Special Topics in Computational Geometry

Valentin Polishchuk

Using material from Anna Lubiw, Joe Mitchell, Antoine Vingeron, Robert Ples, Godfried Toussaint, and others

Administrative

- Special Topics in CG. This is NOT a class in CG. What this course is not about:
  - Polygon triangulation, Art Gallery Problem, Arrangements and Intersection, Queries
  - VD, DT -- only briefly
- Self-contained
- Schedule:
  - 1st week: lectures
    - Day 1: Shortest paths
    - Day 2: CH, k-hull, alpha-shape, VD, DT, beta-skeleton; rotating calipers
    - Day 3: Decide on the projects. Multiple paths, corsets
  - 2nd week: project work
    - Project -- step towards thesis
    - Some projects are already listed on the webpage; some will be announced during the lectures
    - Ask about project details
  - Make-up class for Thu, May 21st
- Reminder: use the course feedback system
- Questions?

CG subject: geometric algs; ''algorithmic geometry``

- Algs: design, analysis, implementation
  - Design – informal
  - Analysis – can be skipped
  - Implementation (formal)
    - performs well -- noone needs analysis. Implementation comes first!

- Analysis: proving runtime and correctness (the alg does serve the purpose)
  - Must be formal. Level of formality?

- Intuitively obvious things are often hard to formally define and work with
  - Working with the intuitive concepts is fine
  - But they may be misleading
  - Must always know the formal definitions and be ready to go down to them

Intuitively obvious vs. Rigorously defined

- Especially true in geometry
  - History of geometry: fighting with axioms = formalizing the intuition
  - CG is no exception
  - Not to make things more complicated than they are, we will often use intuitive notions and statements
  - There are always formal concepts behind

- Polygon. Not easy to give definition. MathWorld

- Model of computation: Real RAM
  - Can do any operation on real numbers in constant time
  - Not too realistic, but not to make things more complicated
    - not to deal with the bit complexity models
- 2D

Definitions:

Chain, Polygon, Domain

Polygonal Chain (Path)

- A simple (i.e., non-self-intersecting) curve that consists of a sequence of straight-line segments
  - Segs are called links or edges
  - Segs endpoints are called vertices

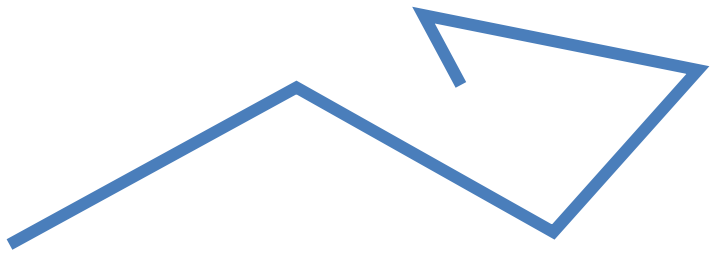
Small print: A curve is a function

Identify the curve with its image

Understand the definition in this sense

Stricter definition: simple piecewise-affine curve

Note: always assume there is a finite number of edges



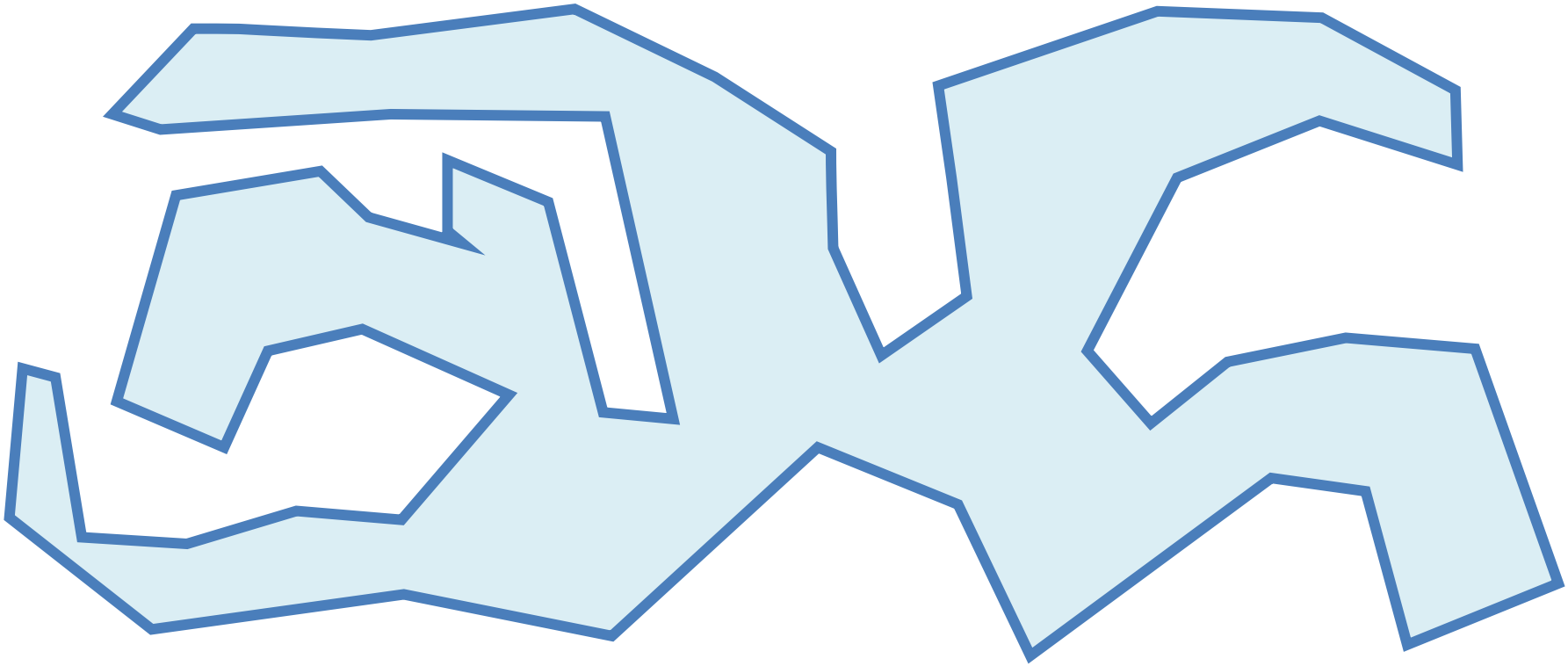
Simple Polygon

- Region bounded by a closed polygonal chain
  - Edges/vertices of the chain – edges/vertices

Note: assume open or closed as convenient

Note: Could define as a simply-connected open subset of the plane with piecewise-linear boundary

Simply-connected: Any loop is contractible. Homeomorphic to a disk. Any two paths with same start and end are homotopically equivalent.



Polygonal Domain

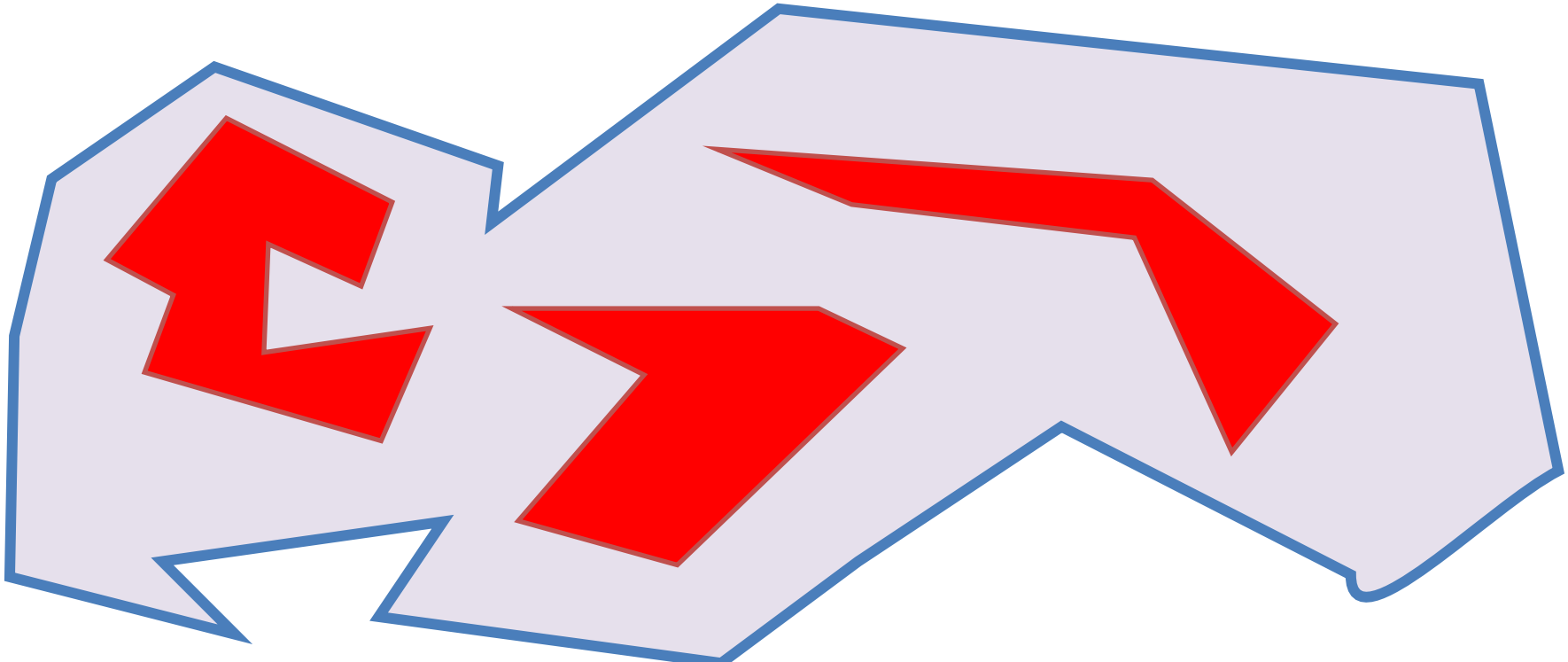
- A simple polygon -- the outer polygon of a domain
- Pairwise-disjoint simple polygons inside the outer polygon -- holes, or obstacles

A polygonal domain is the outer polygon minus the holes

– vertices/edges of the domain -- vertices/edges of the outer polygon and the holes

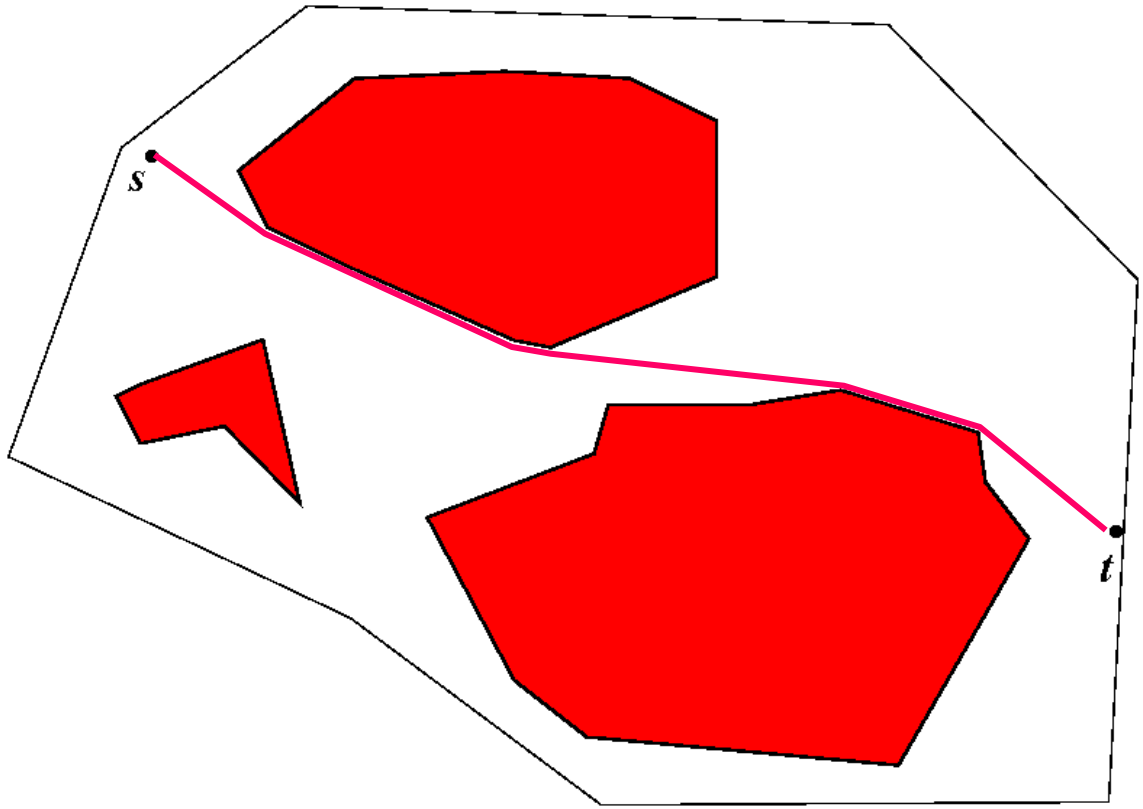
Note: simple polygon -- polygonal domain without holes

simple polygon -- polygon *without* holes, polygonal domain -- polygon *with* holes



Shortest Path problem

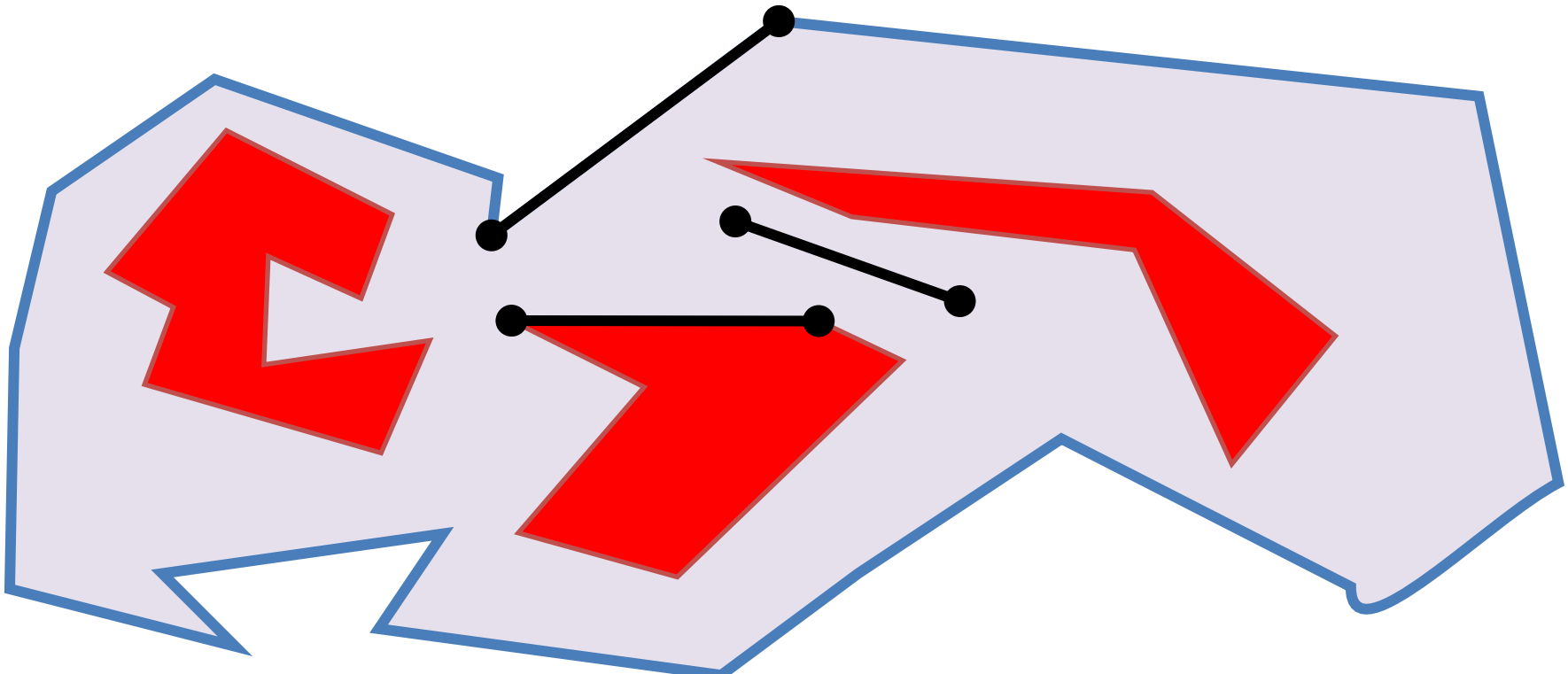
- Given:
  - Polygonal domain
  - Points s and t
  - (mini-holes)
- Find:
  - shortest s-t path
  - avoiding obstacles



Visibility

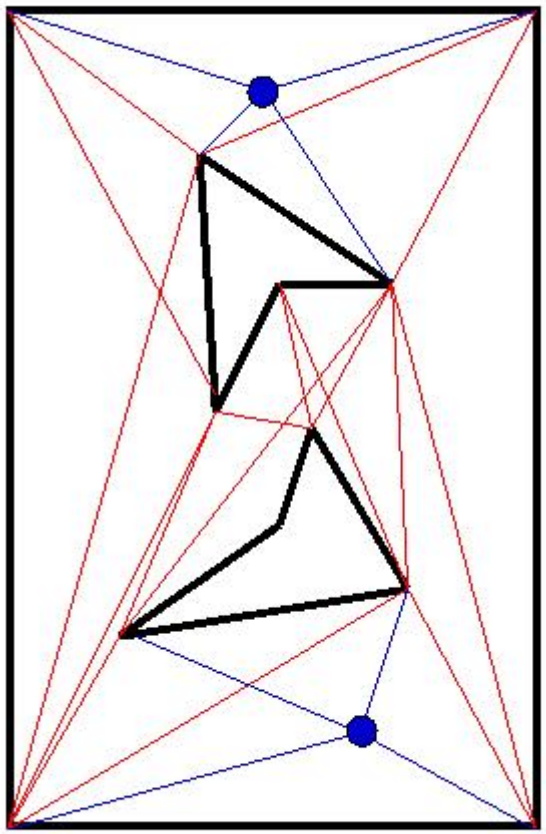
- Two points are (mutually) visible
  - segment between them belongs to the domain

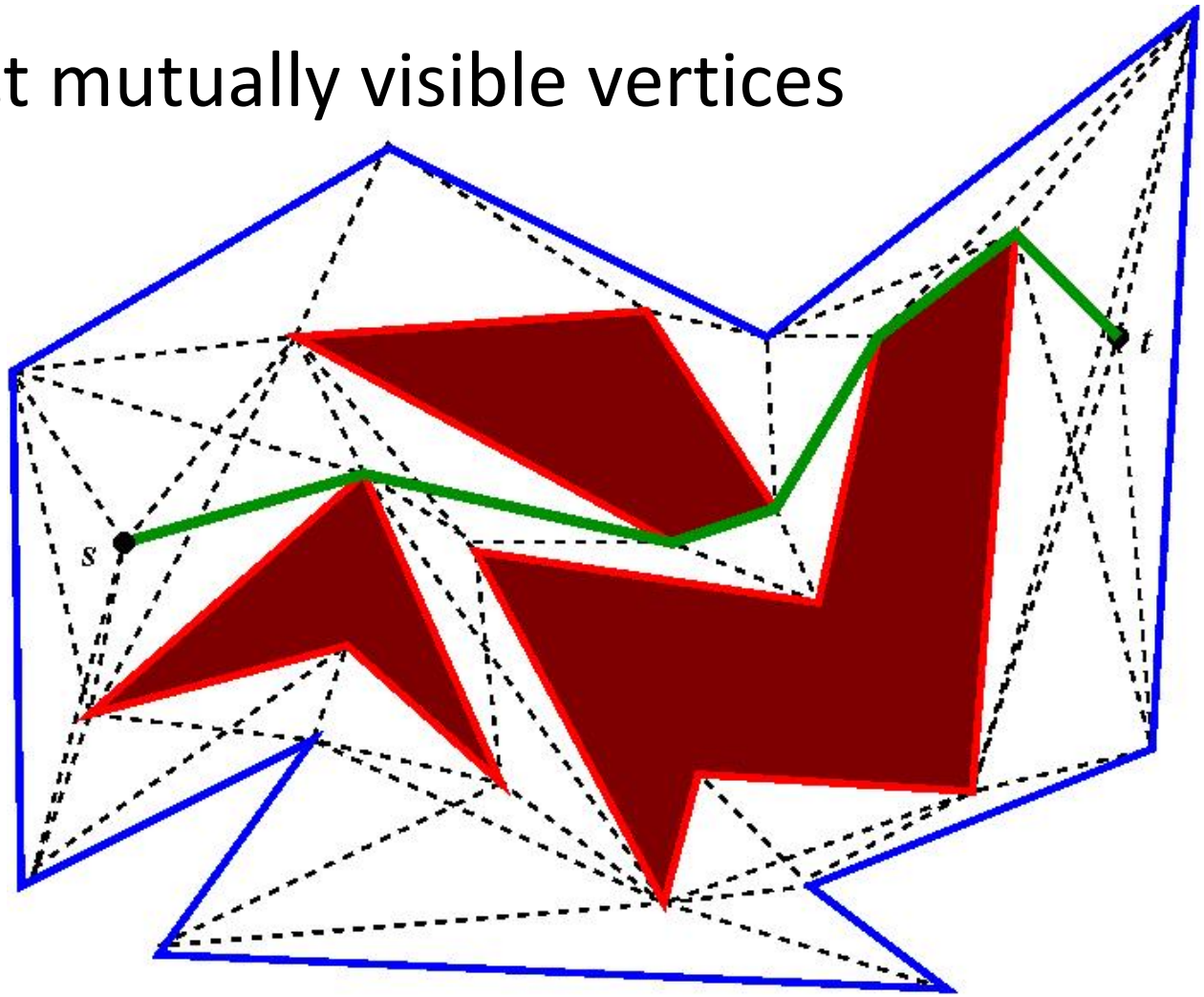
Note: assume outer polygon is closed, holes are open -> edge endpoints see each other



Visibility Graph

- vertices = vertices of P
- edges connect mutually visible vertices





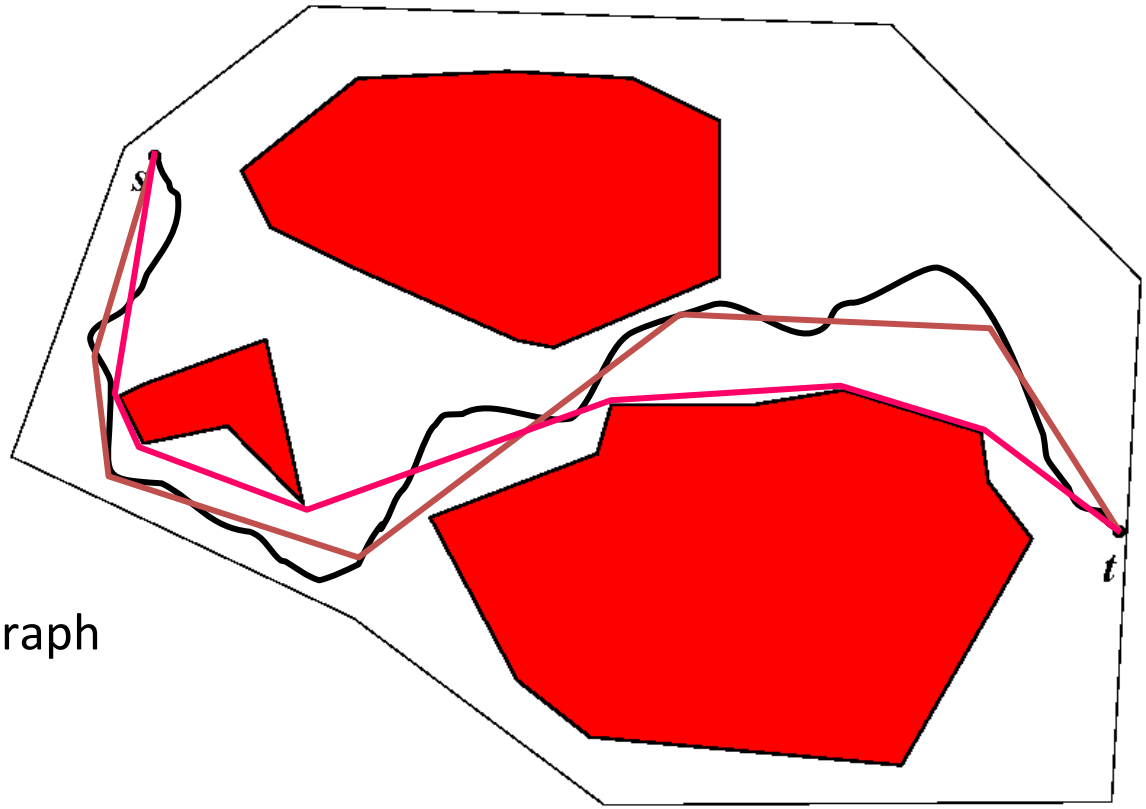
Shortest Path Properties

- Polygonal path
- Bends at vertices

  connects *visible* vertices

  Pf: see O'Rourke's book

- Shortest Path
  - a path in the visibility graph



Algortihm

- Build visibility graph
- Run Dijkstra
  - get the SP
- Running time $O(n^2)$

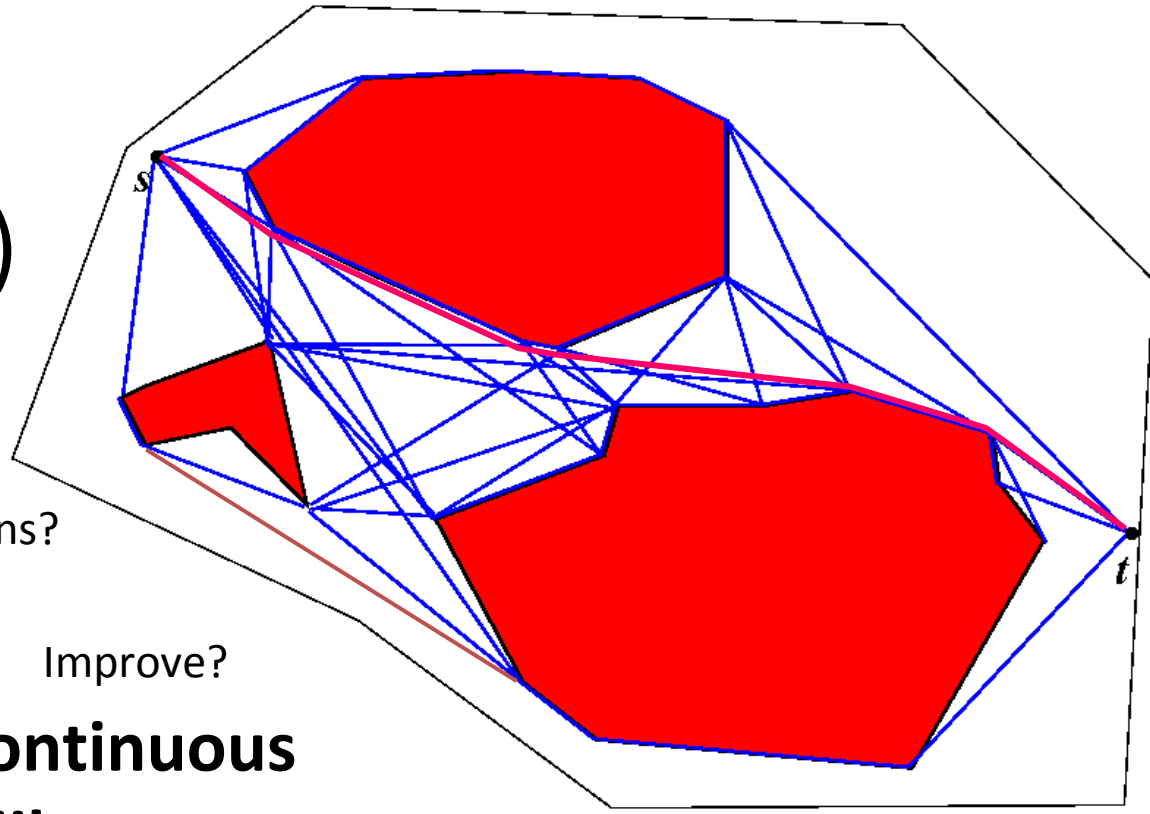
  Dijkstra on G=(V,E):

  $O(|E| + |V| \log |V|)$
- Domain

  n vertices
- Visibility graph

  $|V| = \Theta(n)$

  $|E| = O(n^2)$ edges



Questions?

Improve?

Continuous Dijkstra

[Mitchell'86]

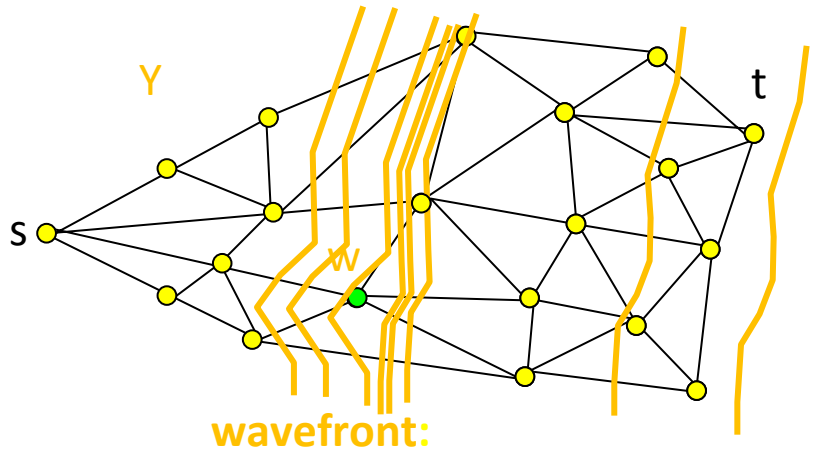
Dijkstra's Algorithm on Graph

- Maintain

  Y – vertices "reached" by SP

  - Initially Y := s

  d(v) – length of SP



wavefront:

bd between reached and not by certain time

- Until t ∈ Y

  w := arg min $_{u \text{ in } V \setminus Y}$ d(u)

  Y := Y + w

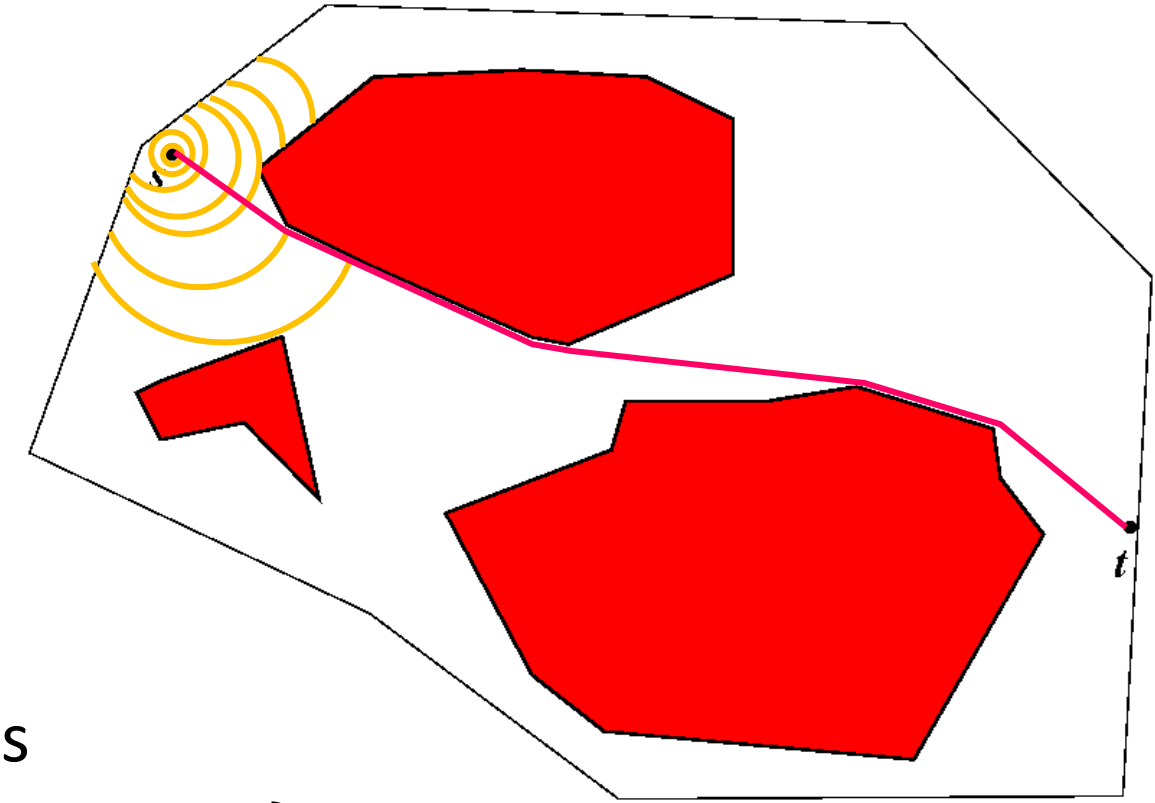
  For all u in V \ Y , adjacent to w

  d(u) := min{ d(u), d(u) + length(w, u) }

"Wave propagation"

Continuous Dijkstra

- Wave propagation
  - starts from s
  - travels at speed 1
- Wavefront at τ
  - bd of what's reached by τ
- Wavefront changes
  - collision with vertex or edge of P



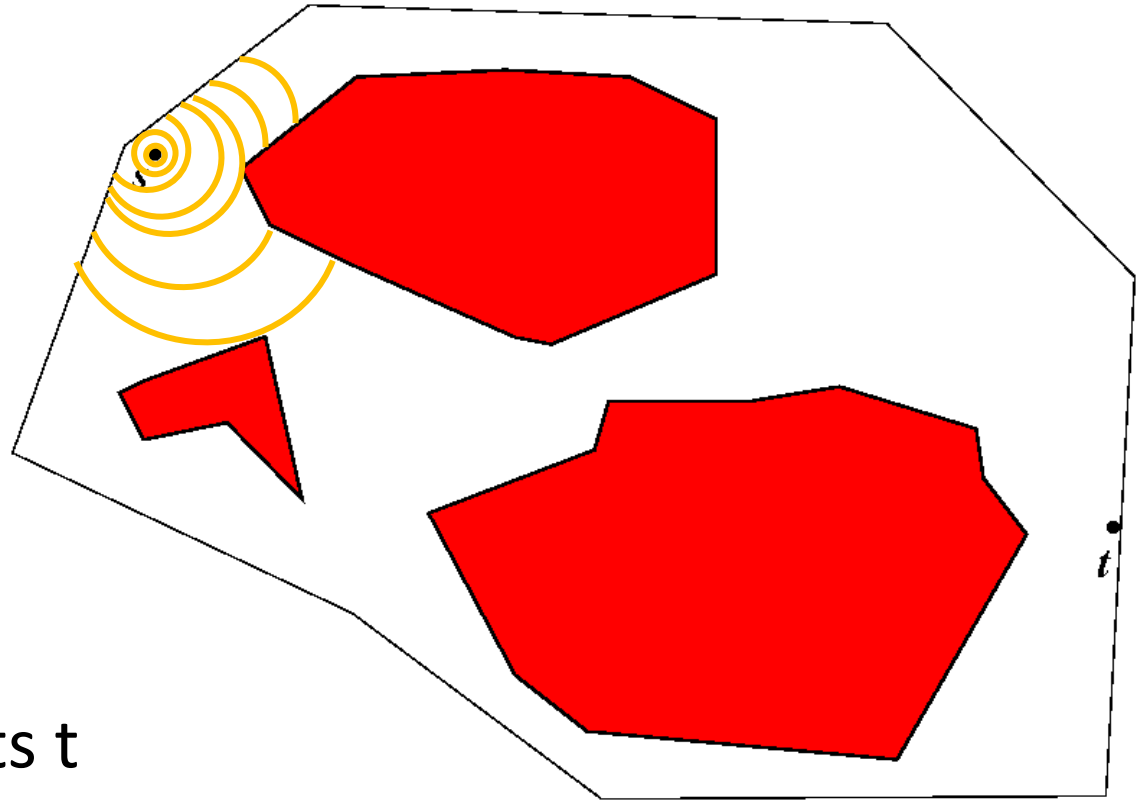
} "Events", O(n) $\Rightarrow$ SP in $o(n^2)$

Determine next event, update the wavefront

o(n) / event

Grass Fire Analogy

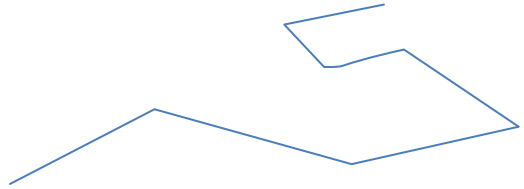
- Domain – grass
- Wave – fire
  - holes don't burn
- Ignite s

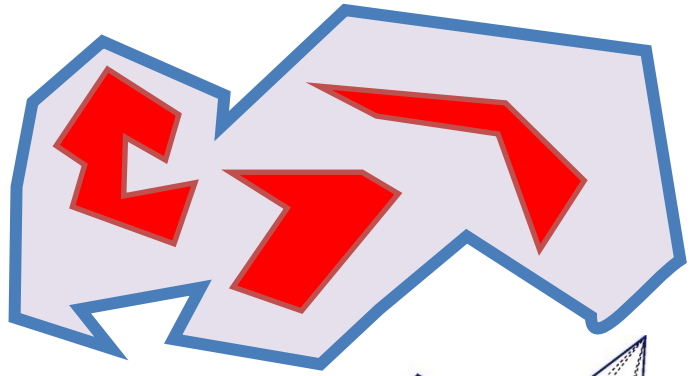


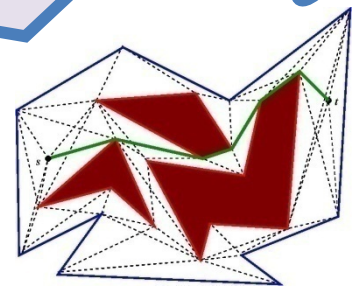
| SP to t | =

time when fire hits t

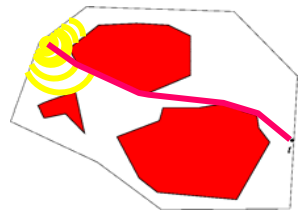
Summary







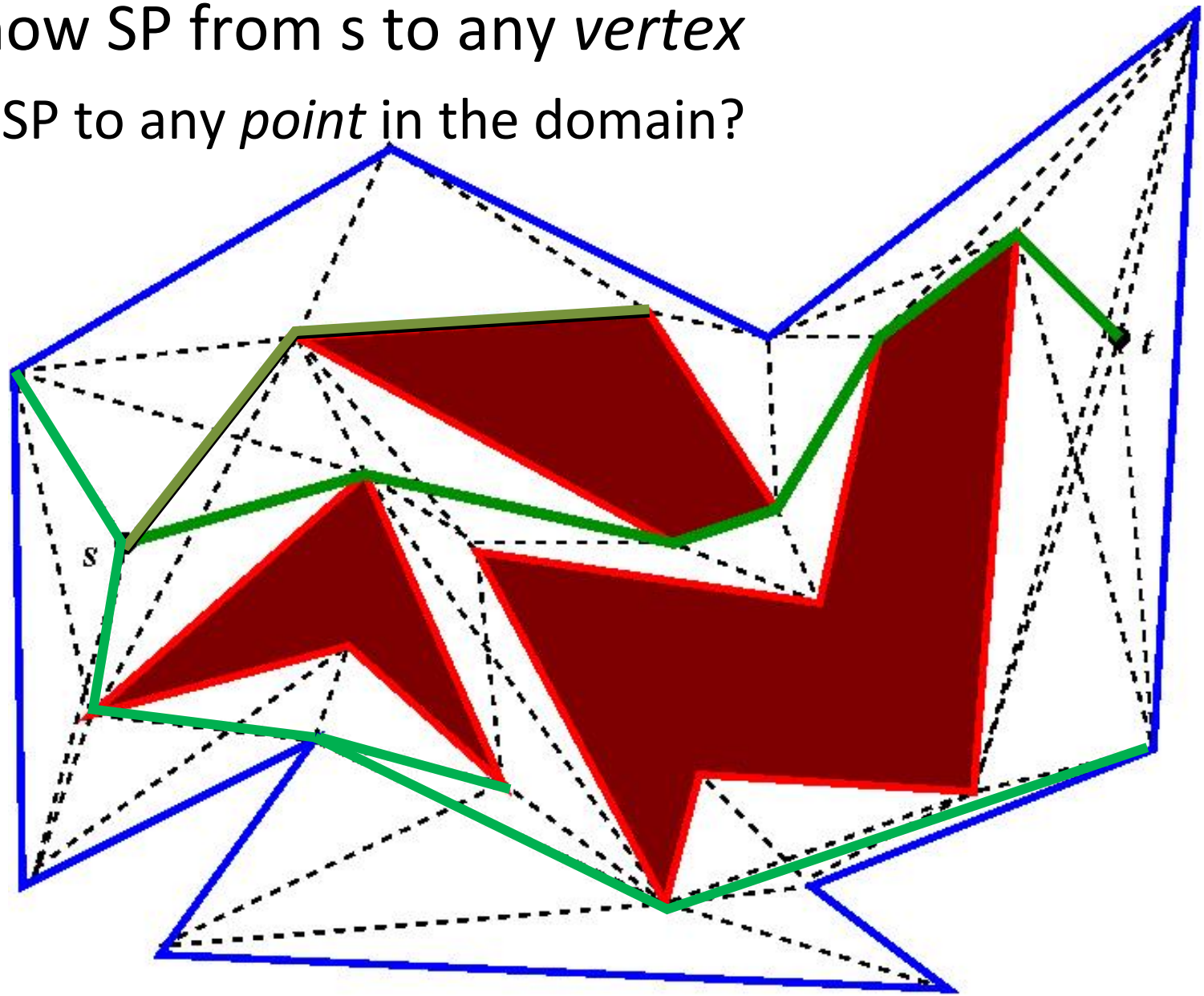
- Polygonal path (chain)
- Simple polygon
- Polygonal domain
- Visibility graph
- Shortest path
  - visibility graph
    - Standard trick: reduce to a known graph problem
    - Geometric problem -- continuum of feasible solutions
    - Graph problem -- countable number
      - known algs
  - continuous Dijkstra



Shortest Paths tree

Draw graph SPT

- Know SP from s to any *vertex*
  - SP to any *point* in the domain?

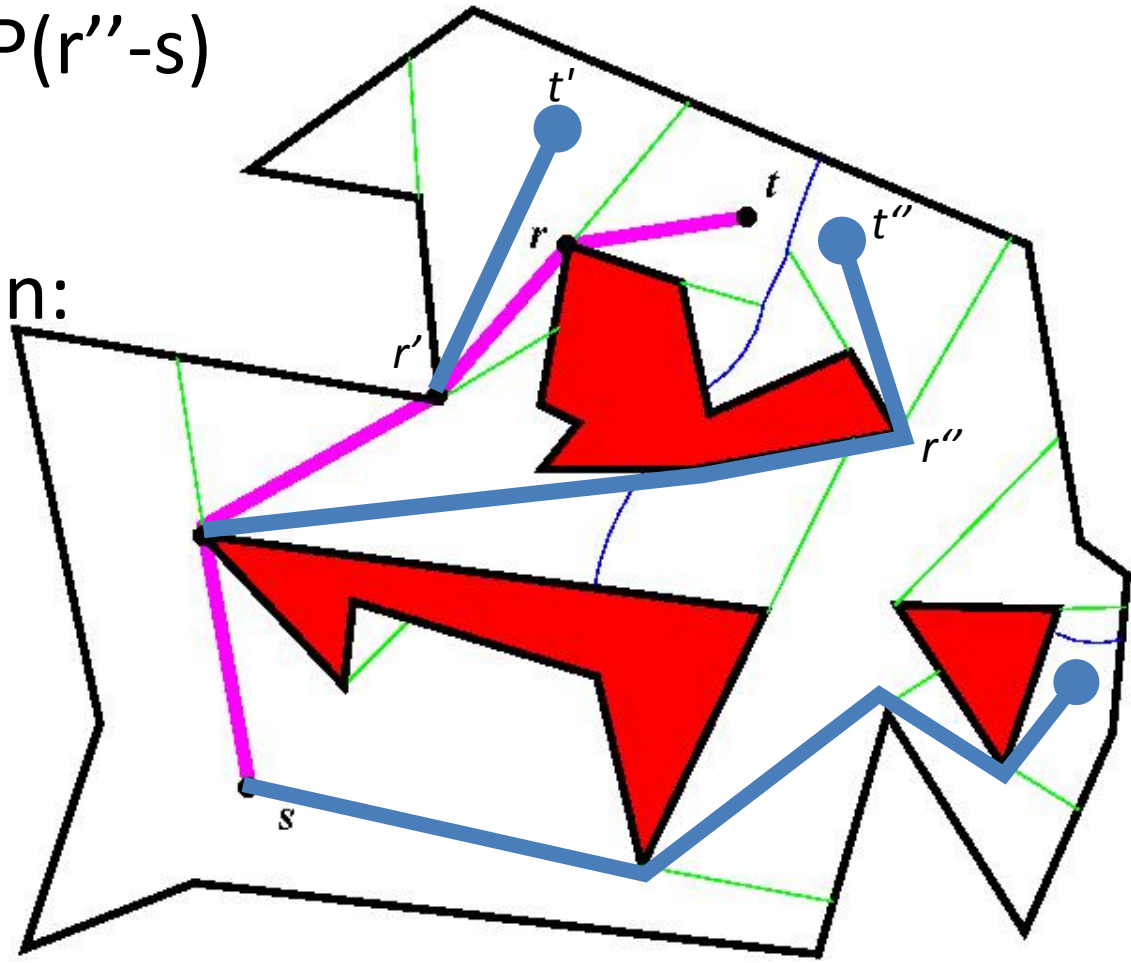


Shortest Path Map

- SP(t-s) = |tr| + SP(r-s)
- SP(t'-s) = |tr'| + SP(r'-s)
- SP(t''-s) = |tr''| + SP(r''-s)

- Solves query version:
  - report SP(p-s)
  - O (log n + # of links in SP)

Decomposition of the domain into cells within a cell SP(p-s) has the same first vertex



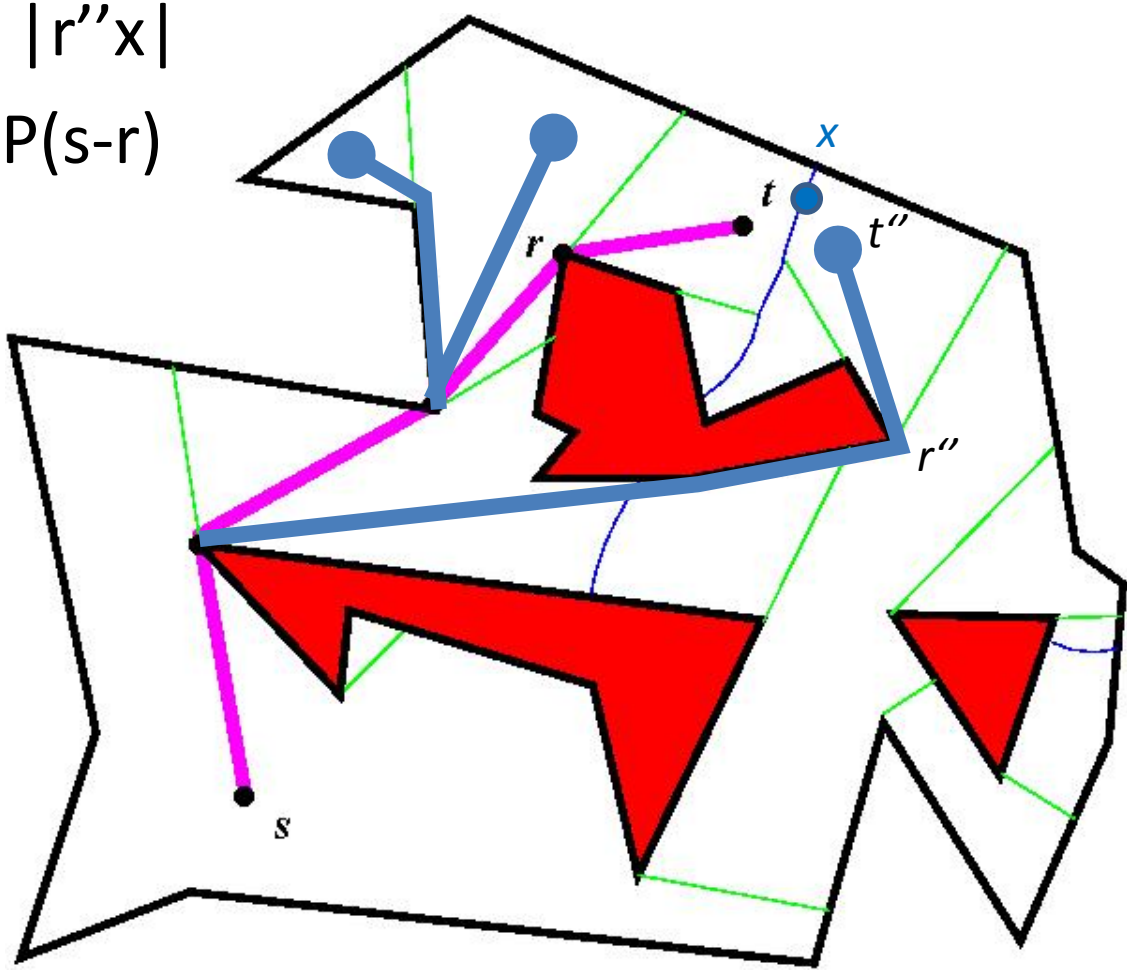
Cells boundaries

- Straight-line seg

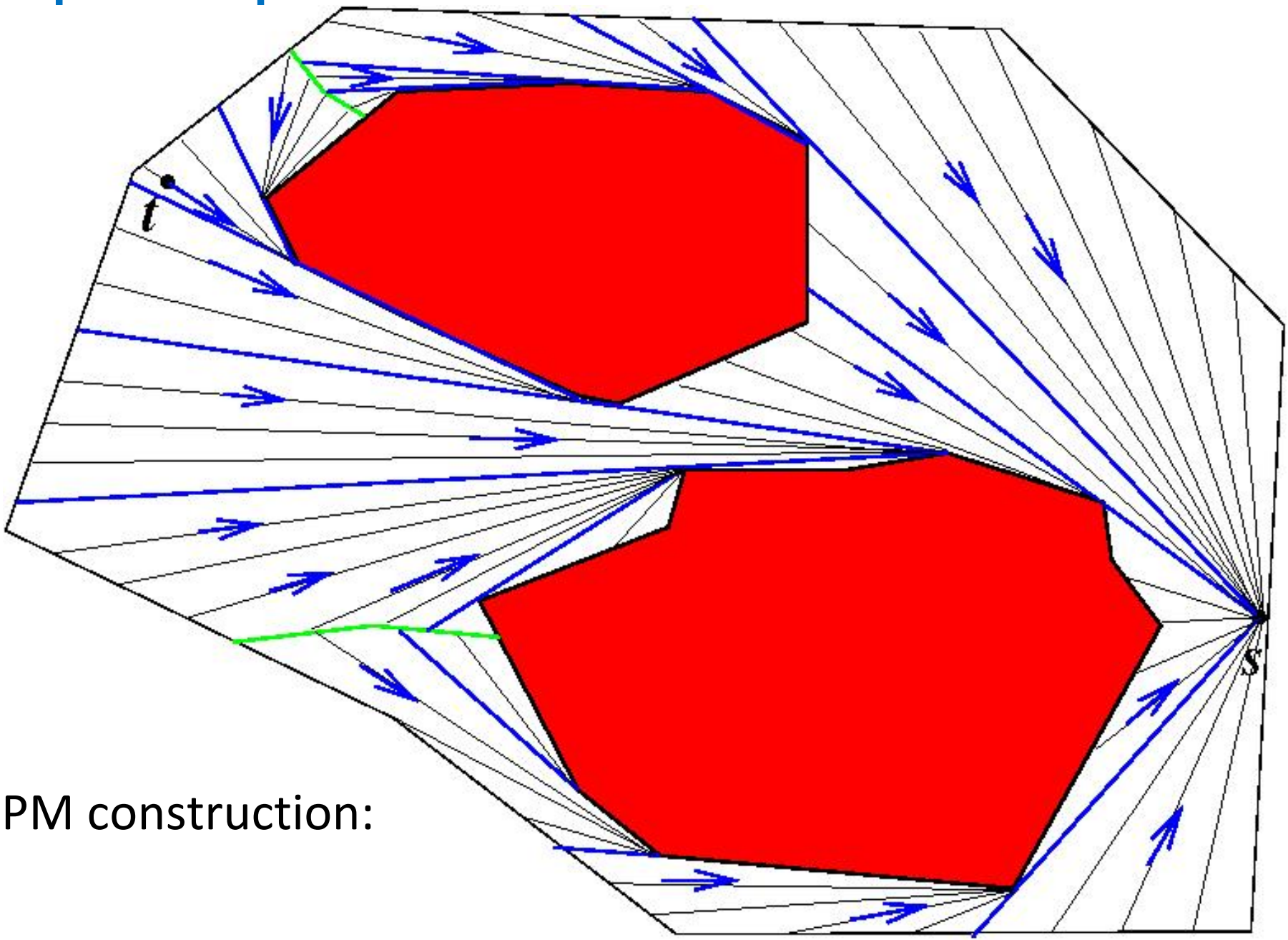
SP(s-r) + |rx|= SP(s-r’’) + |r’’x|

|rx|– |r’’x|= SP(s-r’’) – SP(s-r)

- Hyperbolic arc



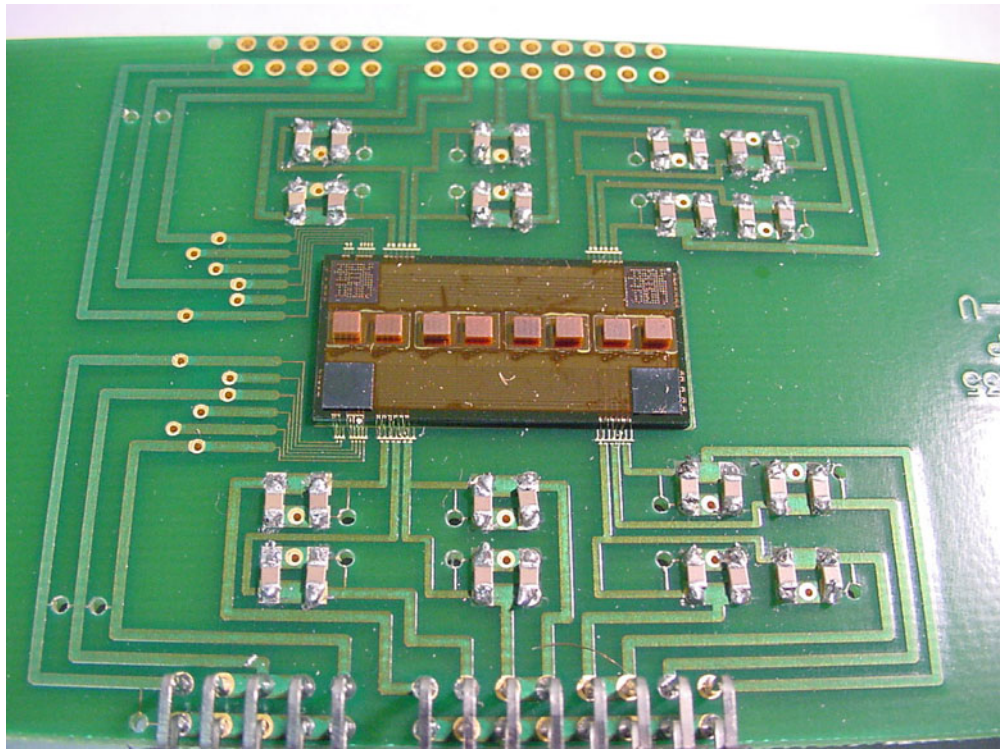
Superimpose Shortest Paths



SPM construction:

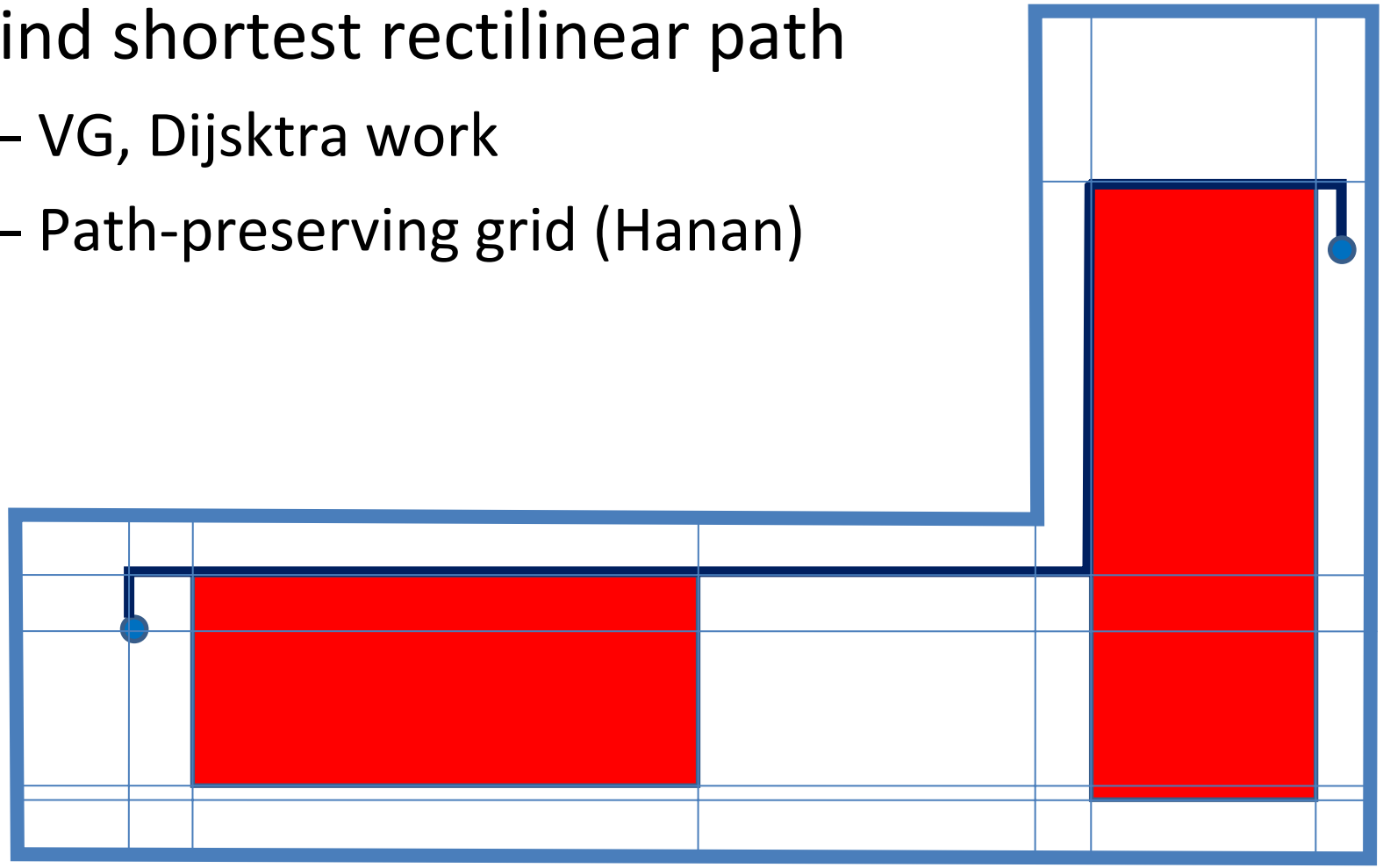
Rectilinear world

- Obstacles' edges are parallel to x or y axis
- Paths' edges are parallel to x or y axis
- Axis aligned path:
  - length = L_1 length



Rectilinear SP

- Given rectilinear domain, s and t
- Find shortest rectilinear path
  - VG, Dijsktra work
  - Path-preserving grid (Hanan)

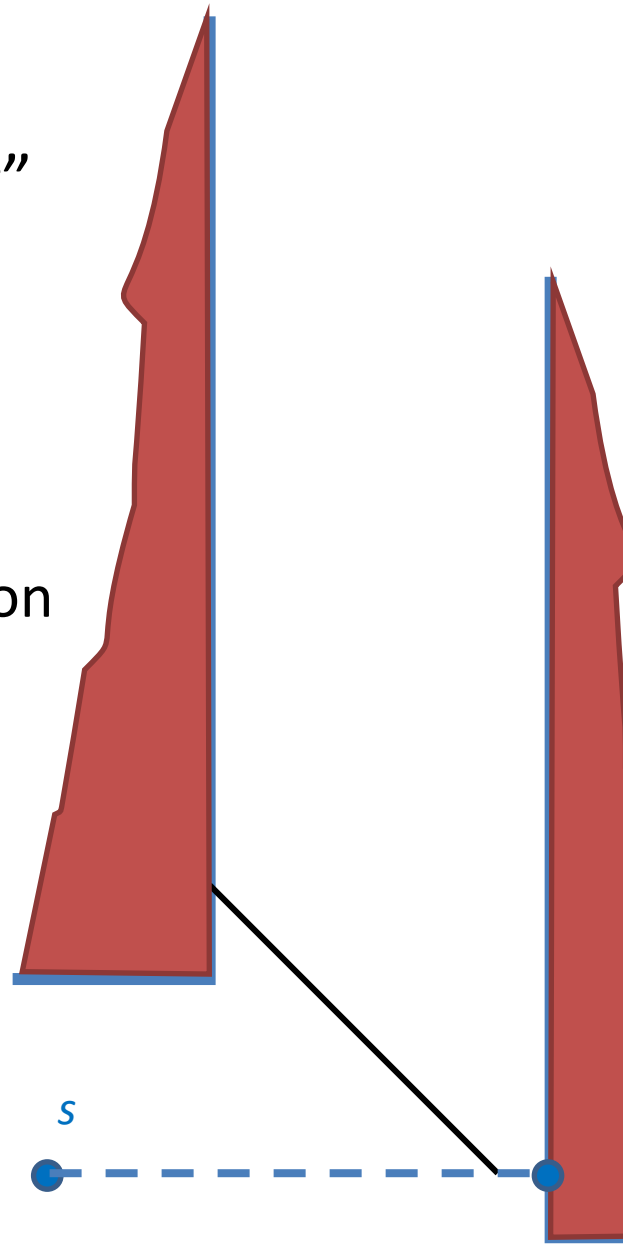


Rectilinear path: Continuous Dijkstra

- Propagate “wavefront” from s
- wavefront = points at the same L_1 dist from s
  - set of ±45° segments (wavelets)
- Stored in priority queue (*event queue)*

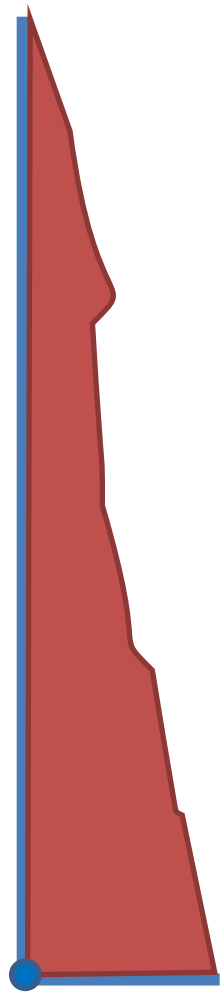
Info with Each Wavelet

- Orientation (NW, SW, NE, SE)
- Endpoints at initiation, before “dragging”
- Track rays (horizontal or vertical)
- Stop points of the track rays
  - obstacle points “hit” by the track rays
- Root
  - vertex responsible for segment propagation
- Contact list
  - obstacle edges segment touches
- Event position
  - position when the contact list changes
- Event point (stop point or vertex)
  - change in contact list at event position
- Event distance (event point from s)



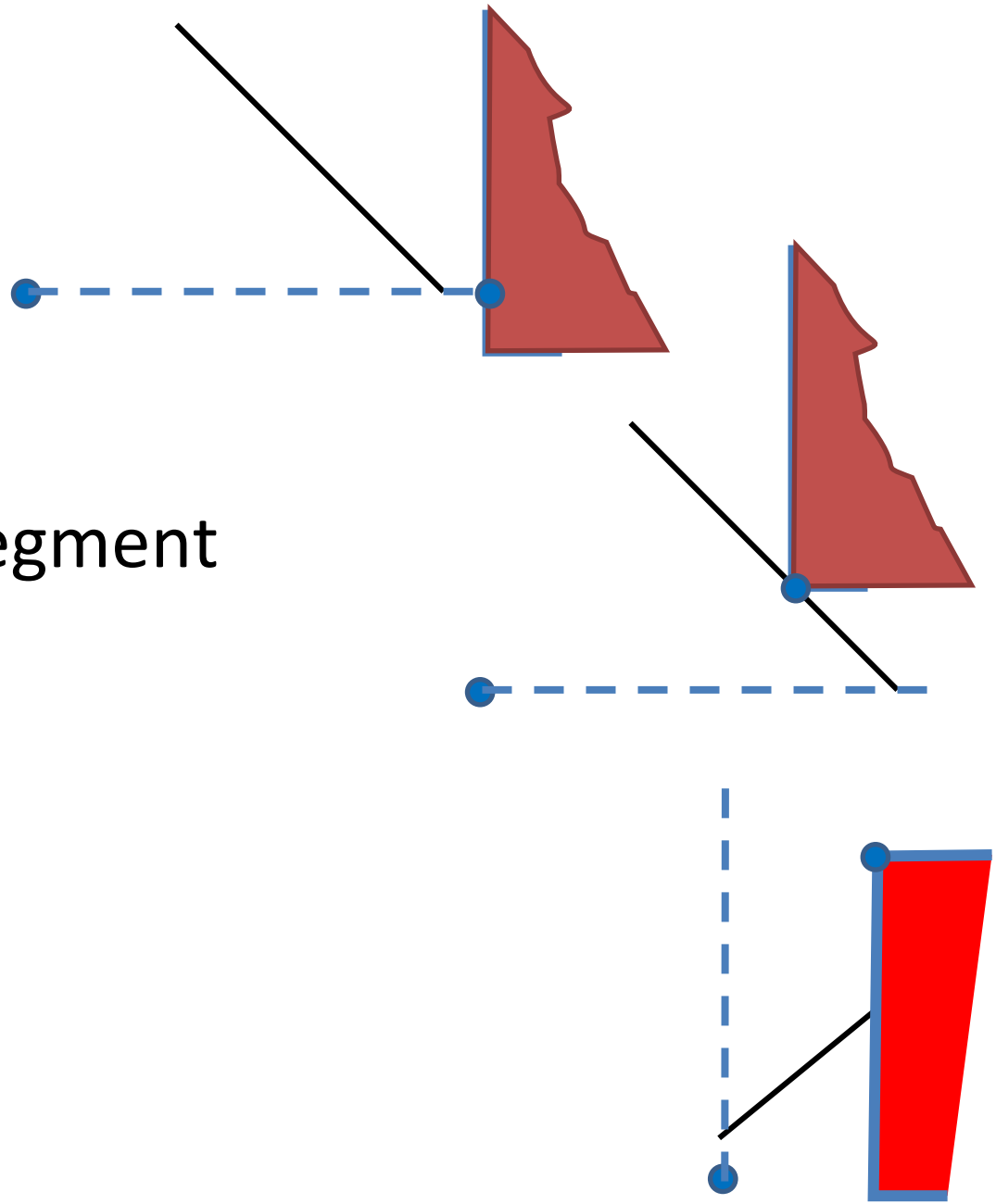
Info with Each Vertex

- Hit lists (NW, SW, SE, NE)
  - Roots of wavelets that have hit the vertex
    - e.g., NW-hit list – roots of NW wavelets
- Permanent label (initially ∞)
  - shortest path from s
- Pointer (initially NIL)
  - parent in the Shortest Path Tree

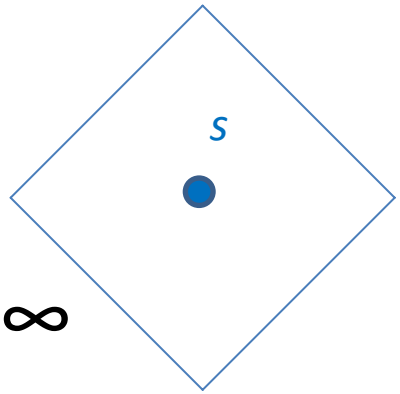


Events

- Stop point
- Interior to segment
- Vertex

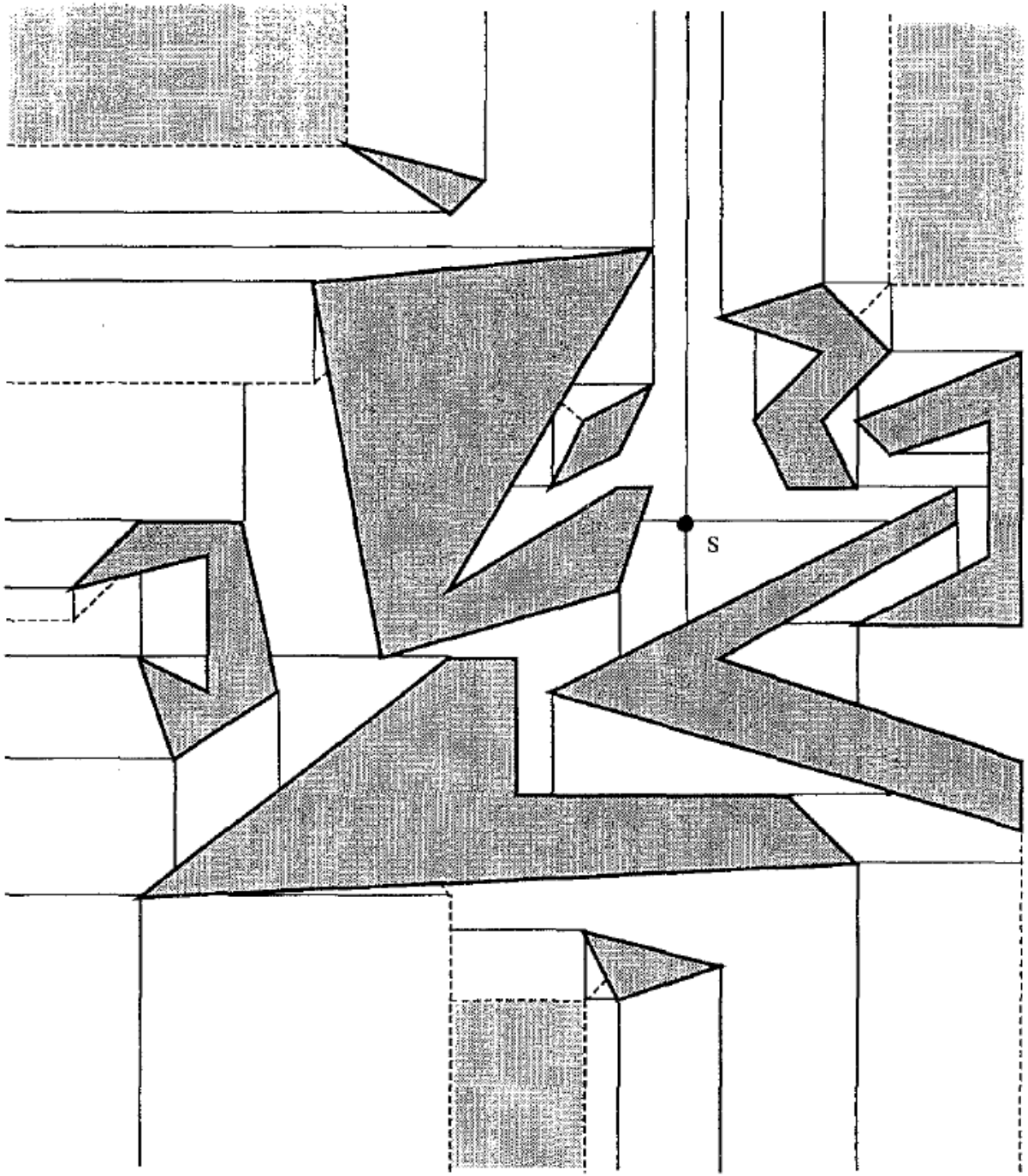


Algorithm



- Initialize
- While exists segment with event dist < ∞
  - Pop the queue

- Why subquadratic?
  - Clipping


S

Link distance

- Find a path with minimum number of links

Projects:

- An example of wave propagation in the rectilinear case
- Survey of the data structure to find minimum-link rectilinear path
- Survey: Bit complexity of minimum-link paths

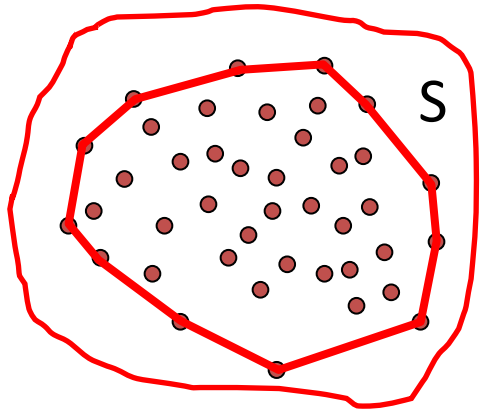
Shape approximation

Assumptions

- Non-degenerate position
  - no 3 points colinear
  - no 4 points cocircular

Convex hull

- Basic, fundamental notion in many disciplines
- "Pulled-taught string around S"



- Convex set, Convex combination

- Definitions:

  1: Smallest convex set containing S

  (smallest: min-area, min-p, or just minimal)

  2: Intersection of all convex sets containing S

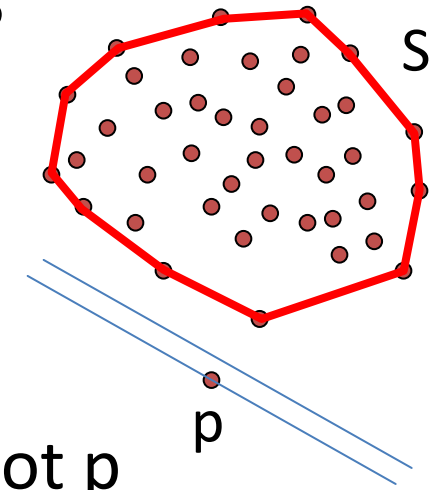
  3: Points -- vectors; set of all convex combinations of points in S

Not constructive definitions

How “deep” a point is

- Formalize via Definition 2:

  Intersection of all convex sets containing S



- p, line l through p

  S is on one side of l, then p is not on CH.

  Proof: Move l, a halfplane contains S but not p

- Halfplane depth d(p) w.r.t. S

  min # of pts of S in l+ for a line l through p

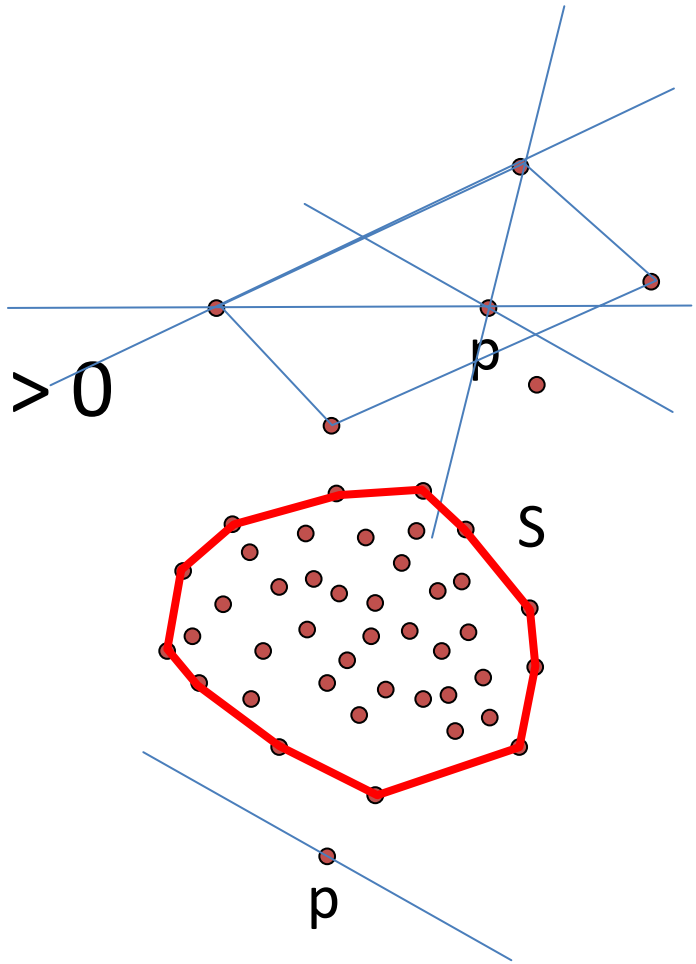
  Small print: l+ -- closed halpfpane.

- d(p) = 0 -> p not in CH(S)

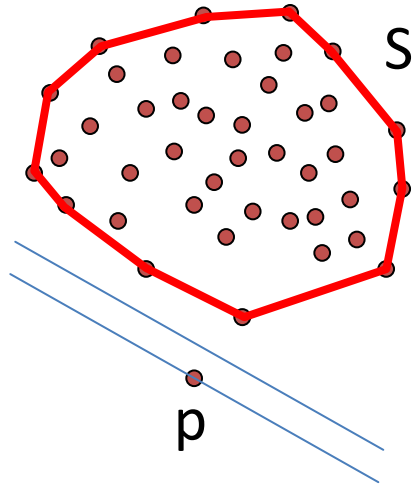
d(p) > 0 -> p in CH

- Definition 3: all convex combinations of points in S

- d(p) = 0 -> p not in CH(S)
- d(p) > 0 -> p in CH
- CH: points of halfplane depth > 0
- (Definition 4)



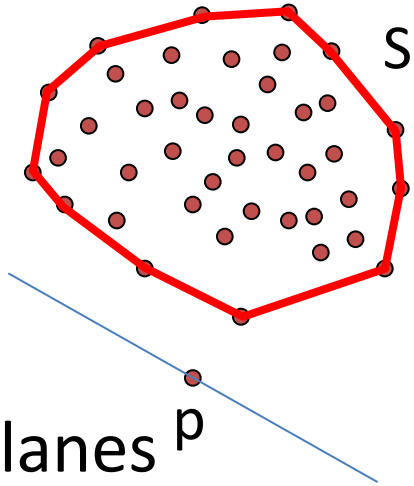
Empty halfplanes



- d(p) = 0 <-> p not in CH(S)
- Claim:

  Points of halfplane depth 0 =
  union of empty halfplanes
- *Proof*:
  - One way is easy:
    - d(p)=0 -> p is in a “witness” halfplane
  - The other is easy too:
    - p in empty halfplane l+ - >
    - take l' , parallel to l, through p
- CH: complement of the union of empty halfplanes
  - (Definition 5)

k-hull



- CH definitions 4 and 5:

  points of halfplane depth > 0

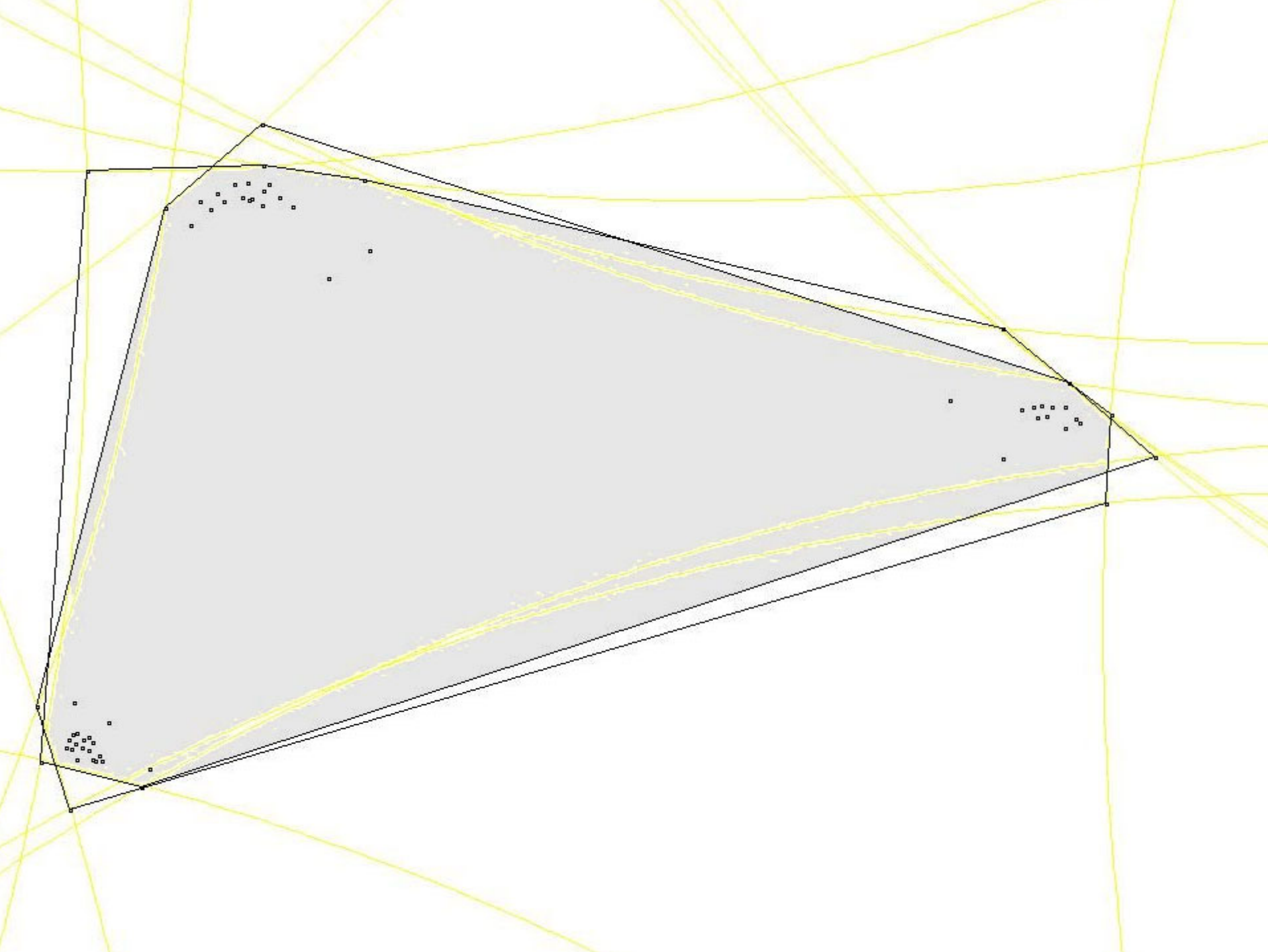
  complement of the union of empty halfplanes

- Generalize: k-hull

  points of halfplane depth > k

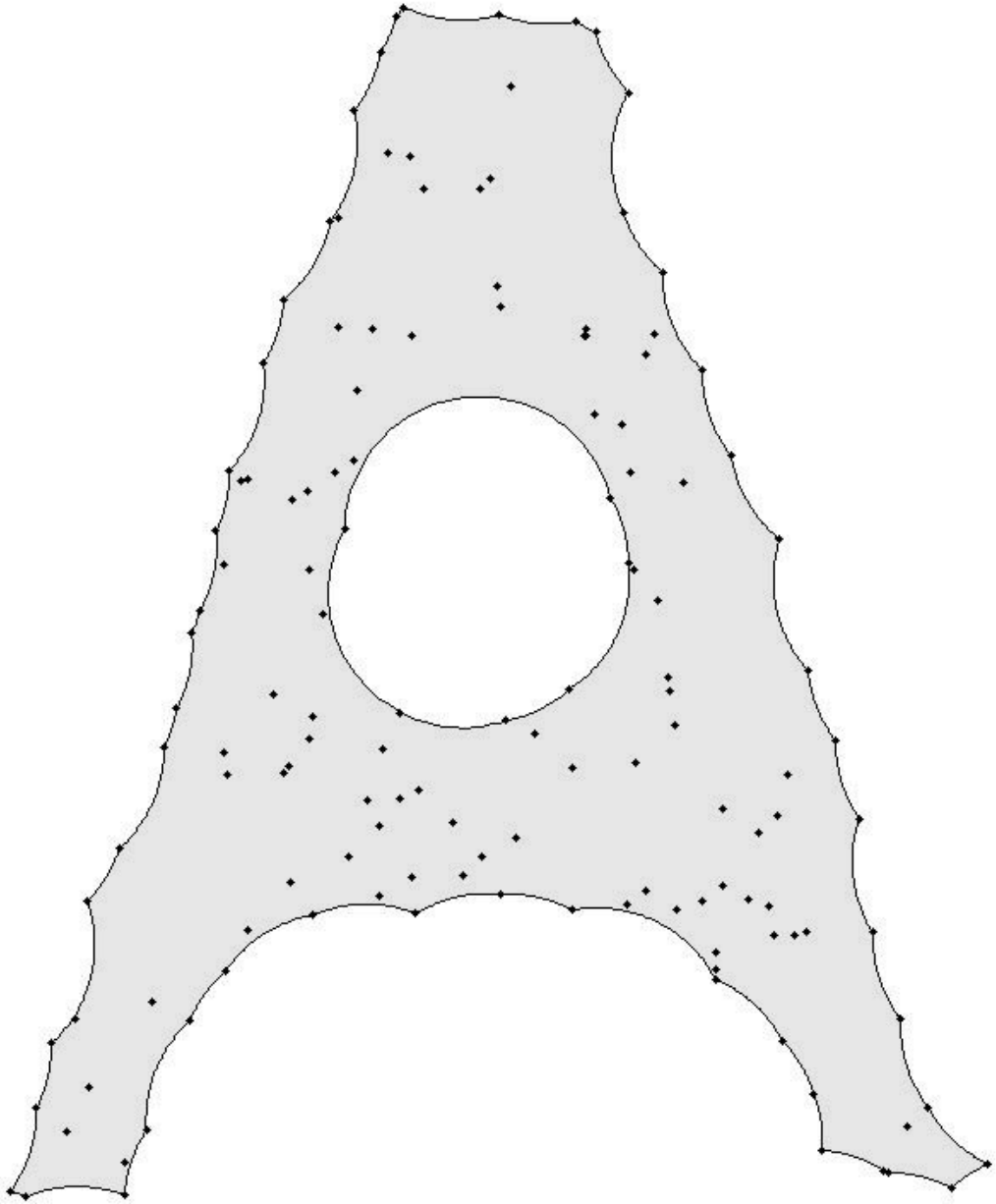
  complement of the union of halfplanes with k pts of S

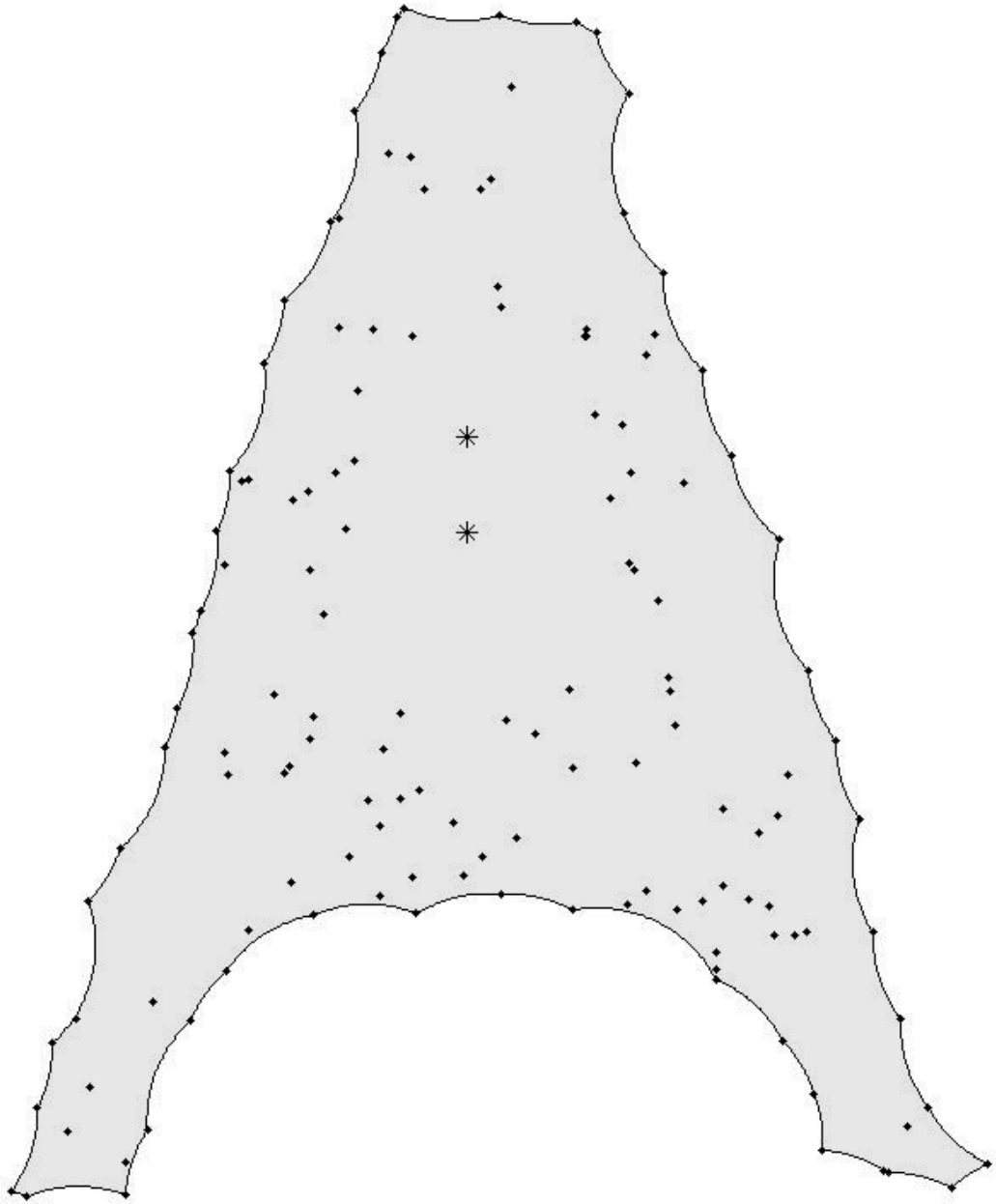
- Field of statistical depth:
  - Tukey depth, statistical depth, travel depth, Mahalanobis depth, majority depth, simplicial depth, CH peeling depth, proximity depth, cone depth



α-hull

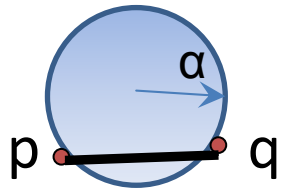
- CH definition 5:
  - complement of the union of empty halfplanes
- Halfplane: disk of radius ∞
  - complement of the union of empty disks of radius ∞
- Generalize: α-hull
  - complement of the union of empty disk of radius α
  - icecream

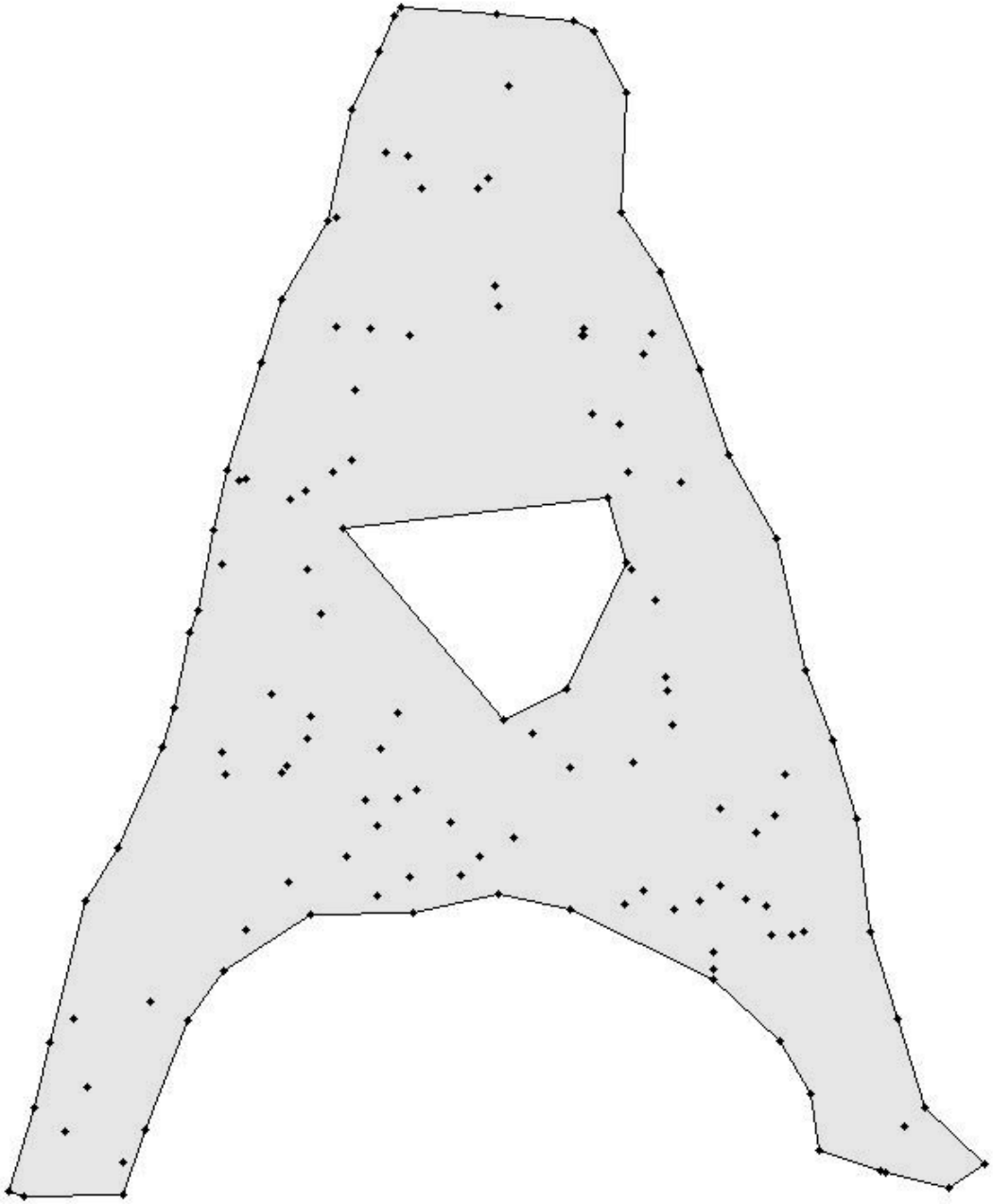


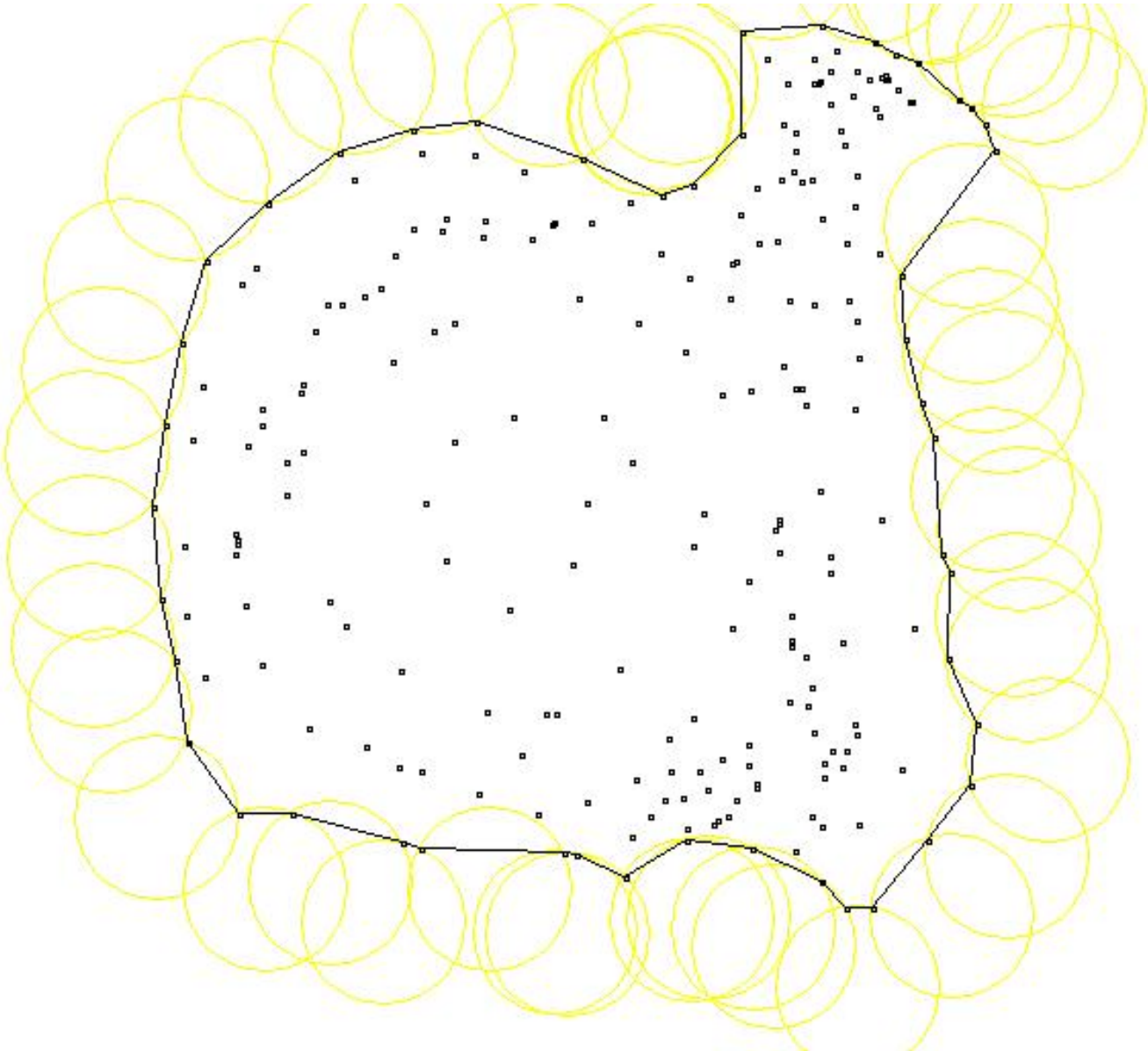


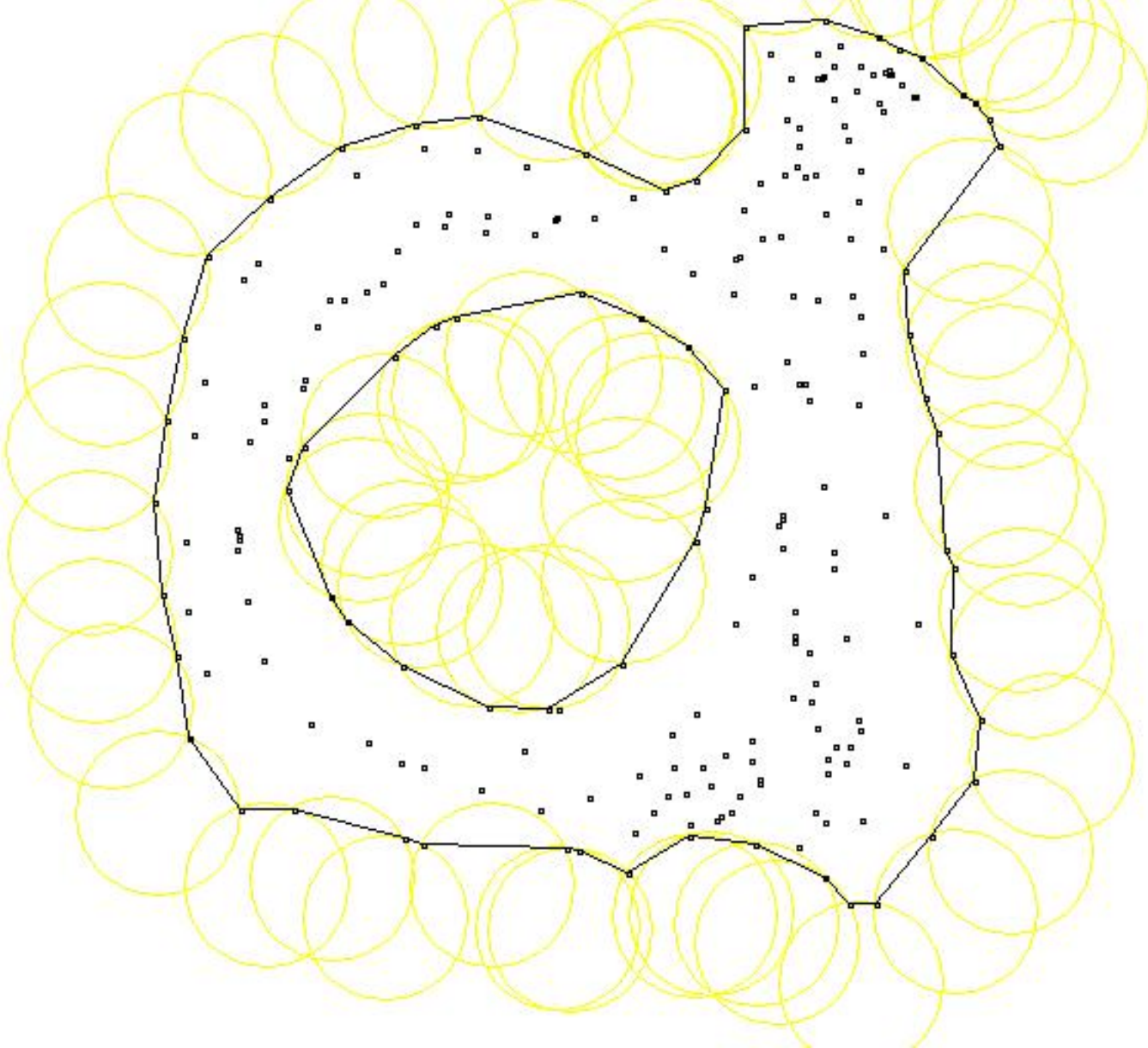
α-shape

- a “straight-line-segment version” of α-hull
  - pq is in α-shape if there is an empty α-disk leaning on p,q

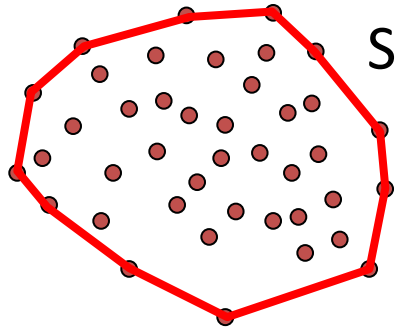








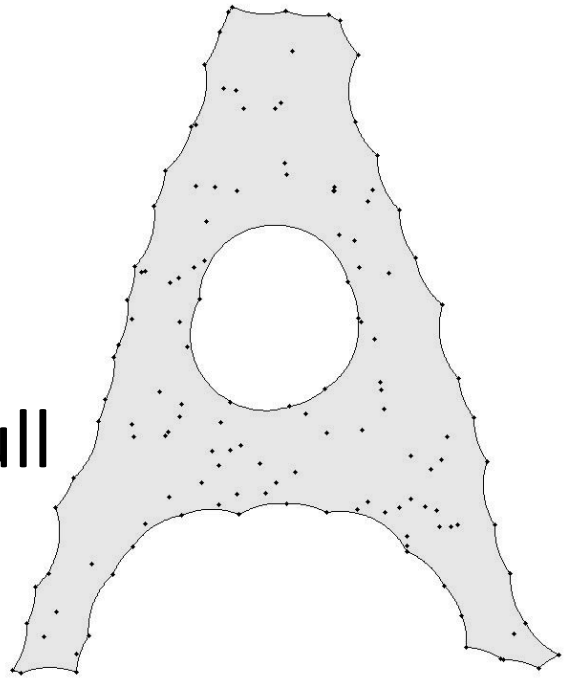
Summary

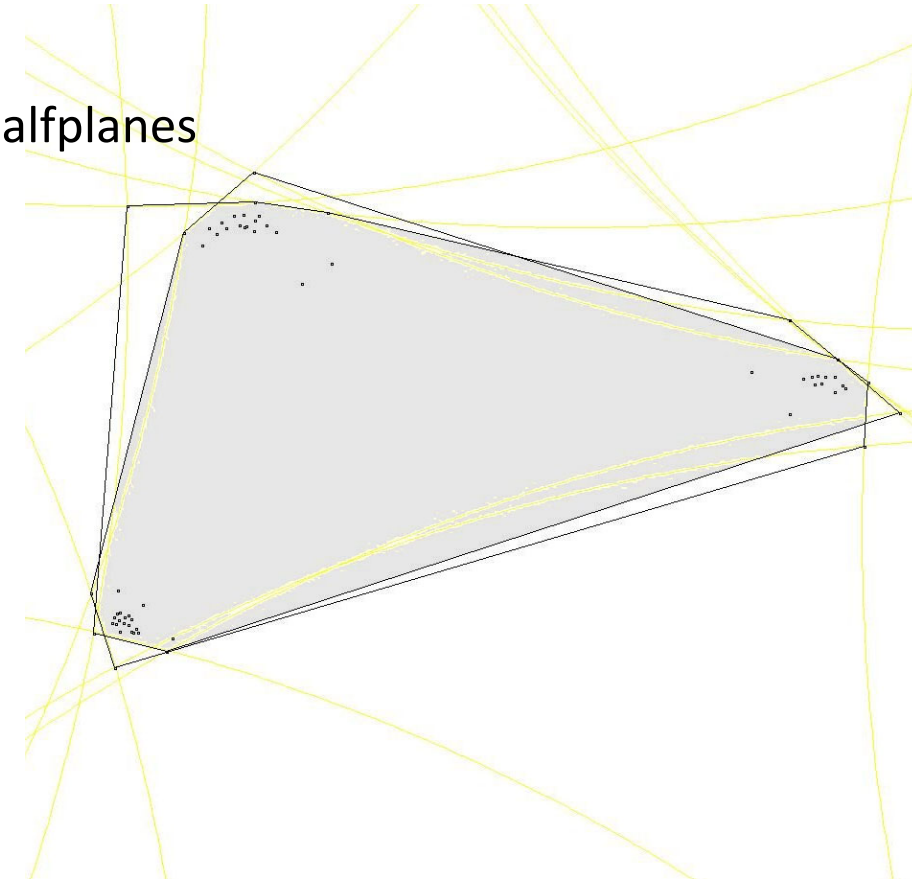


- CH(S):
  1: Smallest convex set containing S
  2: Intersection of all convex sets containing S
  3: Convex combinations of points in S
  4: Points of halfplane depth > 0
  5: Complement of the union of empty halfplanes

- k-hull

- alpha-hull



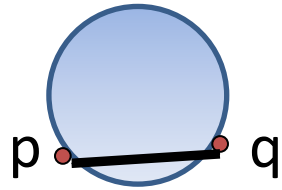


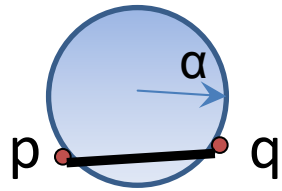
VD, DT

- pq is in DT if

  there is an empty disk leaning on p,q
- pq is in α-shape if

  there is an empty α-disk leaning on p,q





Edge in α-shape is an edge in DT

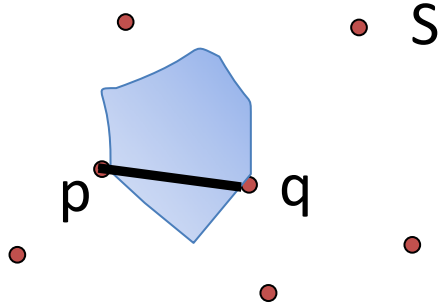
Proximity graphs

- Neighborness graphs, “emptyness” graphs

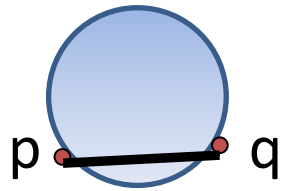
  vertices: S

  pq is an edge if

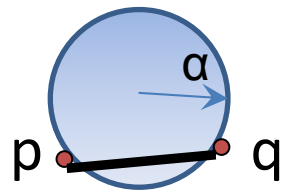
  Neighborhood(p,q) is empty



DT



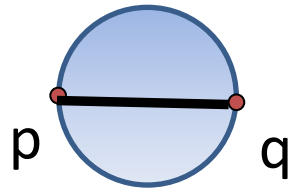
α-shape



Circle-based β-skeleton

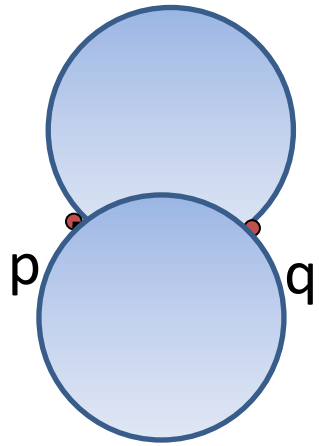
- Gabriel graph
  - empty diametrical disk

  GG = 1-skeleton



- Circle-based β-skeleton, β≥1
  - empty union of disks of
    diam = β•|pq|
- Circle-based β-skeleton, β<1
  - empty intersection of disks of
    diam = |pq|/ β

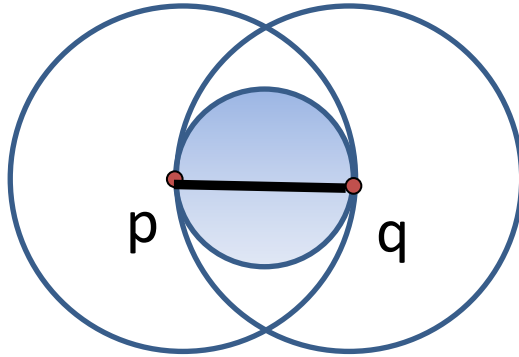
  Complete graph = 0-skeleton



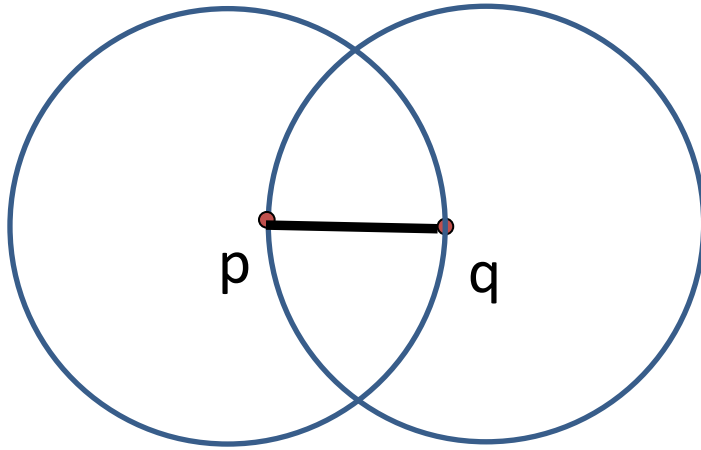
Lune-based β-skeleton

- Relative Neighborhood Graph
  - empty lune
    - RNG is subgrpah GG

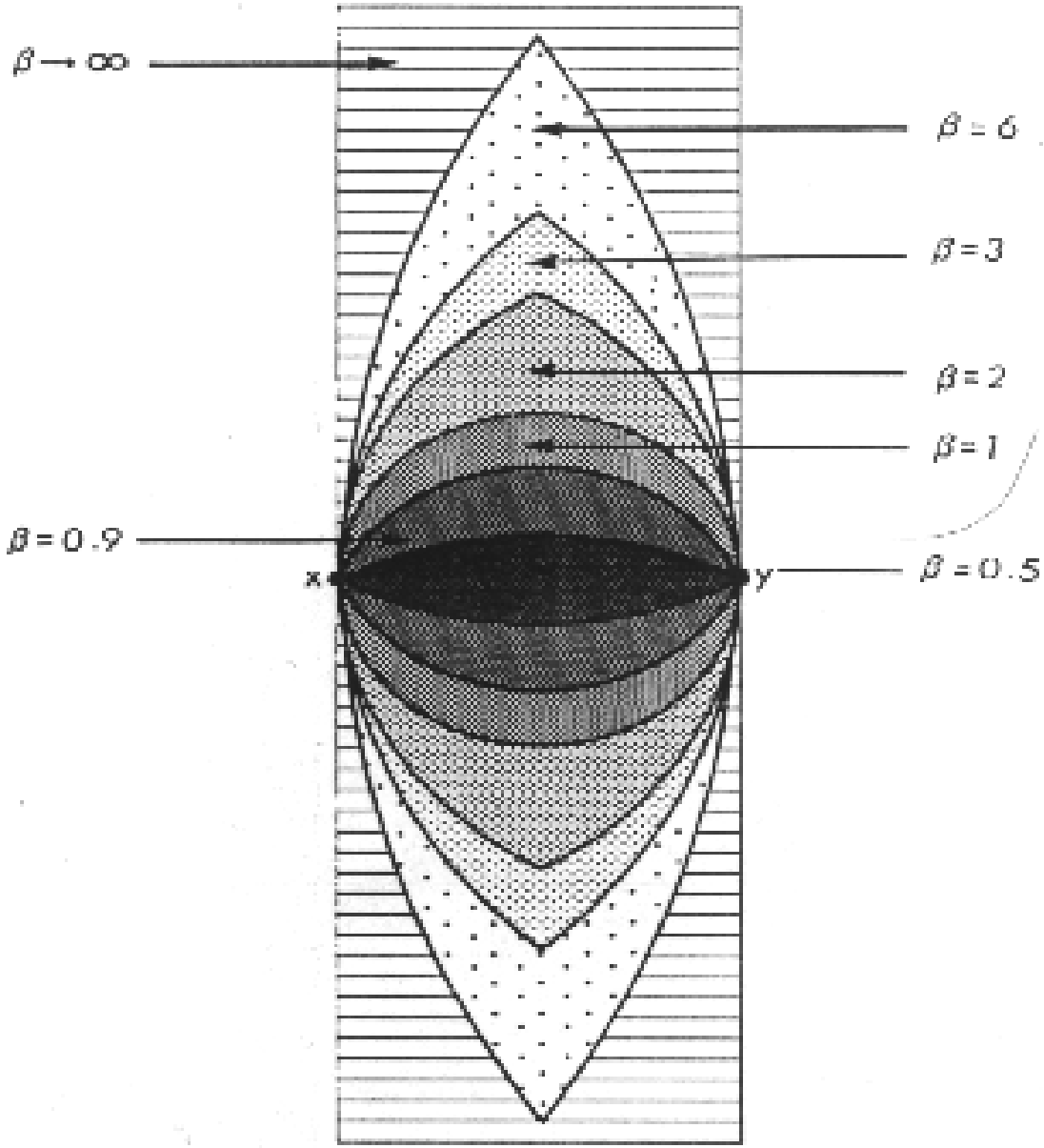
    RNG = 2-skeleton



- Lune-based β-skeleton, β≥1
  - empty lune of disks of diam = β•|pq|



Empty neighborhoods for β-skeleton



Family picture

- *MST (not a proximity graph)*

  is a subgraph of

- lune-based 2-skel = RNG

  is a subgraph of

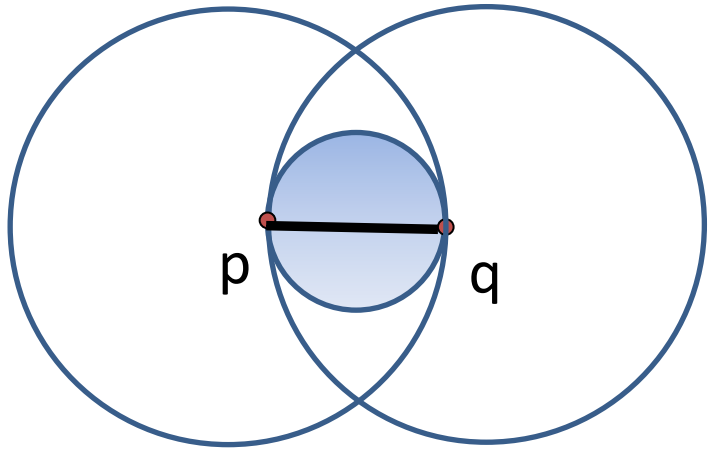
- 1-skeleton = GG

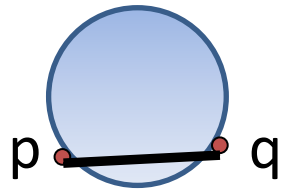
  is a subgraph of

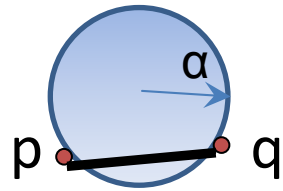
- DT

  is a supergraph of

- α-shape







Rotating calipers

- Width of convex polygon
  - min-width of a slab containing S
- Works also for diameter, minBB

- Project
  - implement rotating calipers
  - build w(l)