CS164: Voronoi/Delaunay Diagrams, Distance Functions

Leonidas Guibas
Computer Science Dept.
Stanford University

Georgy Voronoy (1868-1908)

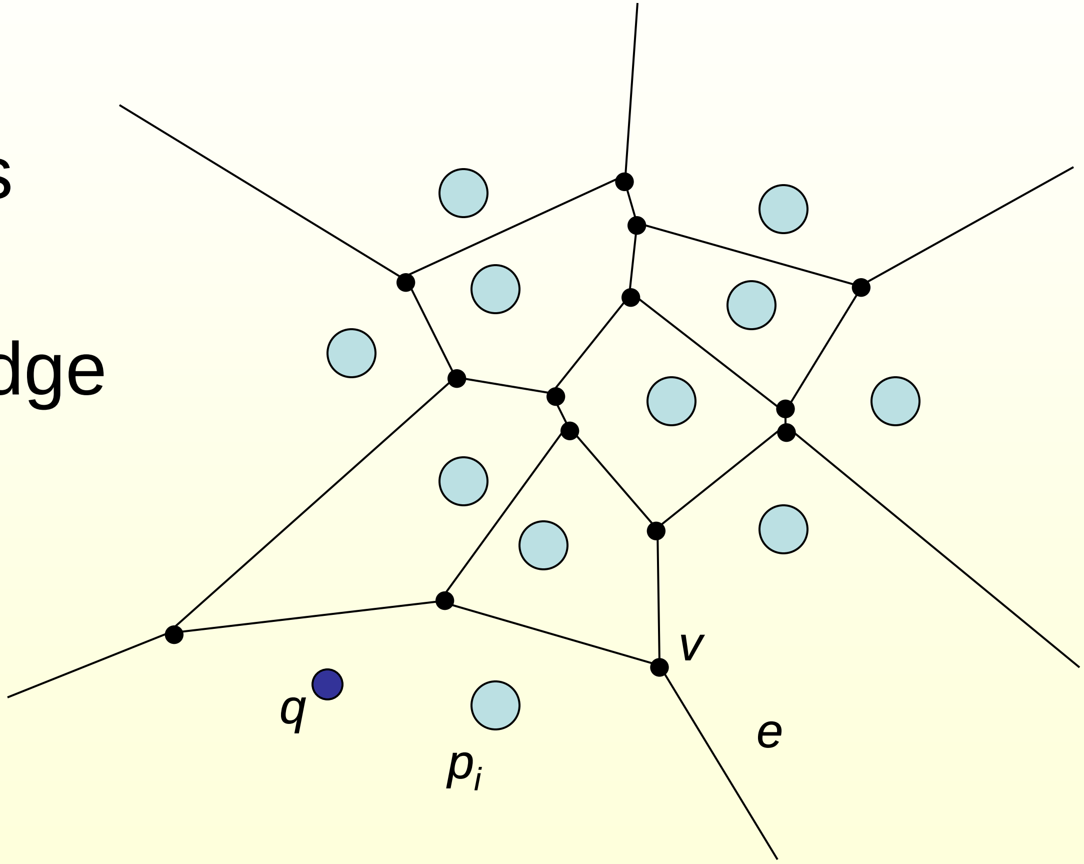
Post Office: What is the area of service?

p_i : site points

q : free point

e : Voronoi edge

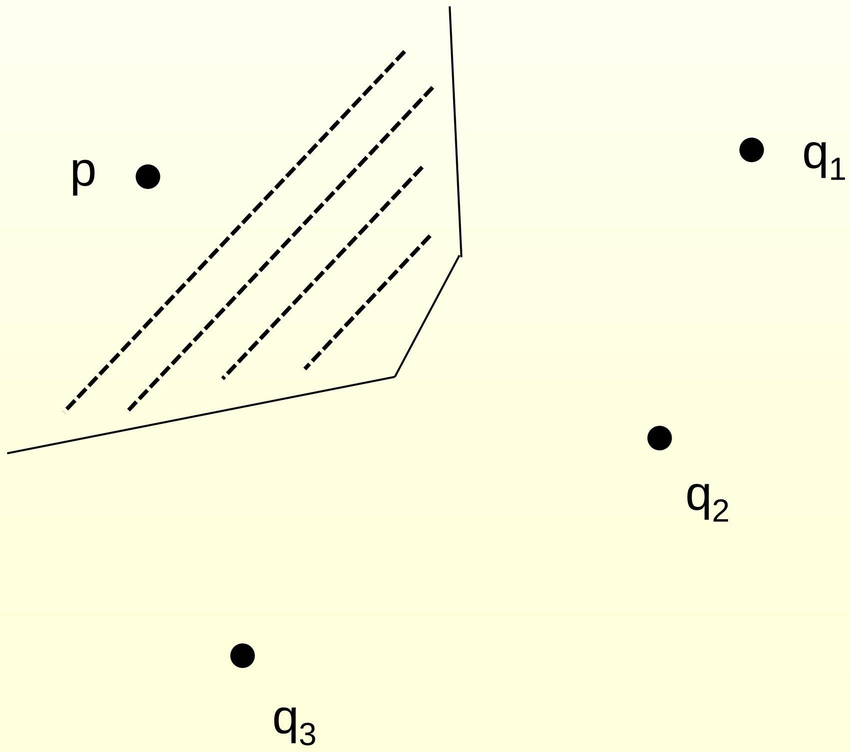
v : Voronoi vertex



Voronoi region

- P = finite set of points in R^2, $p \in P$.

$$V(p) = \{x \in R^2 : \|x - p\| \leq \|x - q\|, \forall q \in P\}$$



Voronoi Decomposition of *P*

- $V(p)$ is a polygon.
- $p, q \in P$ determine an edge:

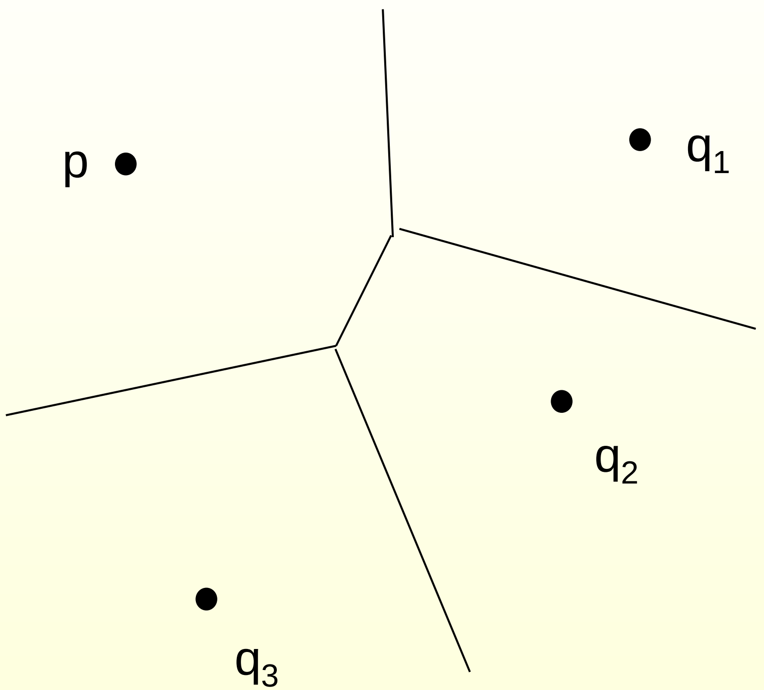
$$V(p,q) = V(p) \cap V(q) =$$
$$\{x \in R^2 : \|x-p\| = \|x-q\| \leq \|x-r\|, \forall r \in P\}$$

- $p, q, r \in P$ determine a vertex:

$$V(p,q,r) = V(p) \cap V(q) \cap V(r) =$$
$$\{x \in R^2 : \|x-p\| = \|x-q\| = \|x-r\| \leq \|x-s\|,$$
$$\forall s \in P\}$$

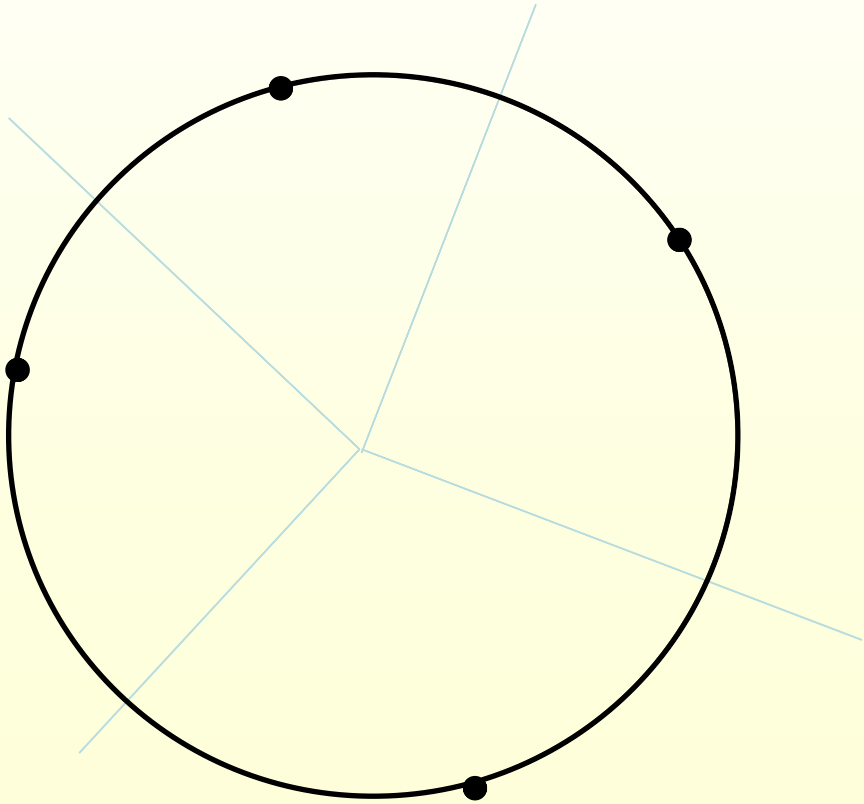
Voronoi Decomposition of *P*

- 2-cells for 1 point
- Edges for 2 points
- Vertices for 3 or more points
- If the points of *P* are in General Position, vertices are always of degree 3



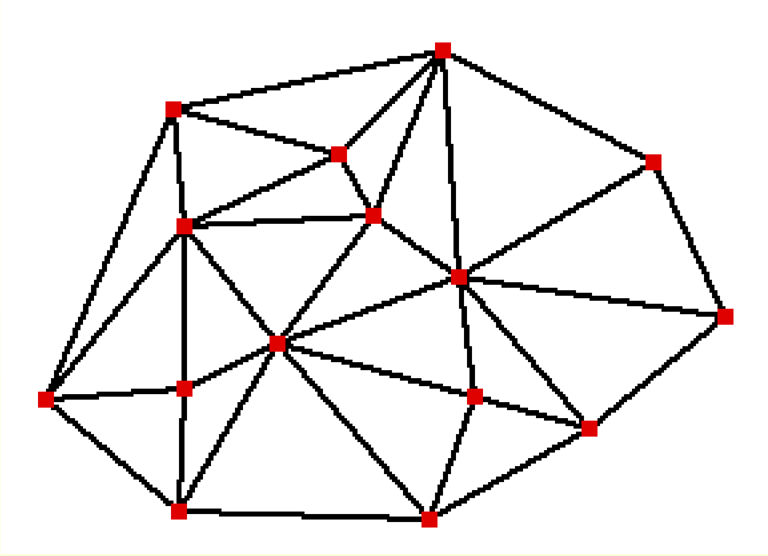
Voronoi Decomposition of P

If 4 (or more) points lie on a circle that contains no other points, we get a 4-valent vertex.



Demo

http://www.pi6.fernuni-hagen.de/GeomLab/VoroGlide/index.html.en

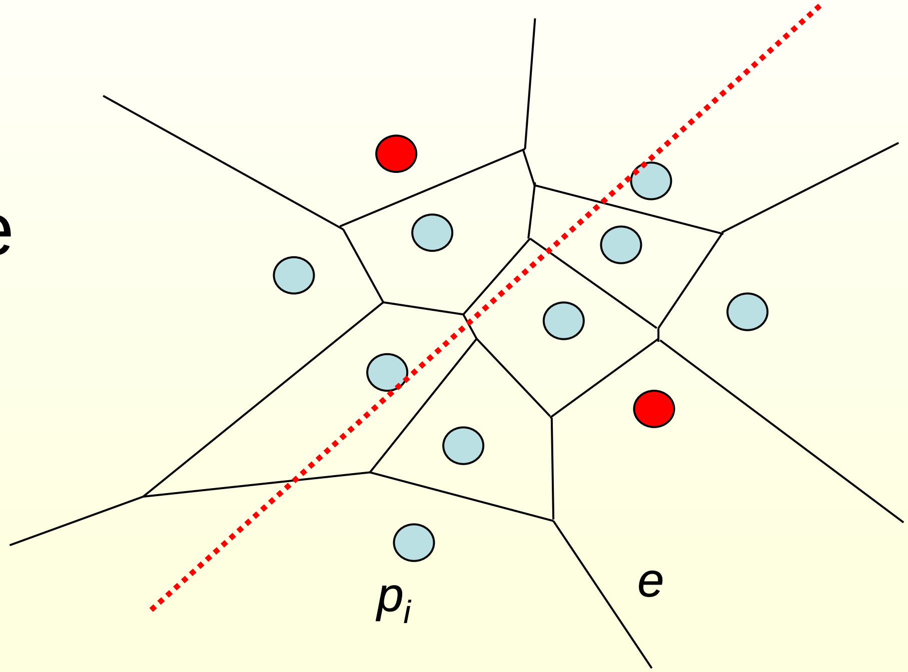


Voronoi diagrams have linear complexity $\{|v|, |e| = O(n)\}$

Intuition: Not all bisectors are Voronoi edges!

p_i : site points

e : Voronoi edge



Claim: For $n \geq 3$, $|v| \leq 2n - 5$ and $|e| \leq 3n - 6$

Constructing Voronoi Diagrams

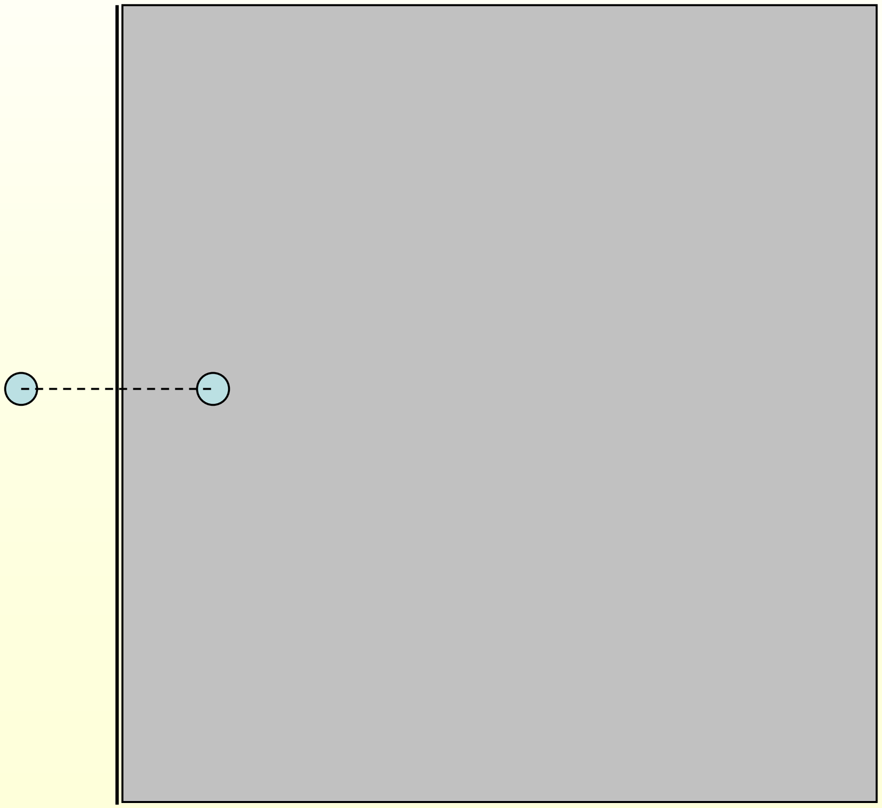
Each region is the intersection of *n* − *1 half-planes*

Constructing Voronoi Diagrams

Given a half plane intersection algorithm…

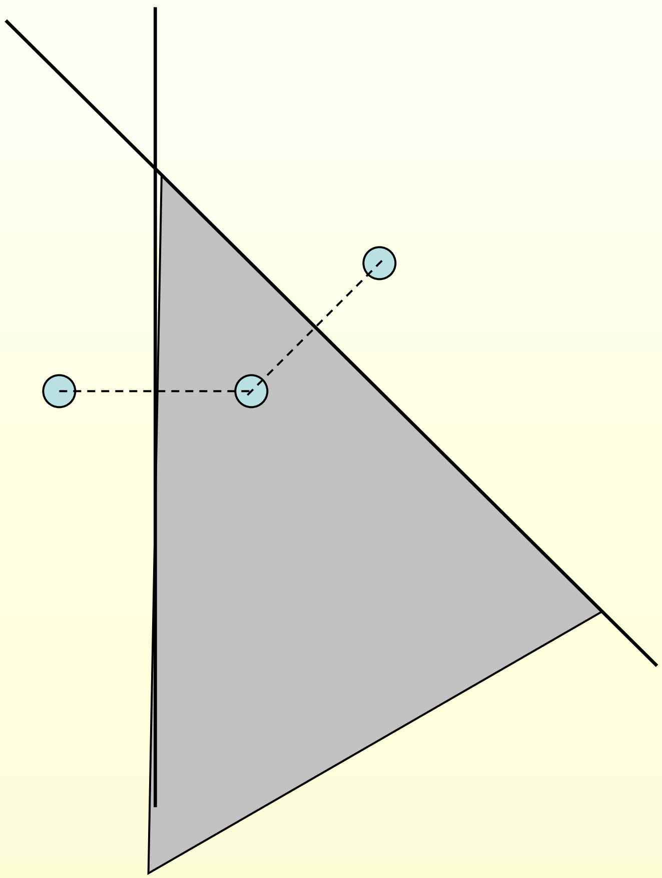
Constructing Voronoi Diagrams

Given a half plane intersection algorithm…



Constructing Voronoi Diagrams

Given a half plane intersection algorithm…

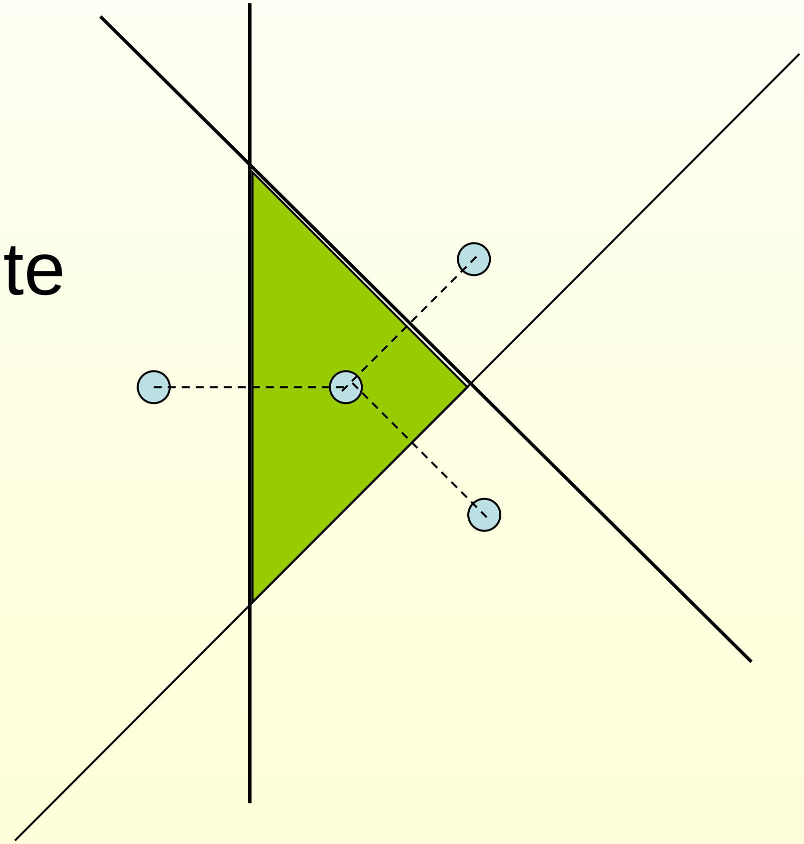


Constructing Voronoi Diagrams

Given a half plane intersection algorithm…

Repeat for each site

Running Time:
O($n^2 \log n$)

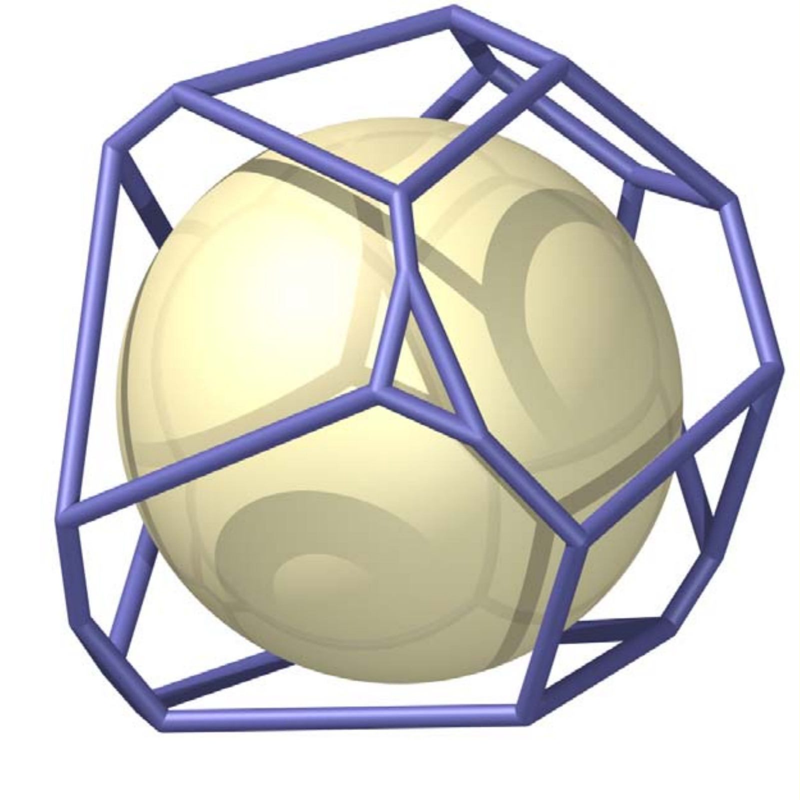


Constructing Voronoi Diagrams

- Half plane intersection O(n^2 log n)
- Fortune's algorithm O(n log n)
  - Sweep line algorithm

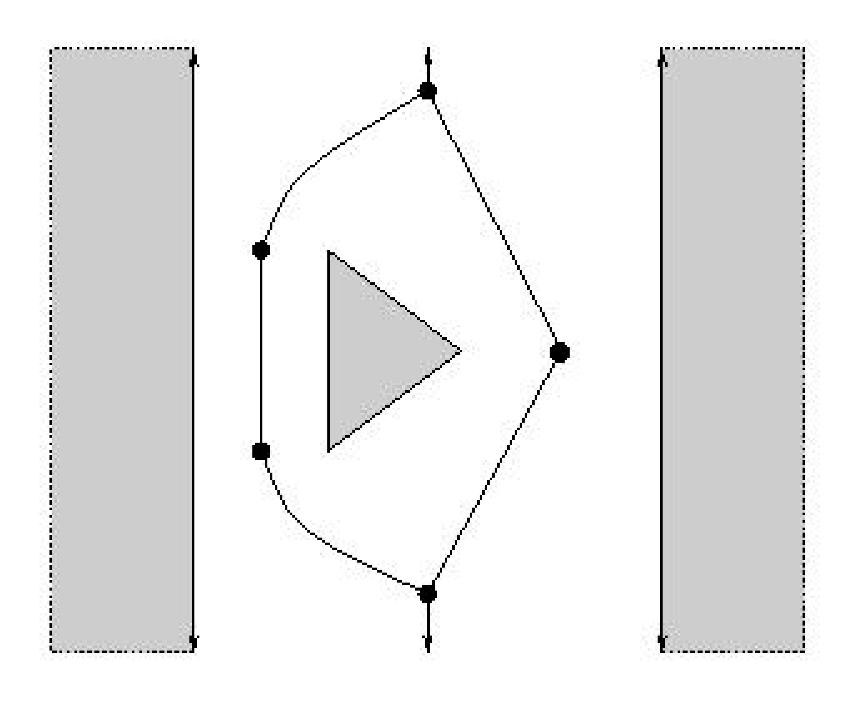
Voronoi Decomposition in R^3

- In R^3, for points in general position we have:
  - 3-cells for 1 point
  - 2-cells for 2 points
  - edges for 3 points
  - vertices for 4 points



Application

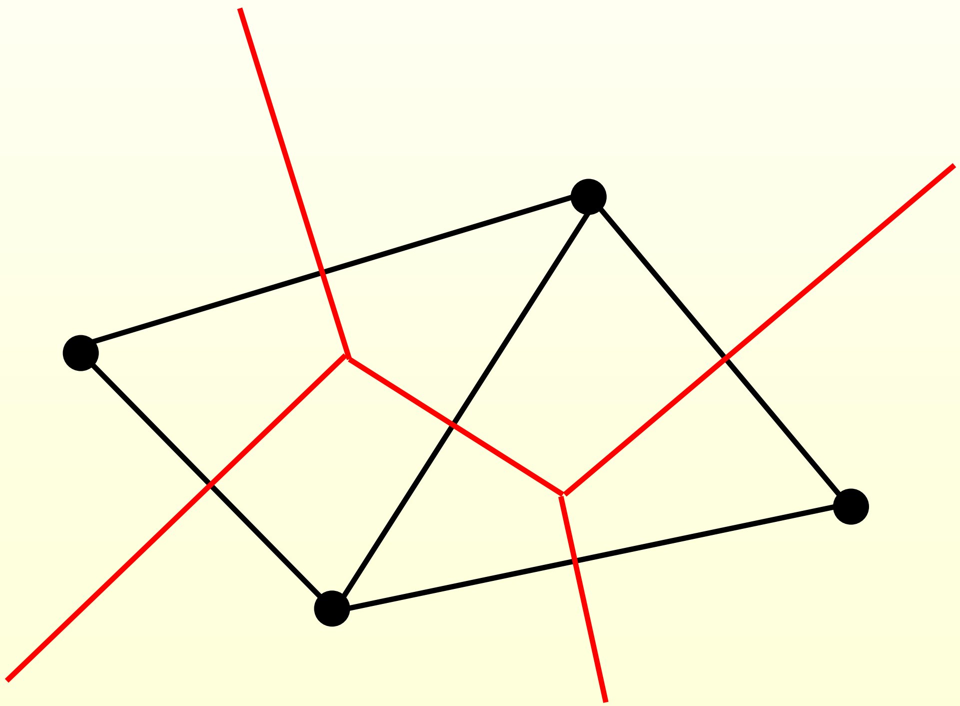
- Clear routes in autonomous robot navigation
  - If the objects are obstacles, then the edges of the graph will be the routes furthest from obstacles.



Delaunay Triangulation

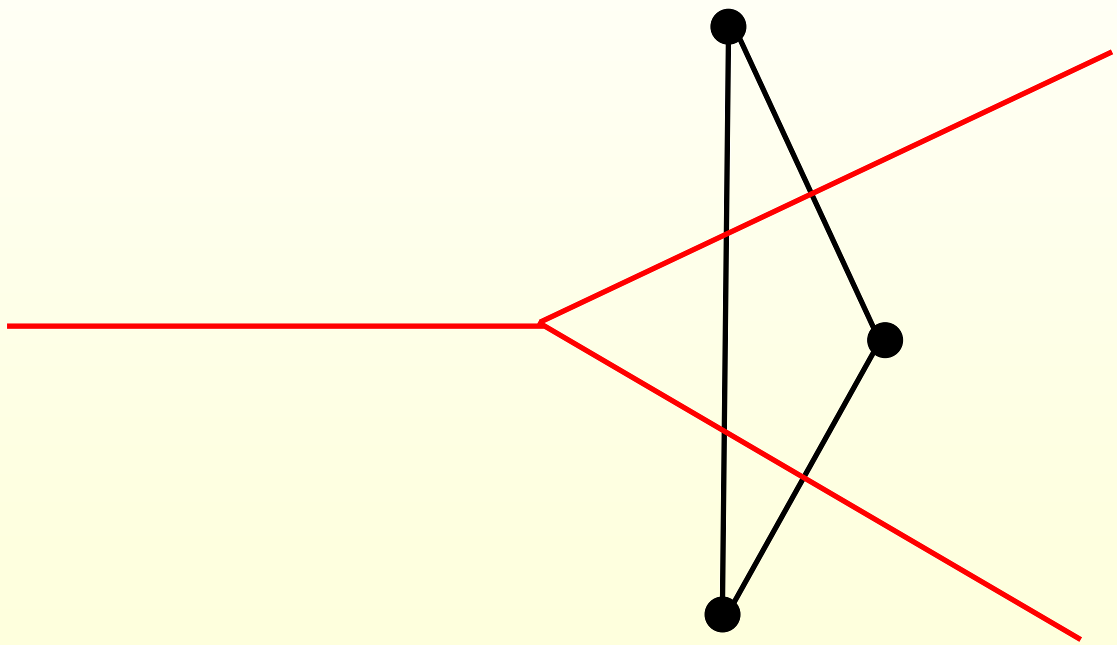
Delaunay Triangulation of P

The dual to the Voronoi Decomposition is the *Delaunay Triangulation*



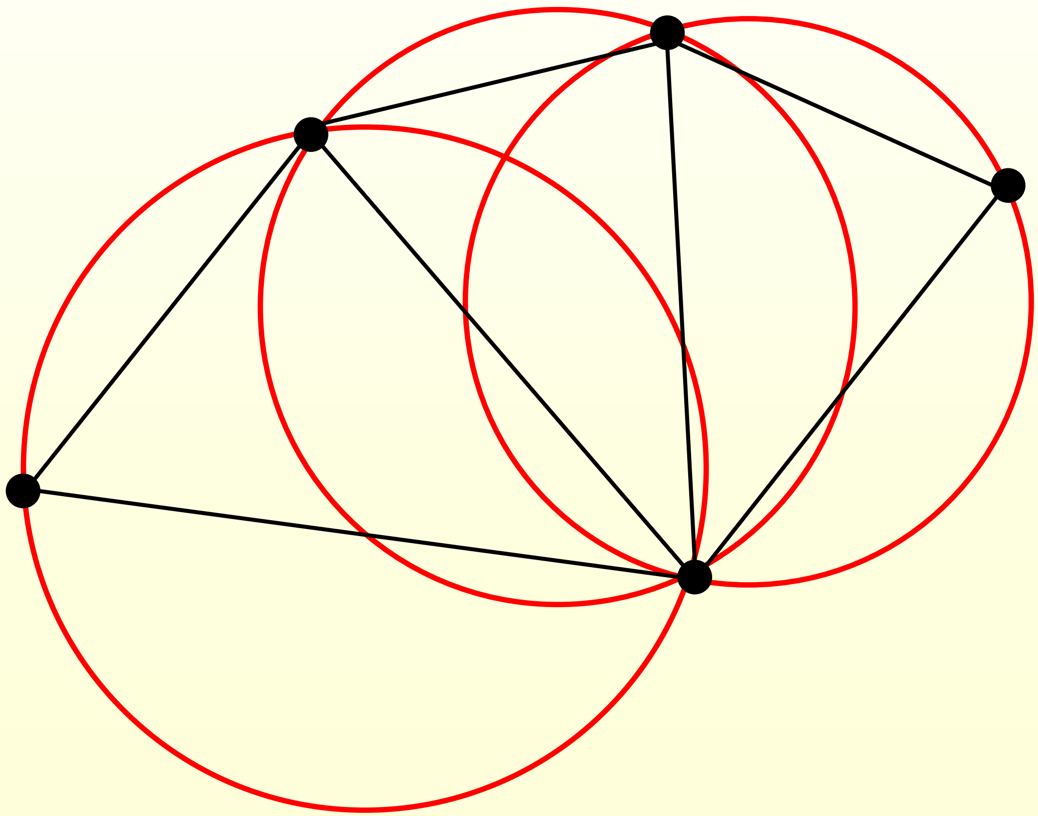
Delaunay Triangulation in P

Dual edges don’t necessarily meet



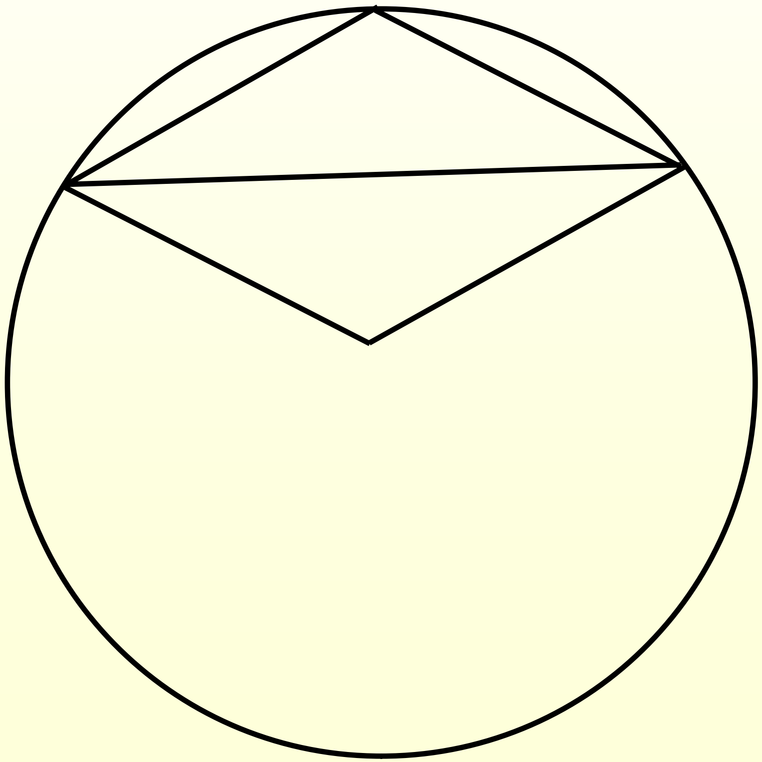
Delaunay Triangulation in P

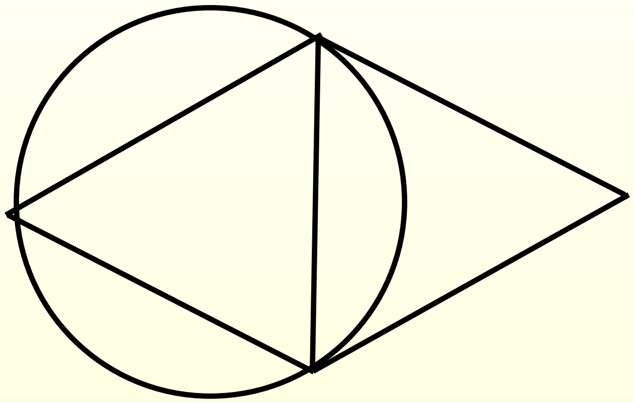
Key property 1: The circumcircle through 3 vertices of a triangle contains no other points.



Delaunay Triangulation in P

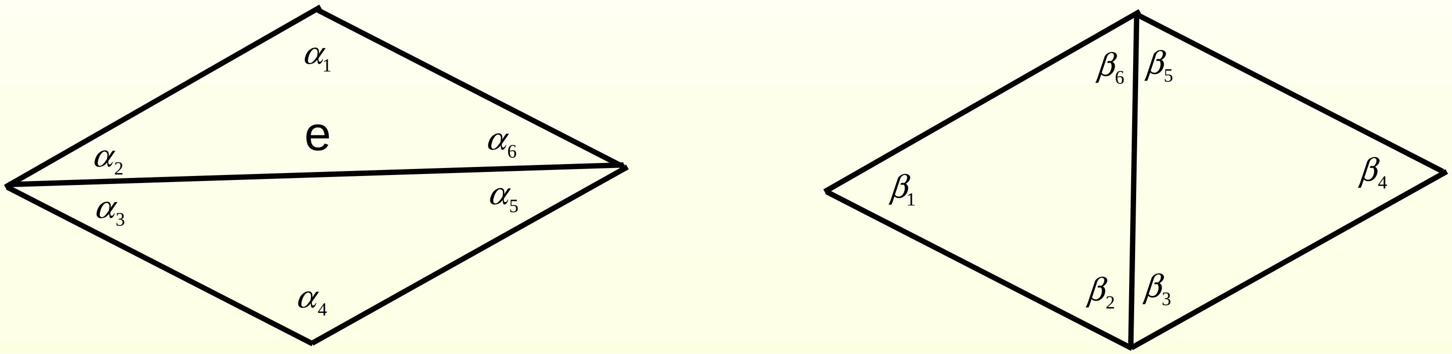
Key property 2: Triangles are thicker





Delaunay Triangulation of P

Key property 2 says no edges is "illegal", where e is illegal if $\min\{\alpha_i\} < \min\{\beta_i\}$



If T is any triangulation we can "flip" edges to make it the Delaunay triangulation of its vertices.

Delaunay Triangulation of P

- T-triangulation with vertices at P.
- Angle Vector of T = $\{\theta_1 < \cdots < \theta_{3m}\}$.
- Order triangulations lexicographically by angle vector.

Theorem: The Delaunay triangulation appears last.

Delaunay Triangulation of P

Proof:

- When we flip an edge, the angle vector increases.
- There are only finitely many triangulations with these vertices.

Delaunay Triangulation of P

Proof gives an algorithm:

- Start with any triangulation
- Flip edges until Delaunay

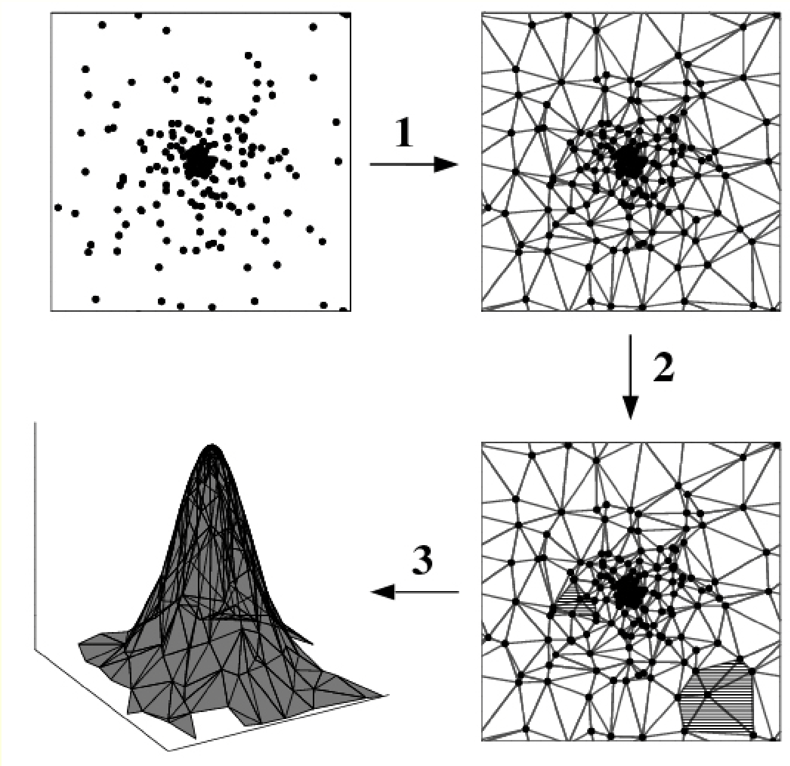
Incremental algorithm - faster

- Add points in random order.
- Maintain the Delaunay triangulation of current set of points.
- Runs in O(nlog(n)), n= number of vertices of T
- Can't be improved because it can be used to sort.
- **The Basic Algorithms and Combinatorics in Computational Geometry notes.**

http://graphics.stanford.edu/courses/cs268-09-winter/notes/basic.pdf

Applications of Delaunay Triangulation

- Meshing in finite element methods
  - Angle guarantee
  - fast algorithms
- Delaunay tessellation field estimator
  - Astrophysical applications



Distance Functions

Object Matching Queries

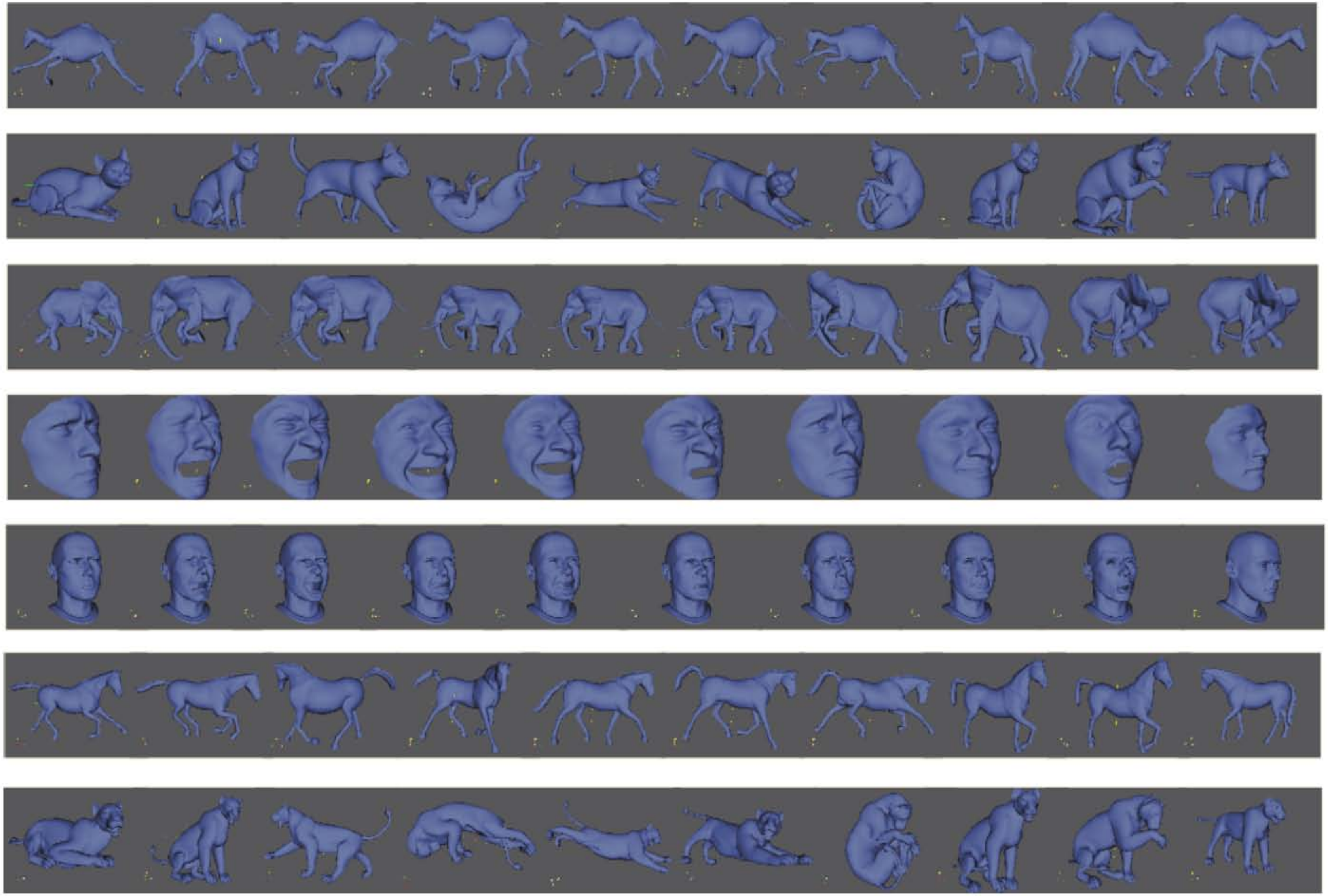
- assume you have database $\mathcal{D}$ of objects.
- assume $\mathcal{D}$ is composed by several objects, and that each of these objects belongs to one of n classes $C_1, \ldots, C_n$.
- imagine you are given a new object o, not in your database, and you are asked to determine whether o belongs to one of the classes. If yes, you also need to point to the class.
- One simple procedure is to say that you will assign object o the class of the *closest* object in $\mathcal{D}$:

$$\text{class}(o) = \text{class}(z)$$

$$\text{where } z \in \mathcal{D} \text{ minimizes } \mathbf{dist}(o, z)$$

- in order to do this, one first needs to define a notion **dist** of *distance* or *dis-similarity between objects.*

Useful for Object/Shape Classification

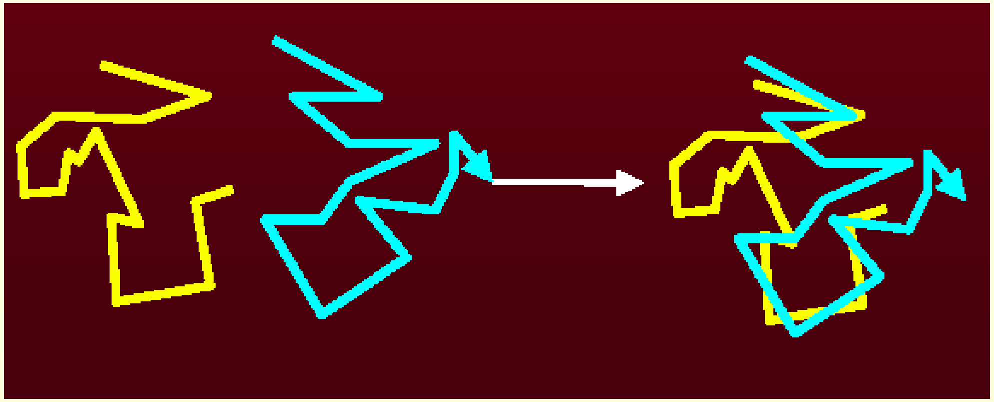


More Shape Similarity Methods

- Hausdorff distance
- Fréchet distance
- Morphing metrics

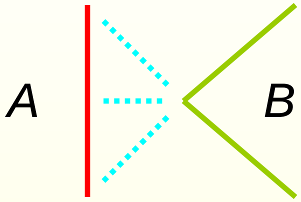
Proteins are defined as having a common fold if they have the same major secondary structures in the same arrangement and with the same topological connections

(SCOP)

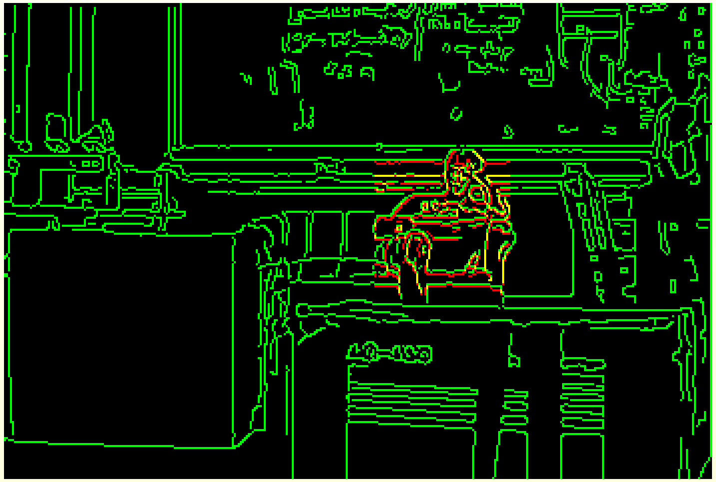


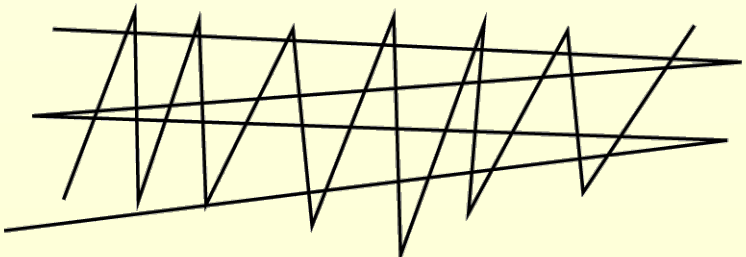
Hausdorff Distance

Many-to-many correspondences can be simpler ...



[Huttenlocher *et. al*., 93]





Hausdorff Definition

We are two point sets $A = \{a_1, a_2, ..., a_n\}$ and $B = \{b_1, b_2, ..., b_m\}$ in E^2. The one-sided Hausdorff distance from A to B is defined as:

$$\tilde{\delta}_H(A,B) = \max_{a \in A} \min_{b \in B} ||a - b||$$

The (bidirectional) Hausdorff distance between A and B is then defined as:

$$\delta_H(A,B) = \max\left(\tilde{\delta}_H(A,B), \tilde{\delta}_H(B,A)\right)$$

For fixed A and B, it can easily be computed in time $O((n+m)\log(n+m))$

Hausdorff Variations

- Order statistics – use percentile max (say the 90% largest distance from *A* to *B*) to avoid undue impact of outliers (fractional Hausdorff)
- Typically, one of the sets (say *B*) may be moved by a transformation group *G*

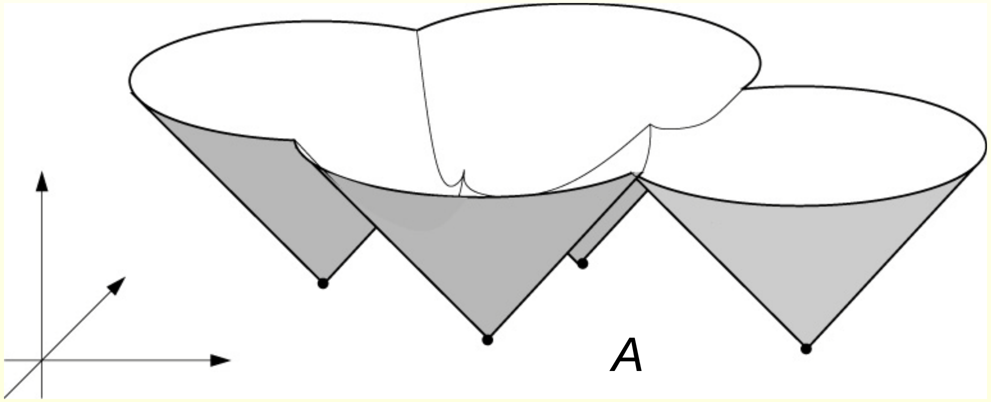
$$\tilde{\delta}_{H,\mathcal{G}}(A,B) = \min_{T\in\mathcal{G}} \max_{a\in A} \min_{b\in B} ||a - T(b)||$$

- Both “vector” and “raster” methods can be used

Computing Hausdorff

In the plane, vector form, under translation ...

The Voronoi surface of A, a piecewise conical surface



A lower envelope surface

$$d(x) = \min_{a \in A} \|x - a\|$$

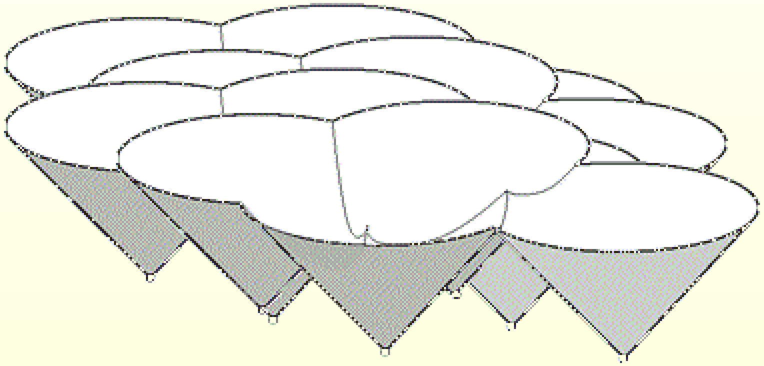
Translate *B* by *t*

$$\delta_b(t) = \min_{a \in A} \|a-(b+t)\| = \min_{a \in A} \|(a-b)-t\| = d_{-b}(t)$$

Computing Hausdorff, II

$$f(t) = \tilde{\delta}_H(B + t, A) = \max_{b \in B} \delta_b(t)$$

Upper envelope of *m* Voronoi surfaces
A-b_1, A-b_2, ..., A-b_m

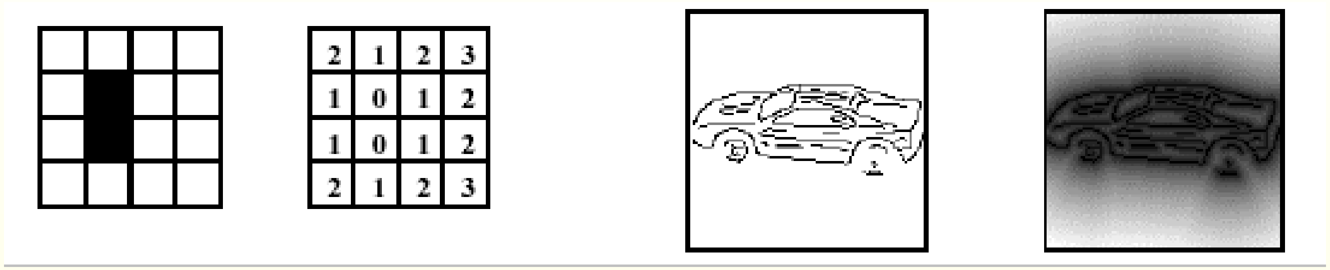


Can be done in time
$O(nm(n+m)\ \mathrm{polylog}(n+m))$

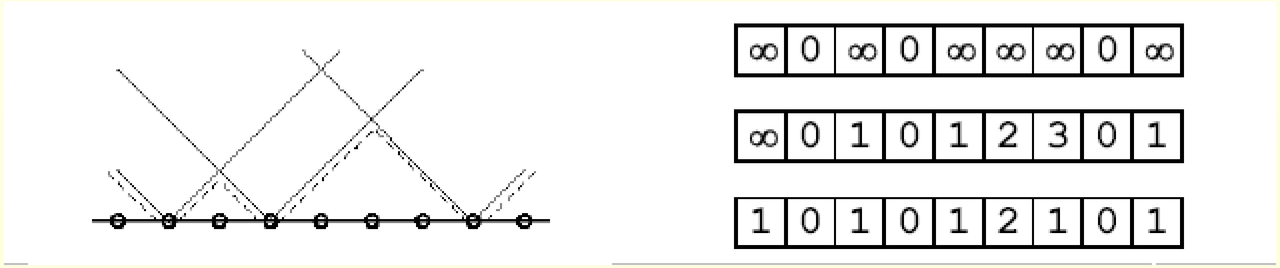
The amount of computation gets out of hand when we allow rotations and go to 3-D.

Raster Hausdorff

Distance transforms computed on a grid



(fast marching, level sets, ...)



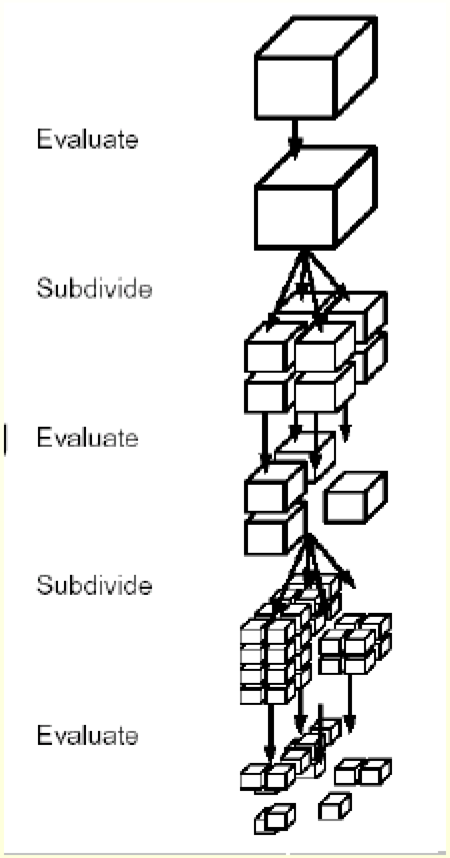
A 1-d example

Fast Hausdorff Search

- Branch and bound hierarchical search of transformation space
- Consider 2-D transformation space of translation in *x* and *y*
  - (Fractional) Hausdorff distance cannot change faster than linearly with translation
    - Similar constraints for other transformations
  - “Quad-tree” decomposition, compute distance for transform at center of each cell
    - If larger than cell half-width, rule out cell
    - Otherwise subdivide cell and consider children

Fast Hausdorff Search, II

- Guaranteed (or admissible) search heuristic
  - Bound on how good answer could be in unexplored region
    - Cannot miss an answer
  - In worst case won't rule anything out.
  - In practice rule out vast majority of transformations
- In practice rule out vast majority of transformations
  - Can use even simpler tests than computing distance at each cell center

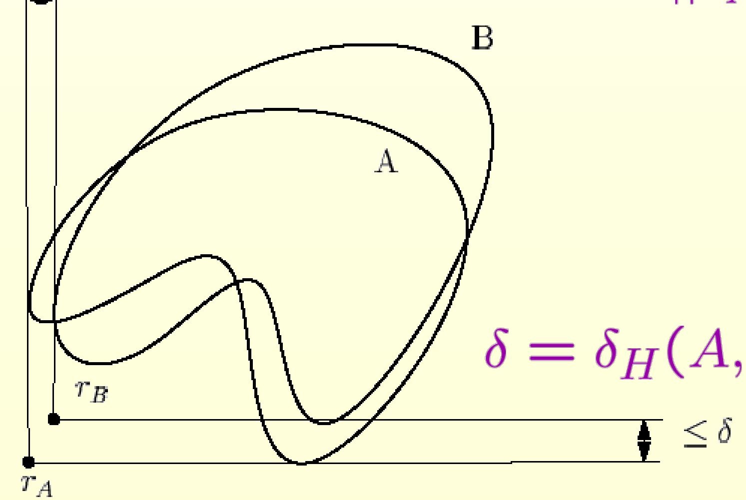


Reference Points

We match shapes by aligning certain well-chosen reference points. Such schemes can give constant-factor approximations to the Hausdorff distance [Alt *et. al.*, 91].

Example: approximate Hausdorff in 2-D under translations by matching lower left corner of bounding box

optimal alignment



$$||r_A - r_B|| \leq \sqrt{2}\delta$$

$$\begin{aligned} \delta_H(A, B') &\leq \delta_H(A, B) + \delta_H(B, B') \\ &\leq (\sqrt{2} + 1)\delta \end{aligned}$$

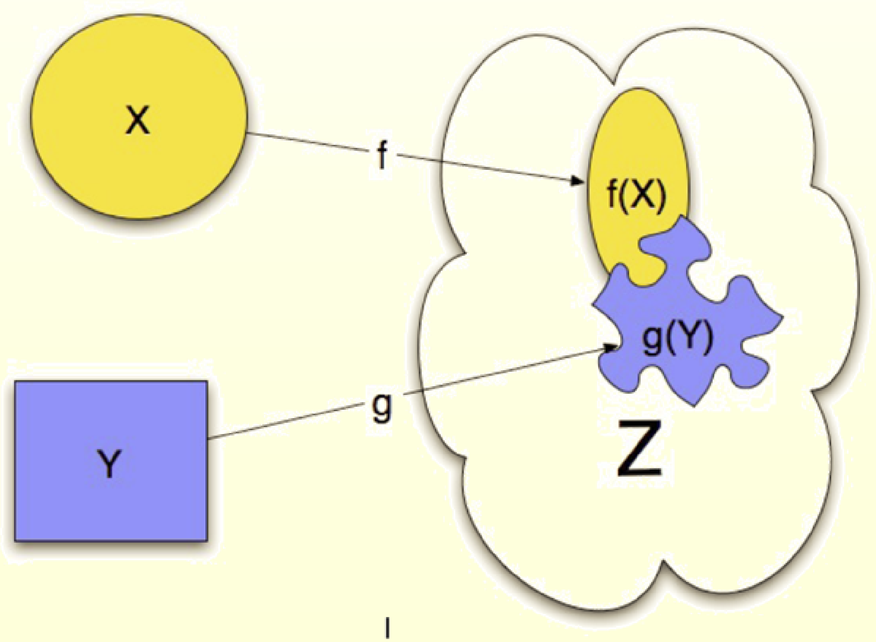
$$\delta = \delta_H(A, B)$$

Can be improved by local resampling

Possible for rigid motions, etc.

Gromov-Hausdoff Distance

$$d_{\mathcal{GH}}(X,Y) = \inf_{Z,f,g} d_{\mathcal{H}}^{Z}(f(X), g(Y))$$



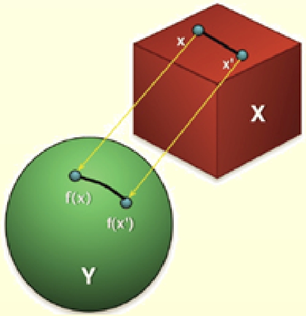
Gromov-Hausdorff Alternate Form

For compact spaces (X, d_X) and (Y, d_Y) let

$$d_{\mathcal{GH}}^{(2)}(X,Y) = \frac{1}{2} \inf_R \max_{(x,y),(x',y') \in R} |d_X(x,x') - d_Y(y,y')|$$

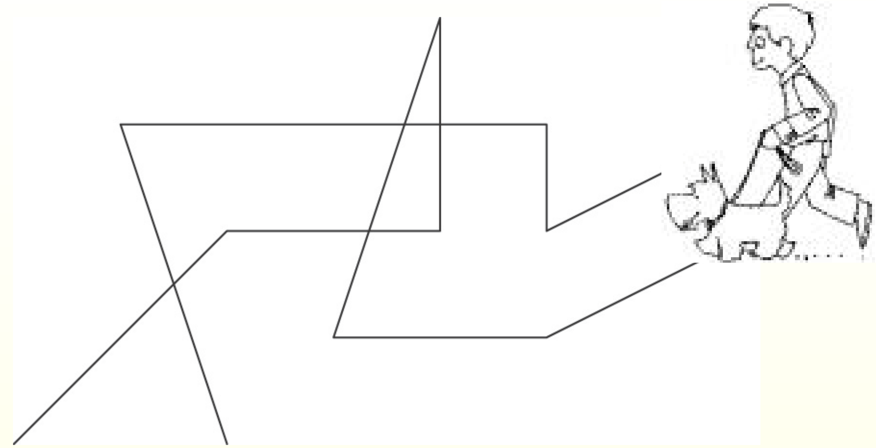
We write, compactly,

$$d_{\mathcal{GH}}^{(2)}(X,Y) = \frac{1}{2} \inf_R \|d_X - d_Y\|_{L^\infty(R \times R)}$$



Hard to compute …

Fréchet Distance

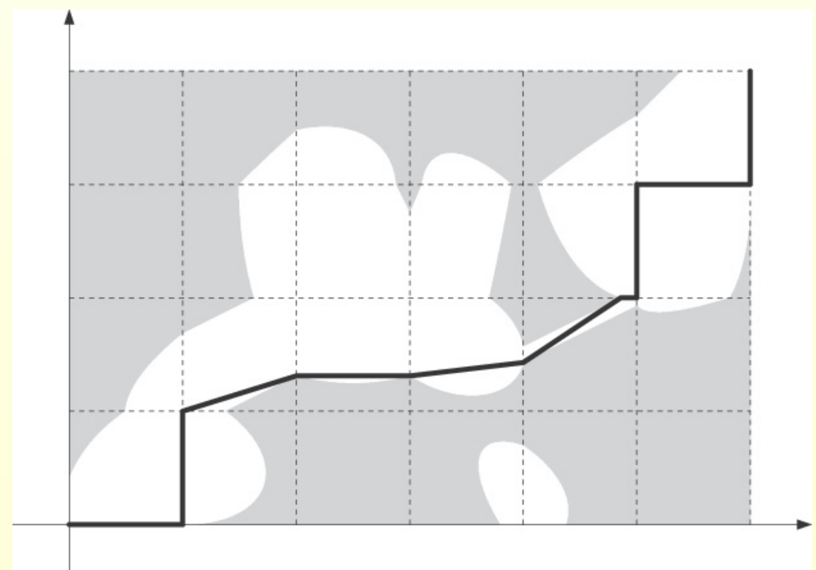


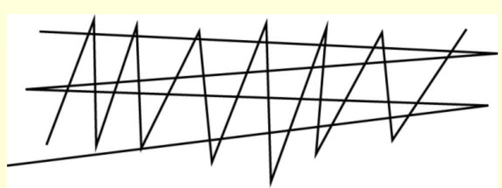
$$\delta_F(f,g) = \inf_{\alpha,\beta} \max_{t\in[0,1]} ||f(\alpha(t)) - g(\beta(t))||$$

From a decision problem to
an optimization problem.

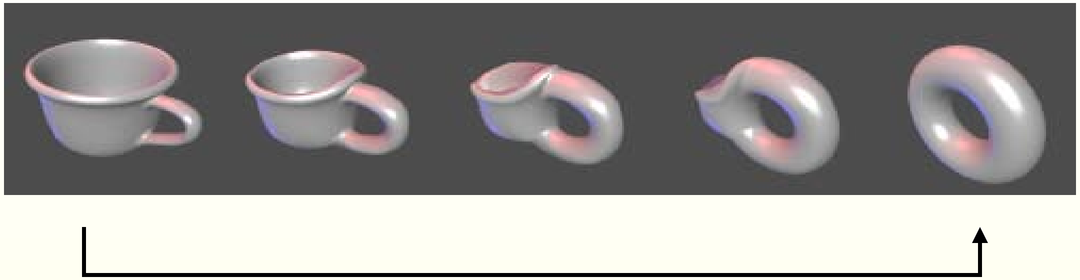
Guess and verify ...

O(*mn* log(*mn*))





Morphing Distance



minimum cost transformation
from *A* to *B*