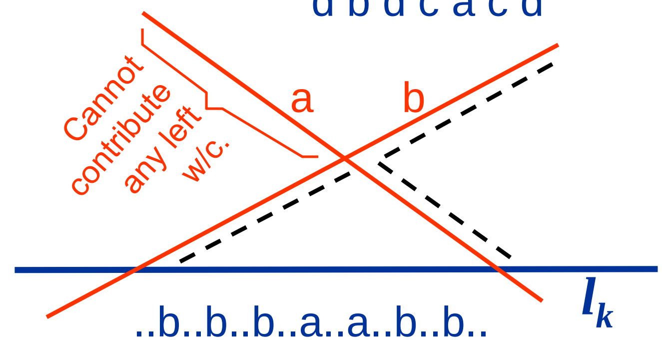
ZONE Complexity

THEOREM: Combinatorial complexity of zone of l_k in $\mathcal{A}(\{ l_1, l_2, \ldots, l_{k-1} \})$ is $\leq$ **5k-6** on each side ($\leq$ **10k–12** on both sides).

Proof 2: By Davenport-Schinzel sequences.
Consider the sequence of only left wall/ceiling edges in the traversal of upper-zone of l_k (similarly for right …).

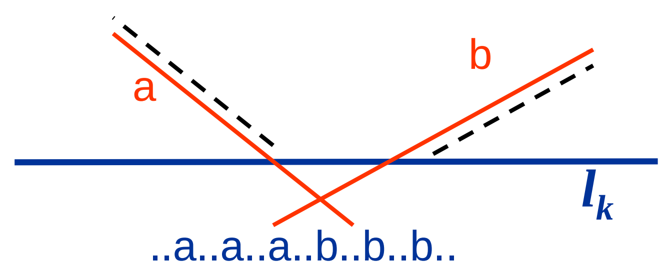


Claim: This is a (k-1, 2) DS sequence.
That is, …a…b…a…b…
is a forbidden sub-sequence.



Total count:

k	floors
2(k-1) -1	left walls / ceilings
2(k-1) -1	right walls / ceilings
5k-6	total



Arrangement Complexity

THEOREM: The arrangement $\mathcal{A}(\mathcal{L})$ of n lines $\mathcal{L} = \{ l_1 , l_2 , \ldots , l_n \}$ in $\Re^2$ can be constructed in optimal time & space $O(n^2)$.

THEOREM: Let $\mathcal{H} = \{ h_1 , h_2 , \ldots , h_n \}$ be a set of n hyper-planes in $\Re^d$.

(a) The combinatorial size of the zone of any hyper-plane in the arrangement $\mathcal{A}(\mathcal{H})$ is $O(n^{d-1})$.

(b) $\mathcal{A}(\mathcal{H})$ can be constructed in optimal time & space $O(n^d)$.

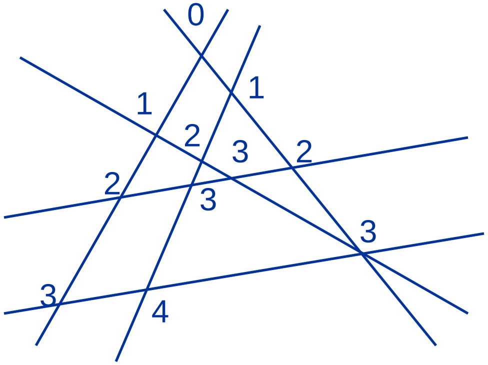
Levels and Discrepancy

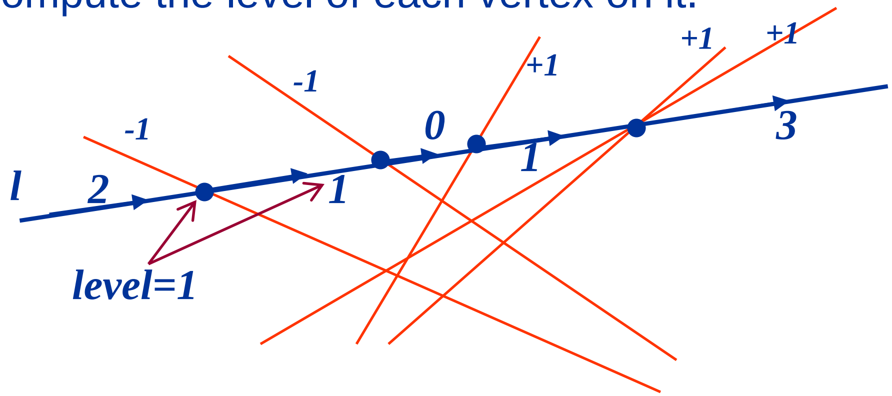
U = [0 : 1] × [0 : 1], S = a set of n points in U.
Dualize: S ⟶ S* ⟶ $\mathcal{A}$(S*).

For each vertex v in $\mathcal{A}$(S*) we need to compute how many lines of S* are

a) strictly above it [let's call this level(v)],
b) pass through it,
c) strictly below it.

We essentially need to compute the levels of all vertices of the arrangement $\mathcal{A}$(S*).
Take a walk along each line, and compute the level of each vertex on it.





Compute level of leftmost vertex of ***l*** in O(n) time, then compute each of its subsequent vertices in order of degree of the vertex.
Total time, over all lines in S*, is O(sum of vertex degrees + n^2) = O(n^2).