Robotic Motion Planning: Roadmap Methods

Robotics Institute

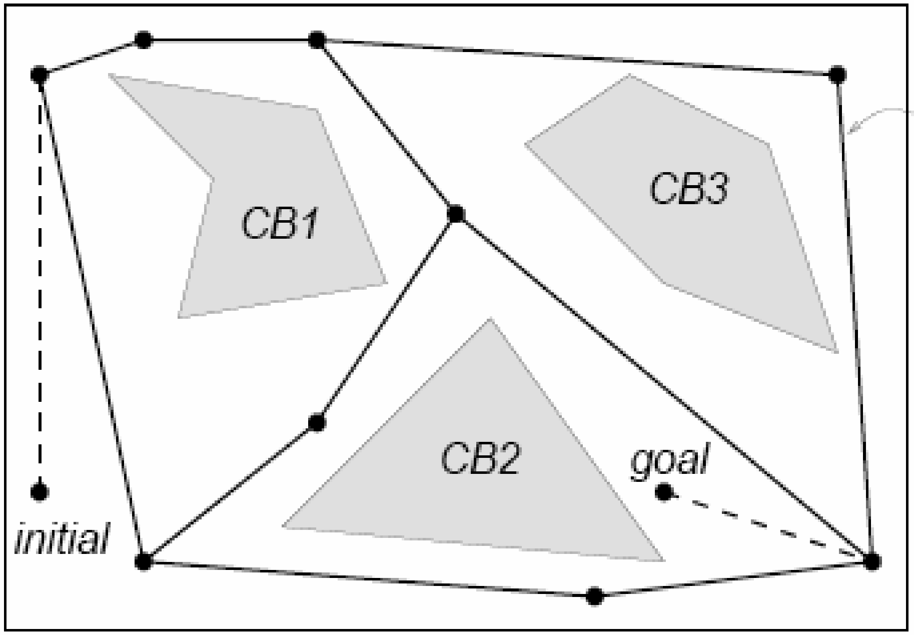
http://voronoi.sbp.ri.cmu.edu/~motion

Howie Choset

http://voronoi.sbp.ri.cmu.edu/~choset

The Basic Idea

- Capture the connectivity of Q_{free} by a graph or network of paths.



RoadMap Definition

- A roadmap, RM, is a union of curves such that for all start and goal points in Q_{free} that can be connected by a path:
  - **Accessibility:** There is a path from $q_{start} \in Q_{free}$ to some $q' \in RM$
  - **Departability:** There is a path from some $q'' \in RM$ to $q_{goal} \in Q_{free}$
  - **Connectivity:** there exists a path in RM between q' and q''
  - **One dimensional**

RoadMap Path Planning

1. Build the roadmap
   a) nodes are points in Q_{free} (or its boundary)
   b) two nodes are connected by an edge if there is a free path between them

2. Connect start end goal points to the road map
   at point q’ and q’’, respectively

3. Connect find a path on the roadmap betwen q’ and q’’

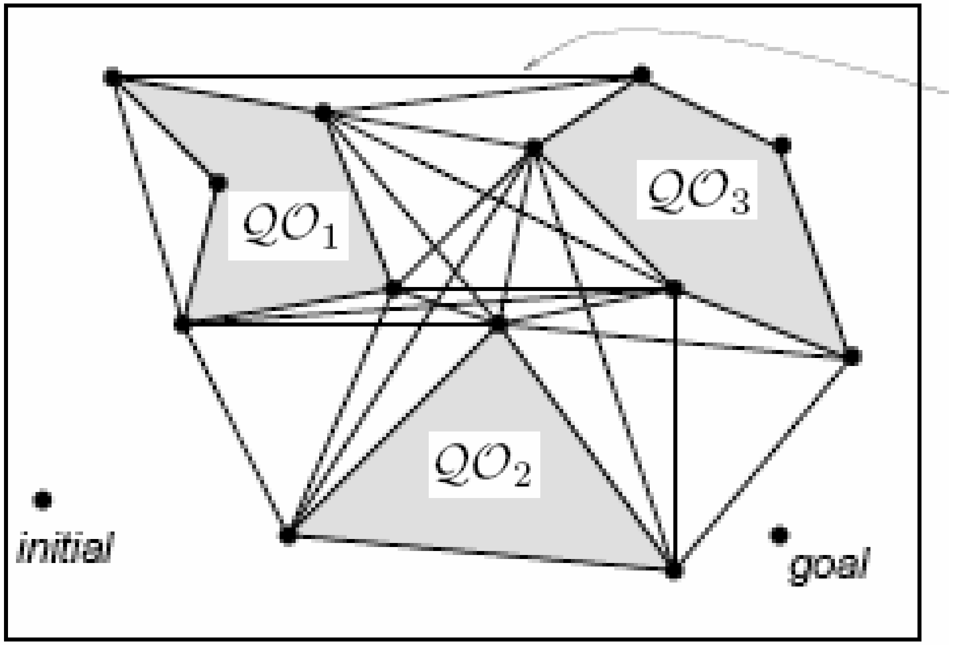
   The result is a path in Q_{free} from start to goal

   Question: what is the hard part here?

Overview

- Deterministic methods
  - Some need to represent $\mathcal{Q}_{\text{free}}$ and some don't.
  - are complete
  - are complexity-limited to simple (e.g. low-dimensional) problems
    - example: Canny's Silhouette method (5.5)
      - applies to general problems
      - is singly exponential in dimension of the problem

Visibility Graph methods

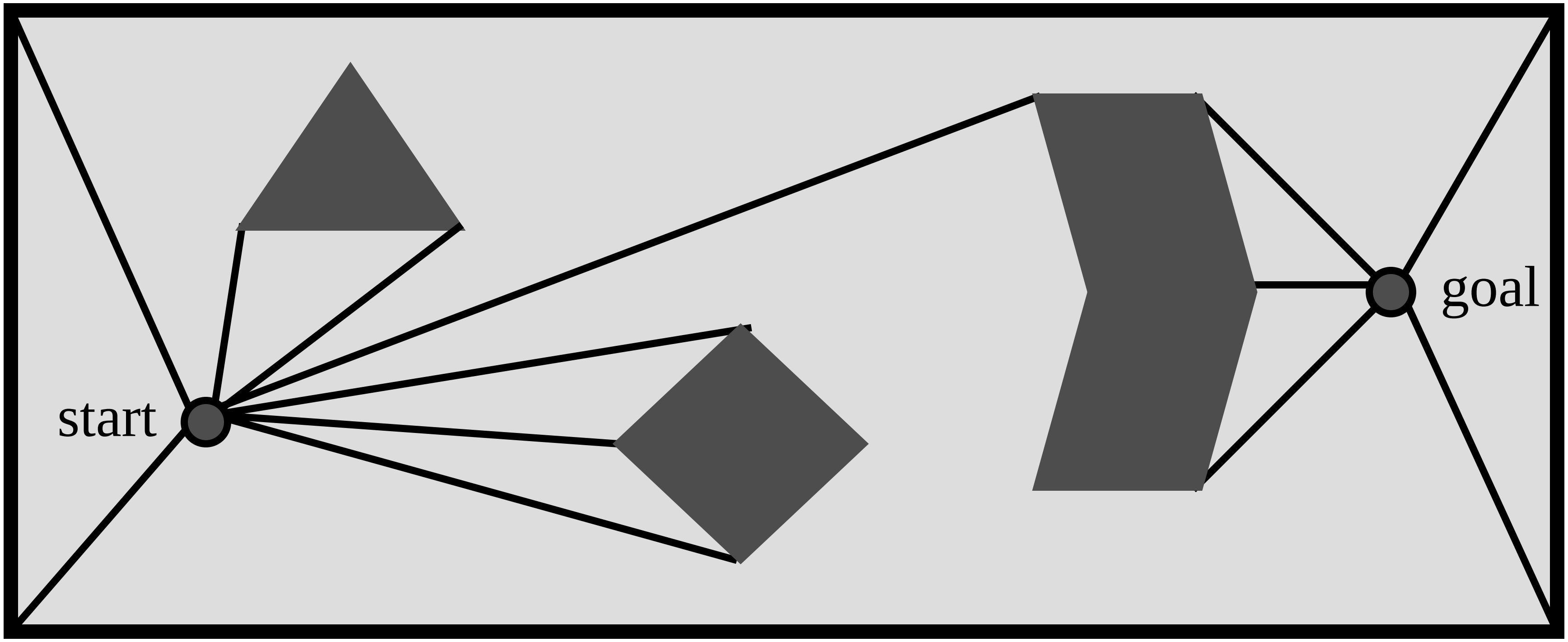


- Defined for polygonal obstacles
- Nodes correspond to vertices of obstacles
- Nodes are connected if
  - they are already connected by an edge on an obstacle
  - the line segment joining them is in free space
- Not only is there a path on this roadmap, but it is the *shortest* path
- If we include the start and goal nodes, they are automatically connected
- Algorithms for constructing them can be efficient
  - O(n^3) brute force

The Visibility Graph in Action (Part 1)

- First, draw lines of sight from the start and goal to all “visible” vertices and corners of the world.

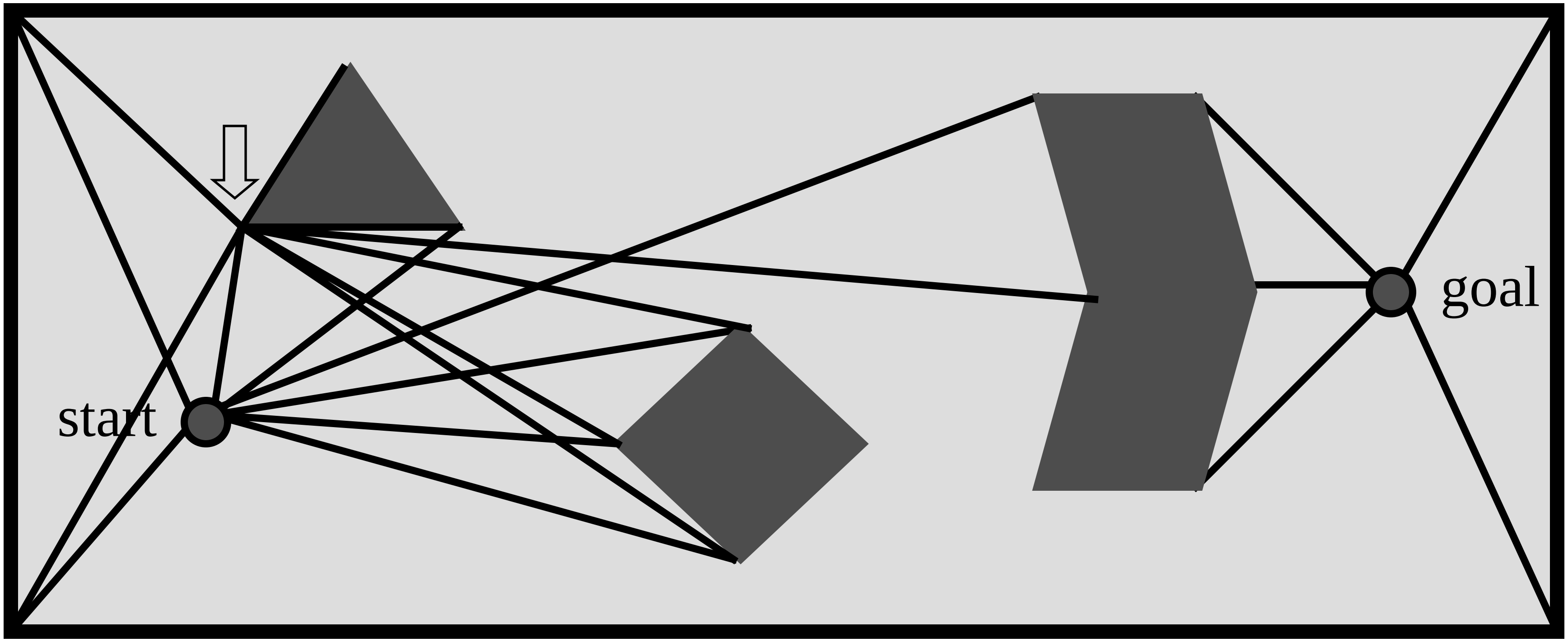
$$e_{ij} \neq \emptyset \iff sv_i + (1-s)v_j \in \mathrm{cl}(\mathcal{Q}_{\mathrm{free}}) \quad \forall s \in (0,1)$$



The Visibility Graph in Action (Part 2)

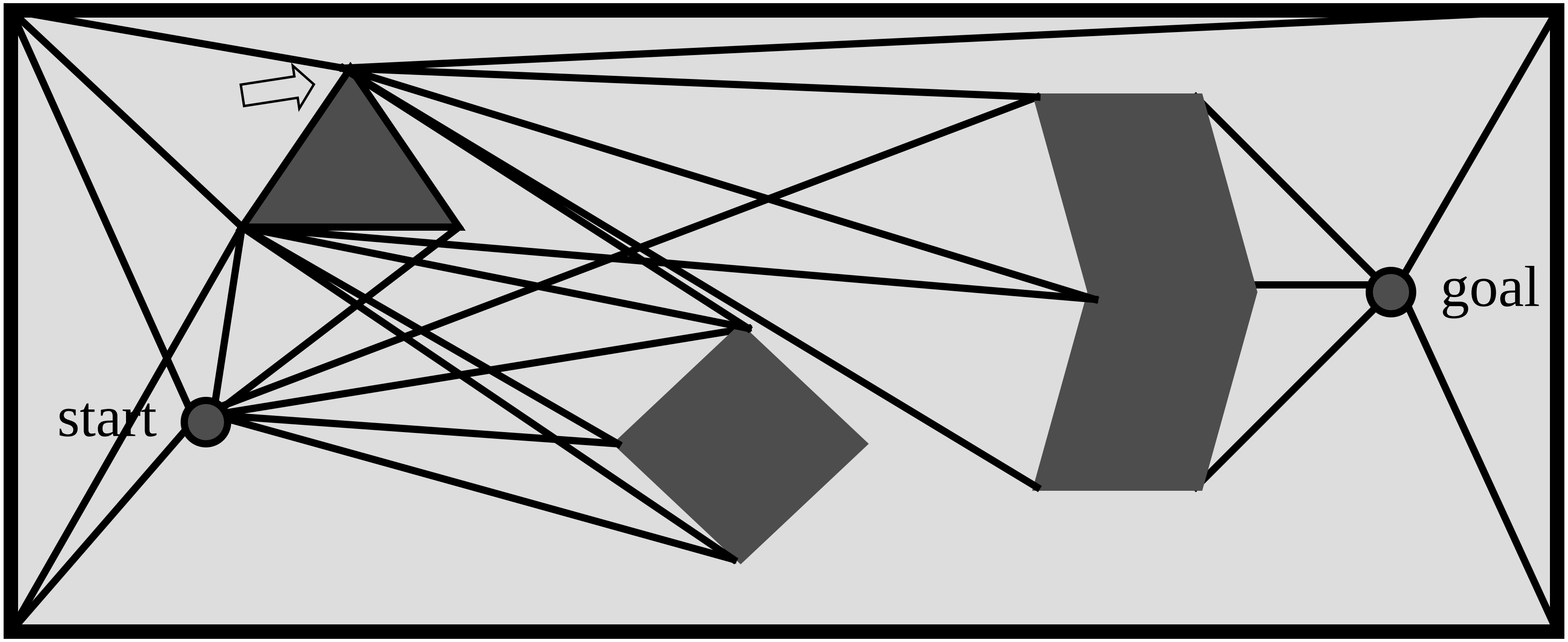
- Second, draw lines of sight from every vertex of every obstacle like before. Remember lines along edges are also lines of sight.

$$e_{ij} \neq \emptyset \iff sv_i + (1-s)v_j \in \mathrm{cl}(\mathcal{Q}_{\mathrm{free}}) \quad \forall s \in (0,1)$$



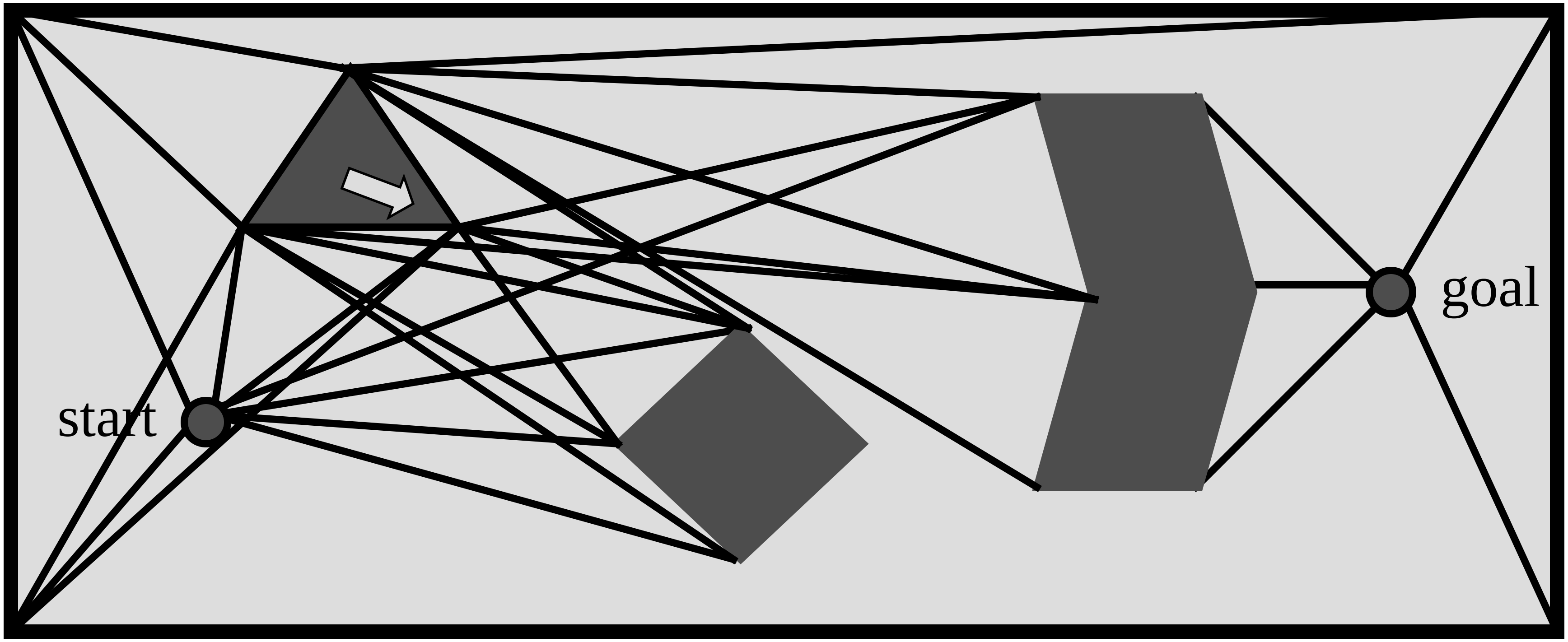
The Visibility Graph in Action (Part 3)

- Second, draw lines of sight from every vertex of every obstacle like before. Remember lines along edges are also lines of sight.



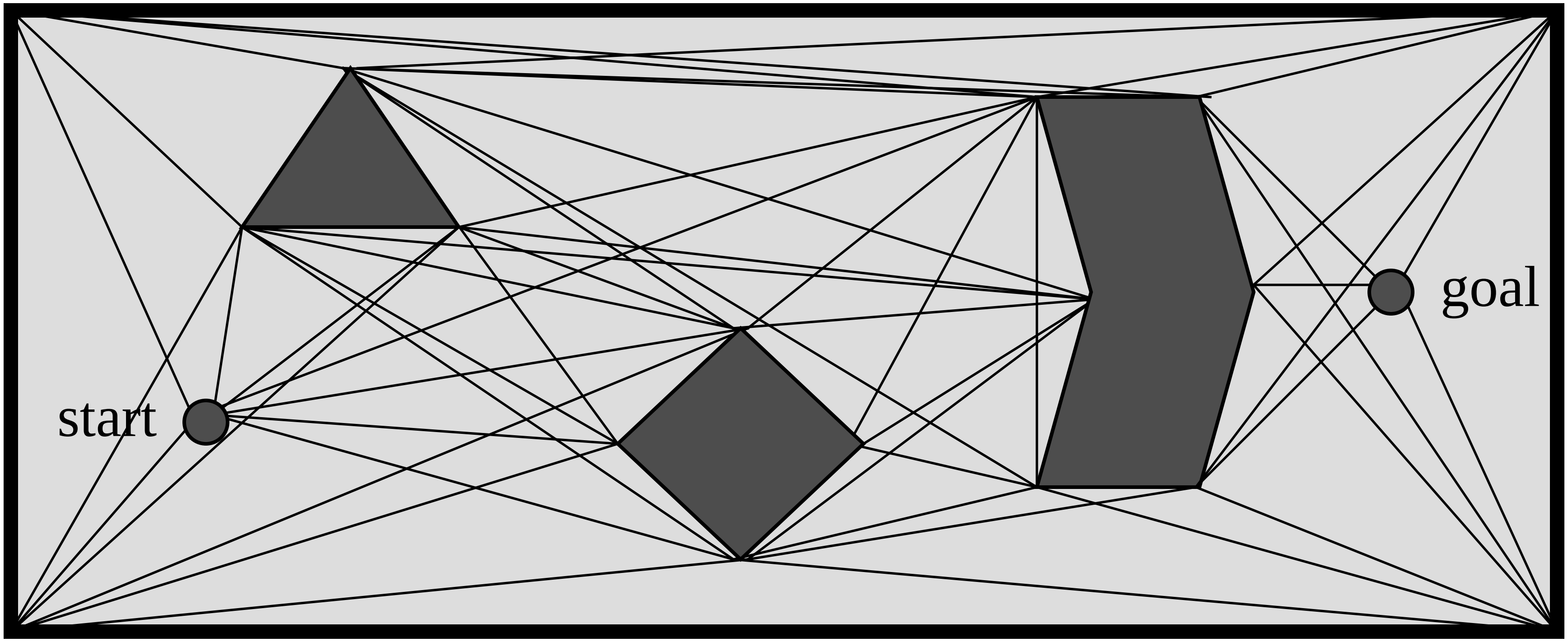
The Visibility Graph in Action (Part 4)

- Second, draw lines of sight from every vertex of every obstacle like before. Remember lines along edges are also lines of sight.

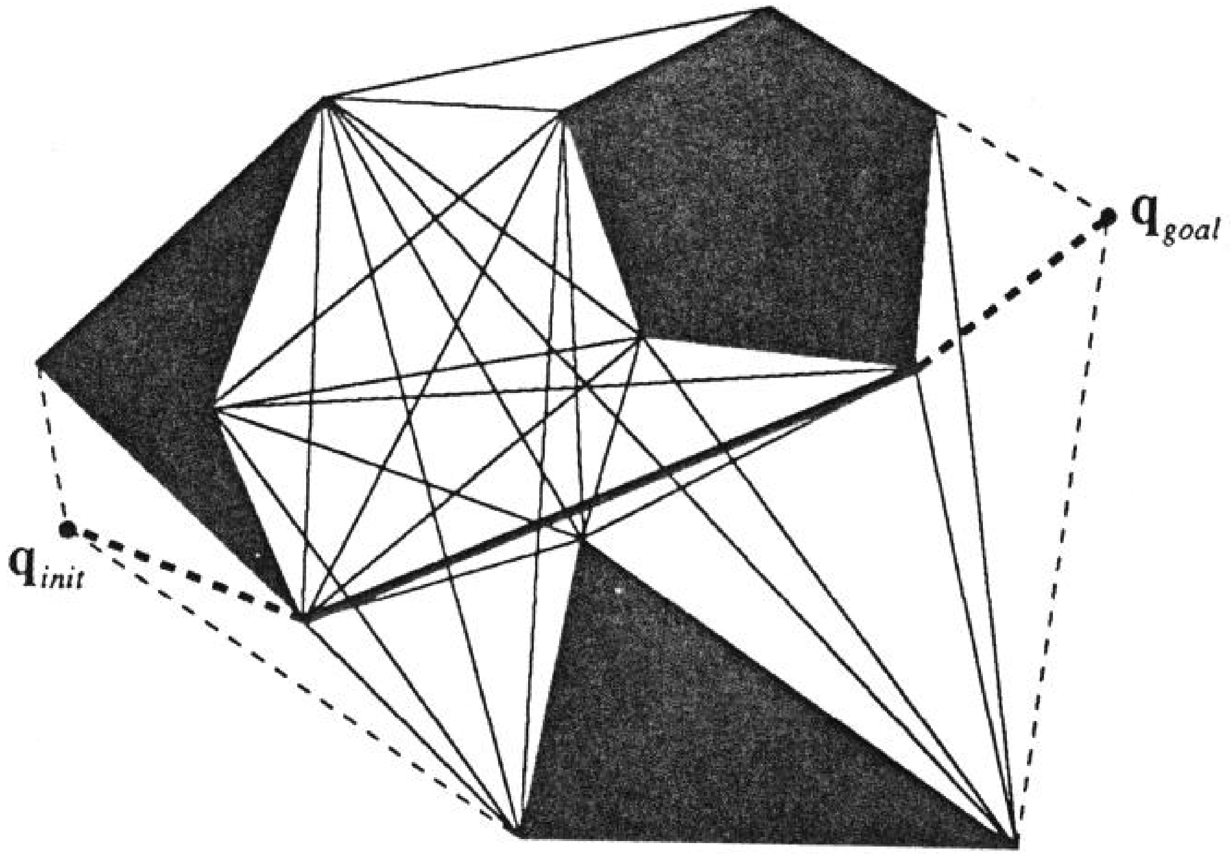


The Visibility Graph (Done)

- Repeat until you're done.

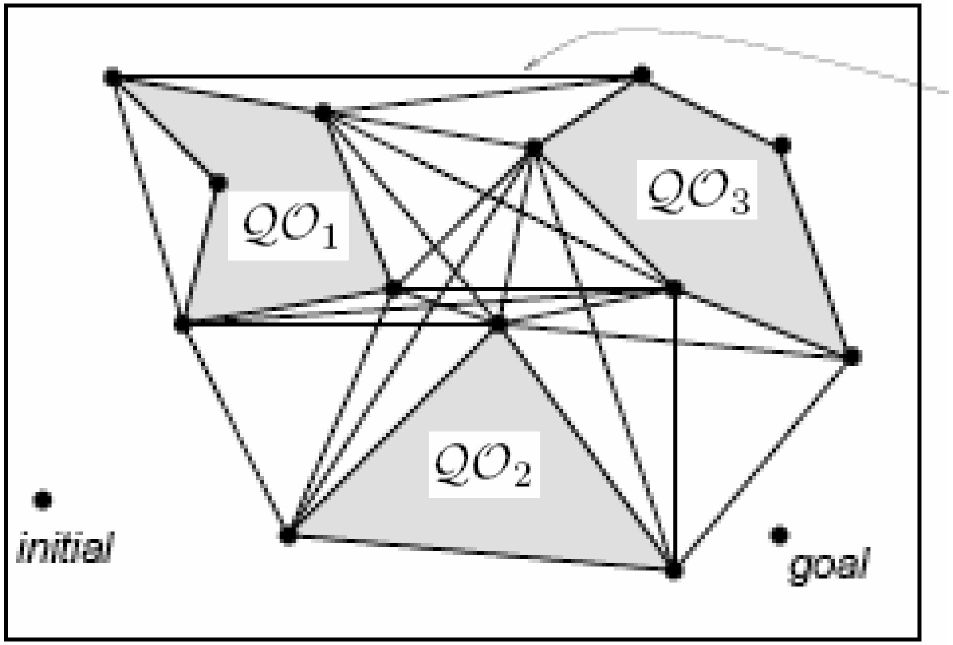


Visibility Graphs



16-735, Howie Choset, with significant copying from G.D. Hager who loosely based his notes on notes by Nancy Amato

Visibility Graph methods



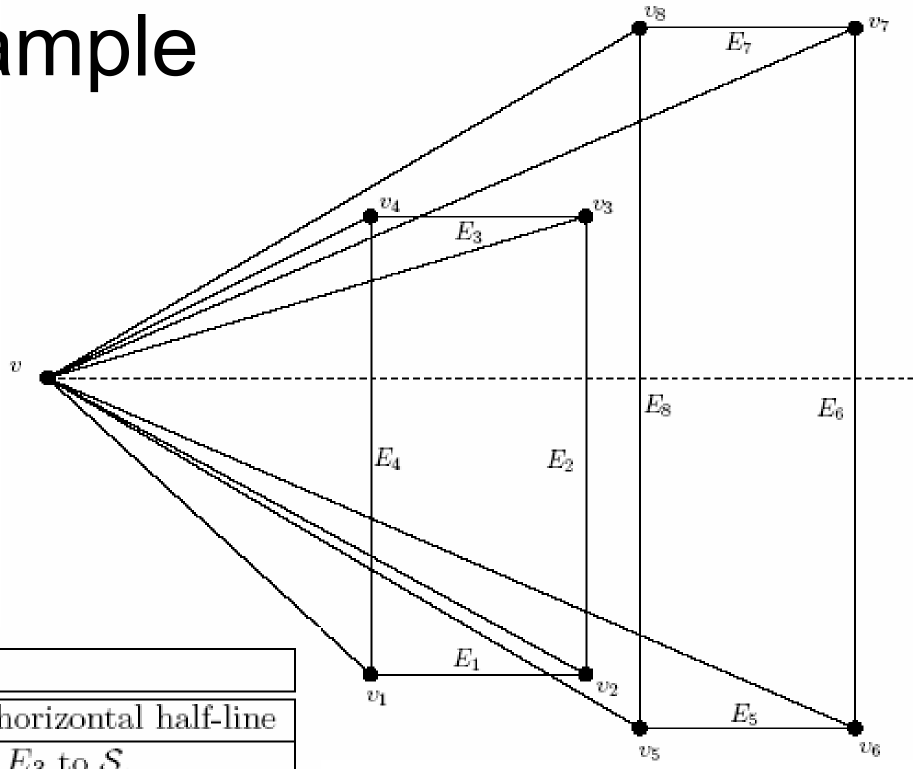
- Defined for polygonal obstacles
- Nodes correspond to vertices of obstacles
- Nodes are connected if
  - they are already connected by an edge on an obstacle
  - the line segment joining them is in free space
- Not only is there a path on this roadmap, but it is the *shortest* path
- If we include the start and goal nodes, they are automatically connected
- Algorithms for constructing them can be efficient
  - O(n^3) brute force

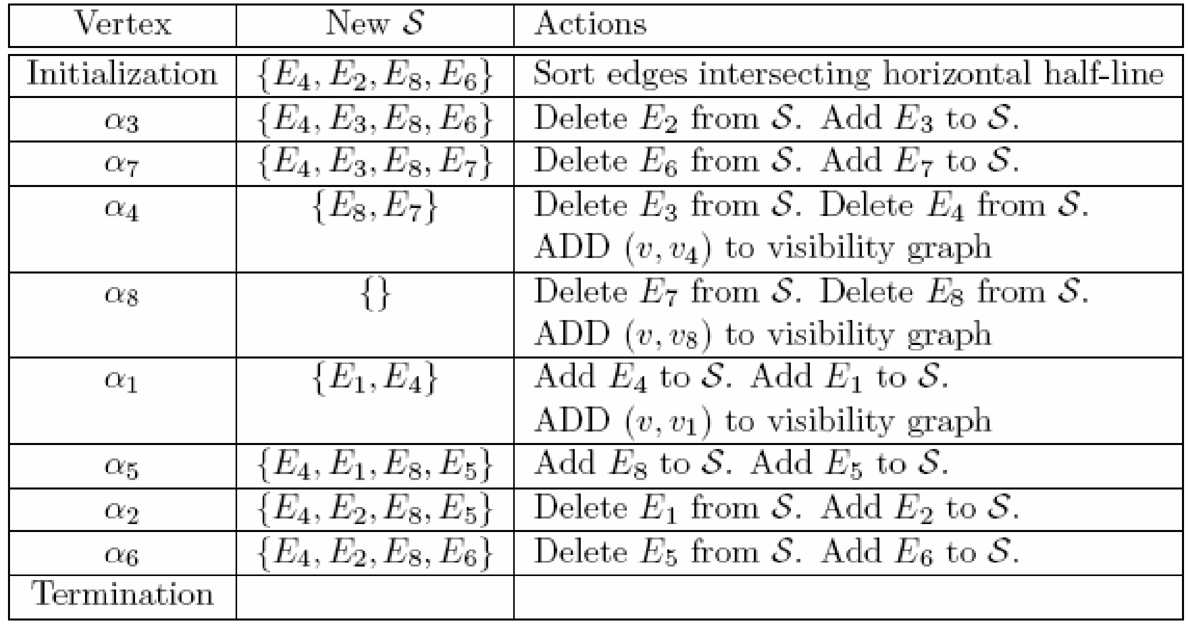
The Sweepline Algorithm

- Goal: calculate the set of vertices v_i that are visible from v
- visibility: a segment v-v_i is visible if
  - it is not within the object
  - the closest line intersecting v-v_i is beyond v_i
- **Algorithm:**
  Initially:
  - calculate the angle α_i of segment v-v_i and sort vertices by this creating list E
  - create a list of edges that intersect the horizontal from v sorted by intersection distance
- For each α_i
  - if v_i is visible to v then add v-v_i to graph
  - if v_i is the "beginning" of an edge E, insert E in S
  - if v_i is the "end" of and edge E, remove E from S

Analysis: For a vertex, n log n to create initial list, log n for each α_i
Overall: n log (n) (or n^2 log (n) for all n vertices

Example



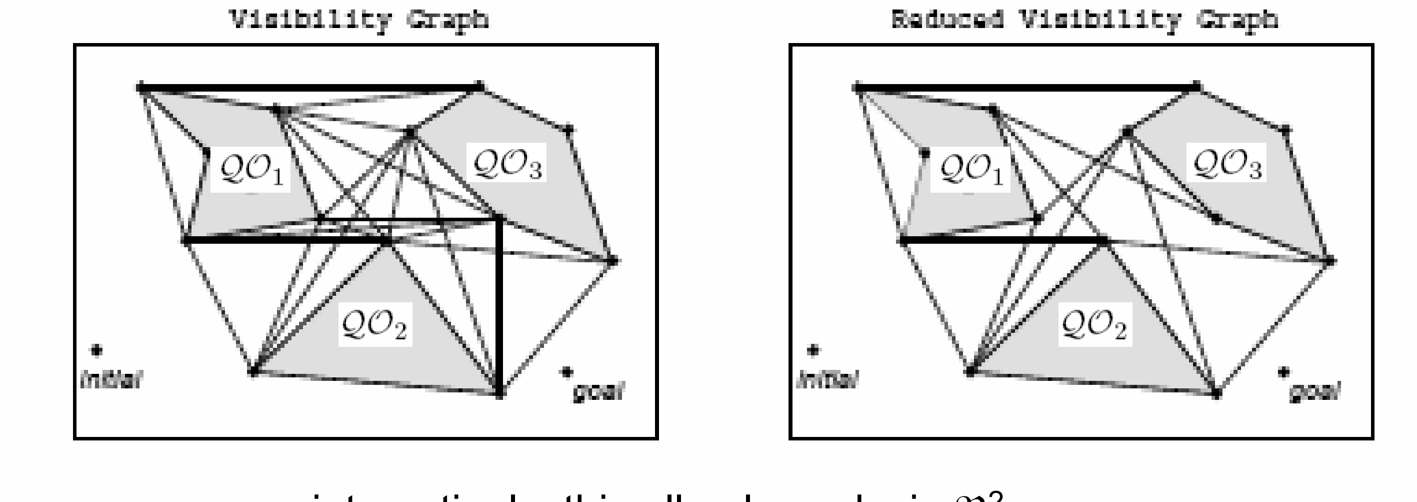


Vertex	New $\mathcal{S}$	Actions
Initialization	$\{E_4, E_2, E_8, E_6\}$	Sort edges intersecting horizontal half-line
α_3	$\{E_4, E_3, E_8, E_6\}$	Delete E_2 from $\mathcal{S}$. Add E_3 to $\mathcal{S}$.
α_7	$\{E_4, E_3, E_8, E_7\}$	Delete E_6 from $\mathcal{S}$. Add E_7 to $\mathcal{S}$.
α_4	$\{E_8, E_7\}$	Delete E_3 from $\mathcal{S}$. Delete E_4 from $\mathcal{S}$. ADD (v, v_4) to visibility graph
α_8	$\{\}$	Delete E_7 from $\mathcal{S}$. Delete E_8 from $\mathcal{S}$. ADD (v, v_8) to visibility graph
α_1	$\{E_1, E_4\}$	Add E_4 to $\mathcal{S}$. Add E_1 to $\mathcal{S}$. ADD (v, v_1) to visibility graph
α_5	$\{E_4, E_1, E_8, E_5\}$	Add E_8 to $\mathcal{S}$. Add E_5 to $\mathcal{S}$.
α_2	$\{E_4, E_2, E_8, E_5\}$	Delete E_1 from $\mathcal{S}$. Add E_2 to $\mathcal{S}$.
α_6	$\{E_4, E_2, E_8, E_6\}$	Delete E_5 from $\mathcal{S}$. Add E_6 to $\mathcal{S}$.
Termination		

Reduced Visibility Graphs

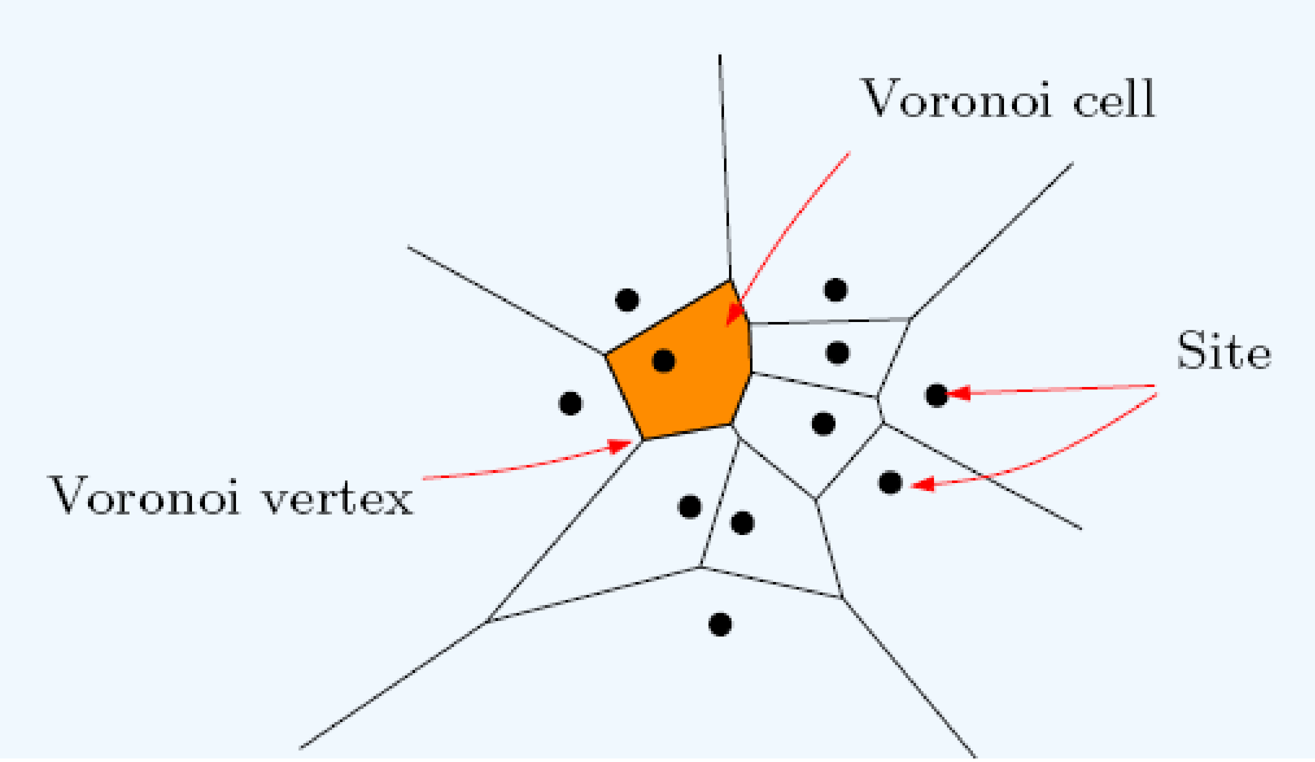
- The current graph as too many lines
  - lines to concave vertices
  - lines that “head into” the object

- A reduced visibility graph consists of
  - nodes that are convex
  - edges that are “tangent” (i.e. do not head into the object at either endpoint)



interestingly, this all only works in $\Re^2$

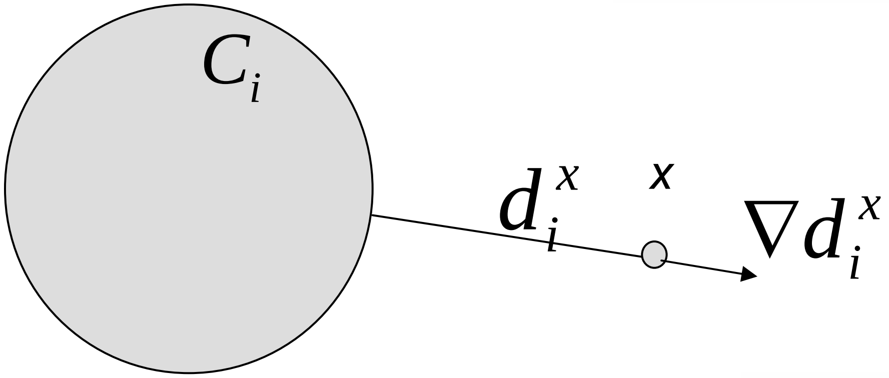
Voronoi Diagrams



Beyond Points: Basic Definitions

Single-object distance function

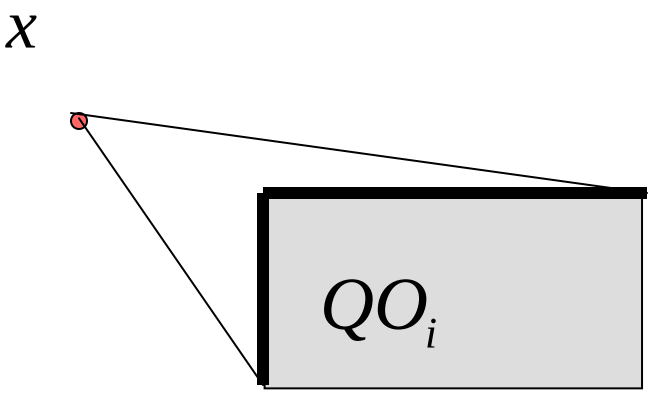
$$d_i^x(q) = \min_{c \in \mathcal{QO}_i} d(q, c)$$



$$\nabla d_i^x(q) = \frac{q - c}{d(q, c)}$$

X for “X-ray”

Points within line of sight



$$\tilde{C}_i(x) = \{c \in QO_i : \forall t \in [0,1], x(1-t) + ct \in Q_{\text{free}}\}$$

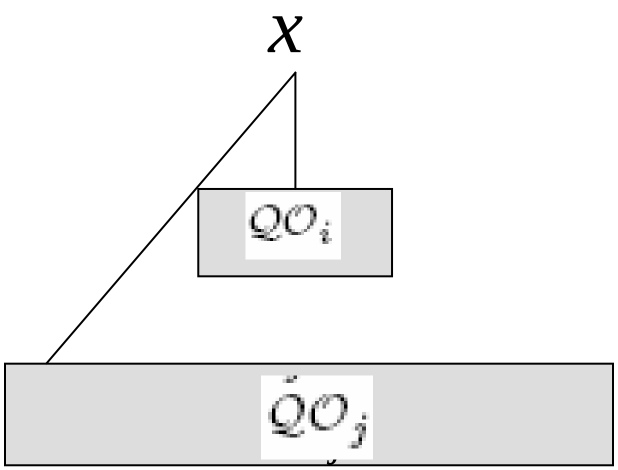
Visible Distance Functions

- *Single-object*

$$d_i(x) = \begin{cases} \text{distance to } \mathcal{QO}_i \text{ if } c_i \in \tilde{C}_i(x) \\ \infty \quad\quad\quad\quad \text{otherwise} \end{cases}$$

- *Multi-object*

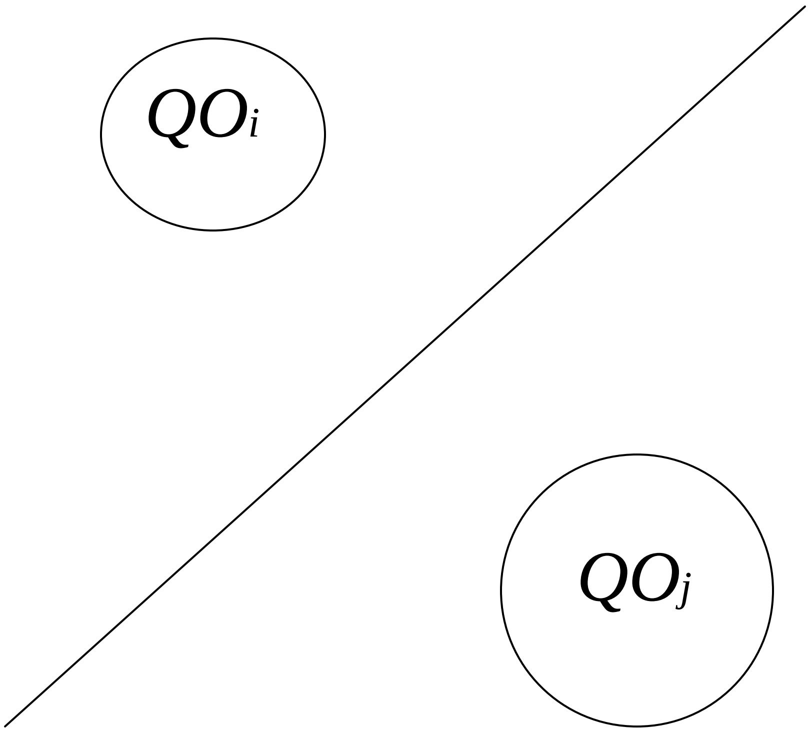
$$D(x) = \min_i d_i(x)$$



Two-Equidistant

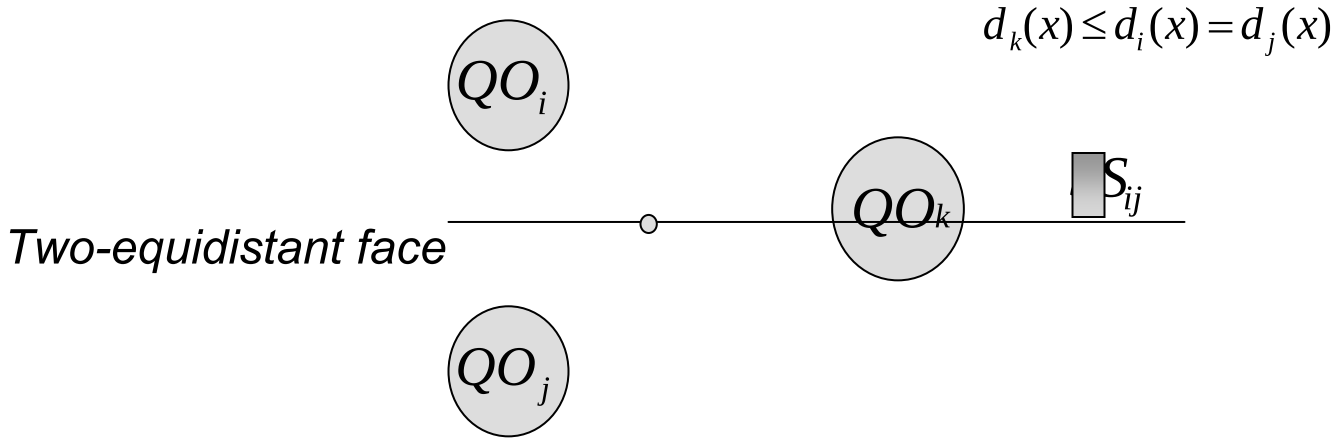
- *Two-equidistant surface*

$$S_{ij} = \{x \in Q_{\text{free}} : d_i(x) - d_j(x) = 0\}$$



More Rigorous Definition

Going through obstacles

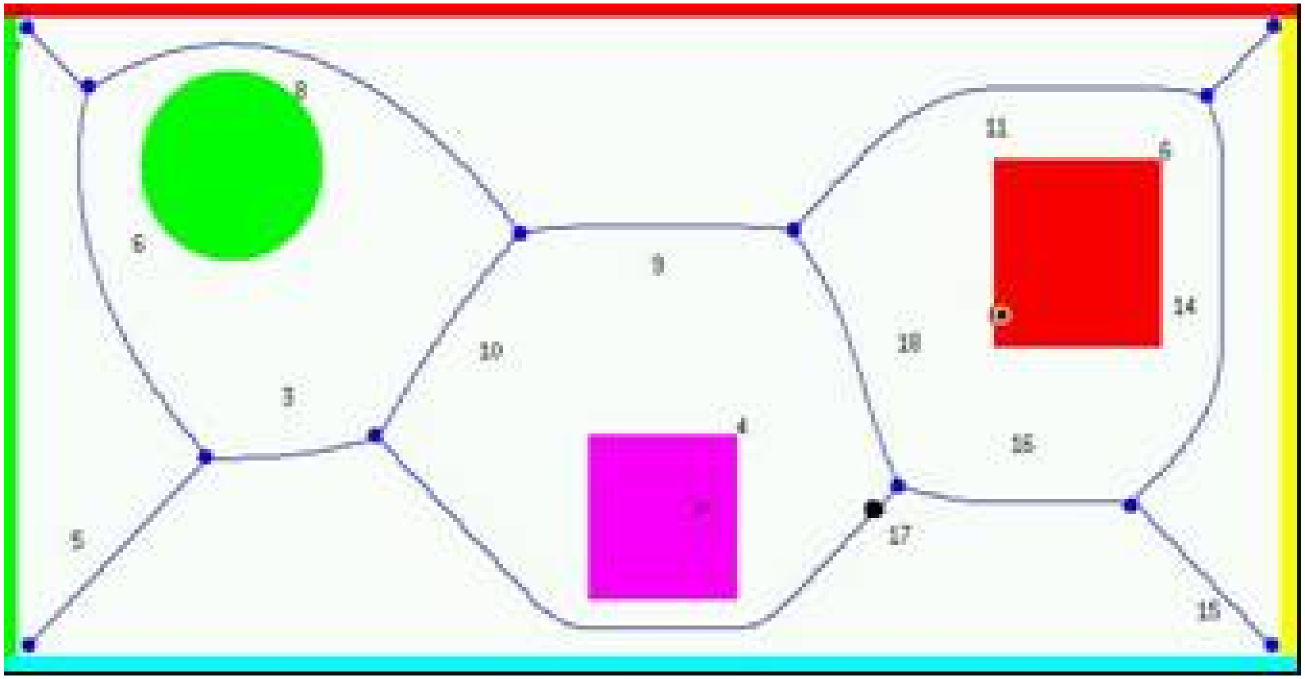


$d_k(x) \leq d_i(x) = d_j(x)$

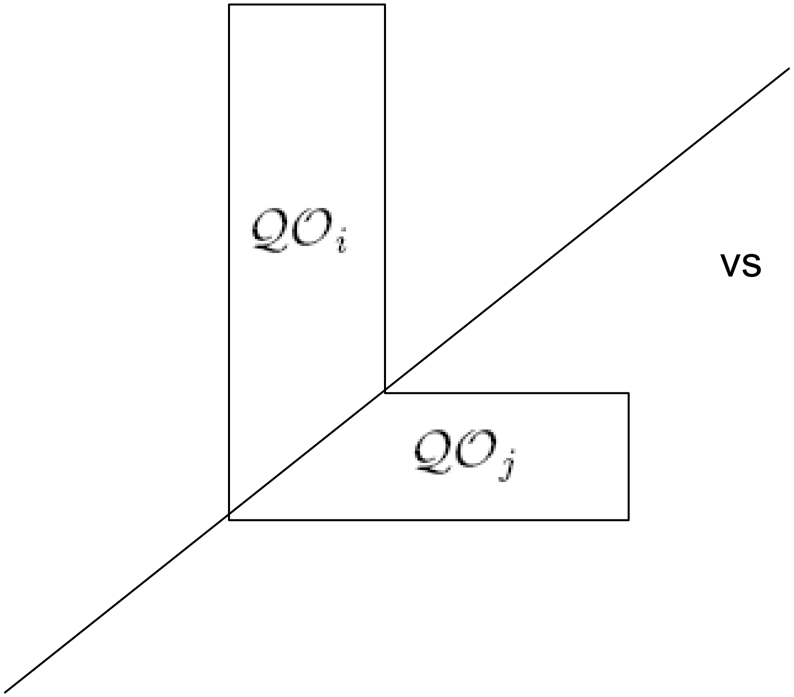
$$F_{ij} = \{x \in S_{ij} : d_i(x) = d_j(x) \leq d_h(x), \forall h \neq i, j\}$$

General Voronoi Diagram

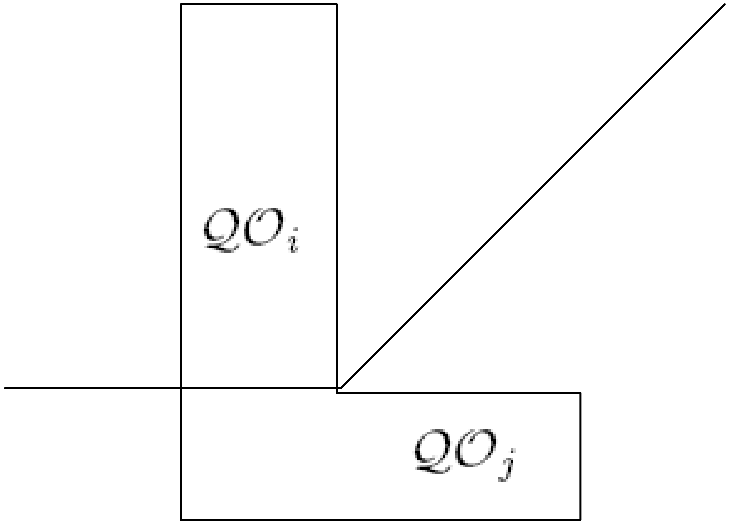
$$\mathrm{GVD} = \bigcup_{i=1}^{n-1} \bigcup_{j=i+1}^{n} F_{ij}$$



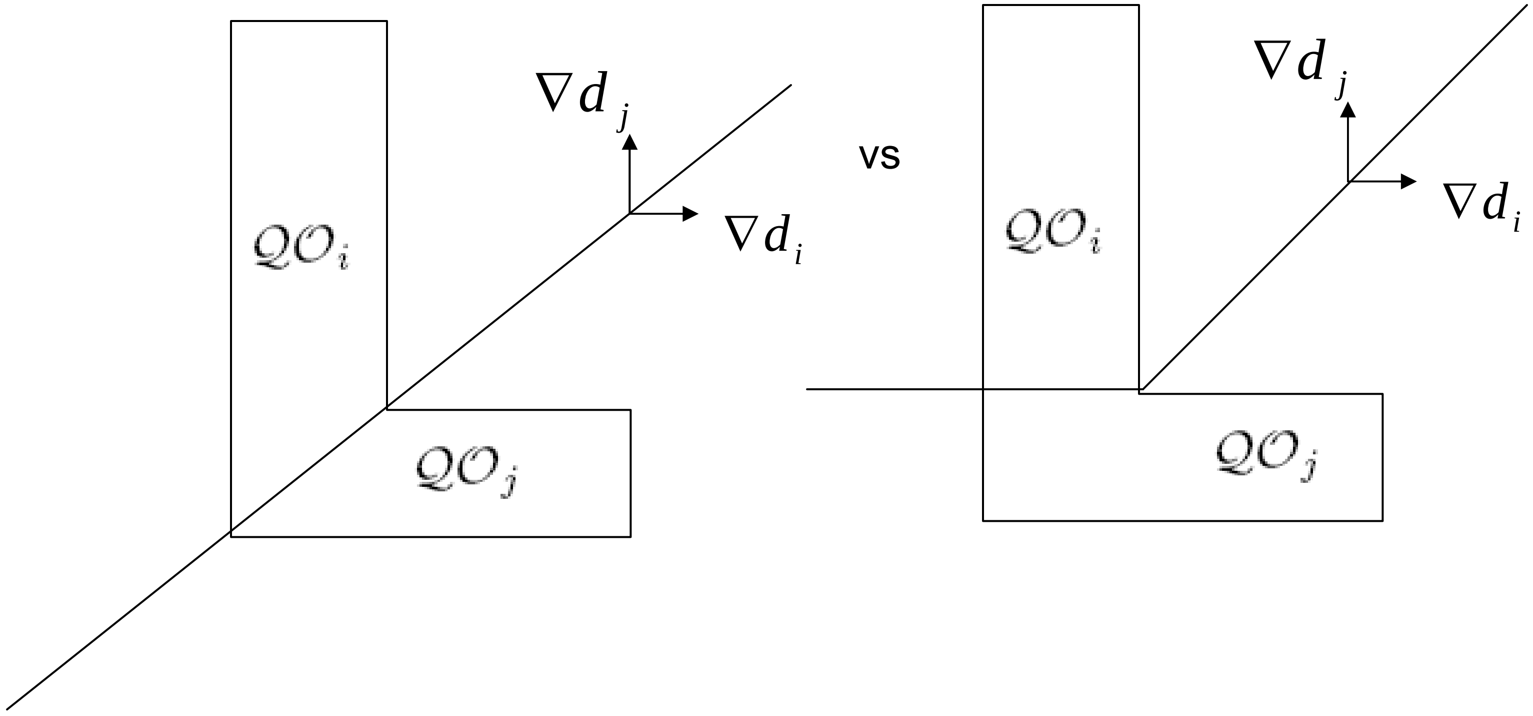
What about concave obstacles?



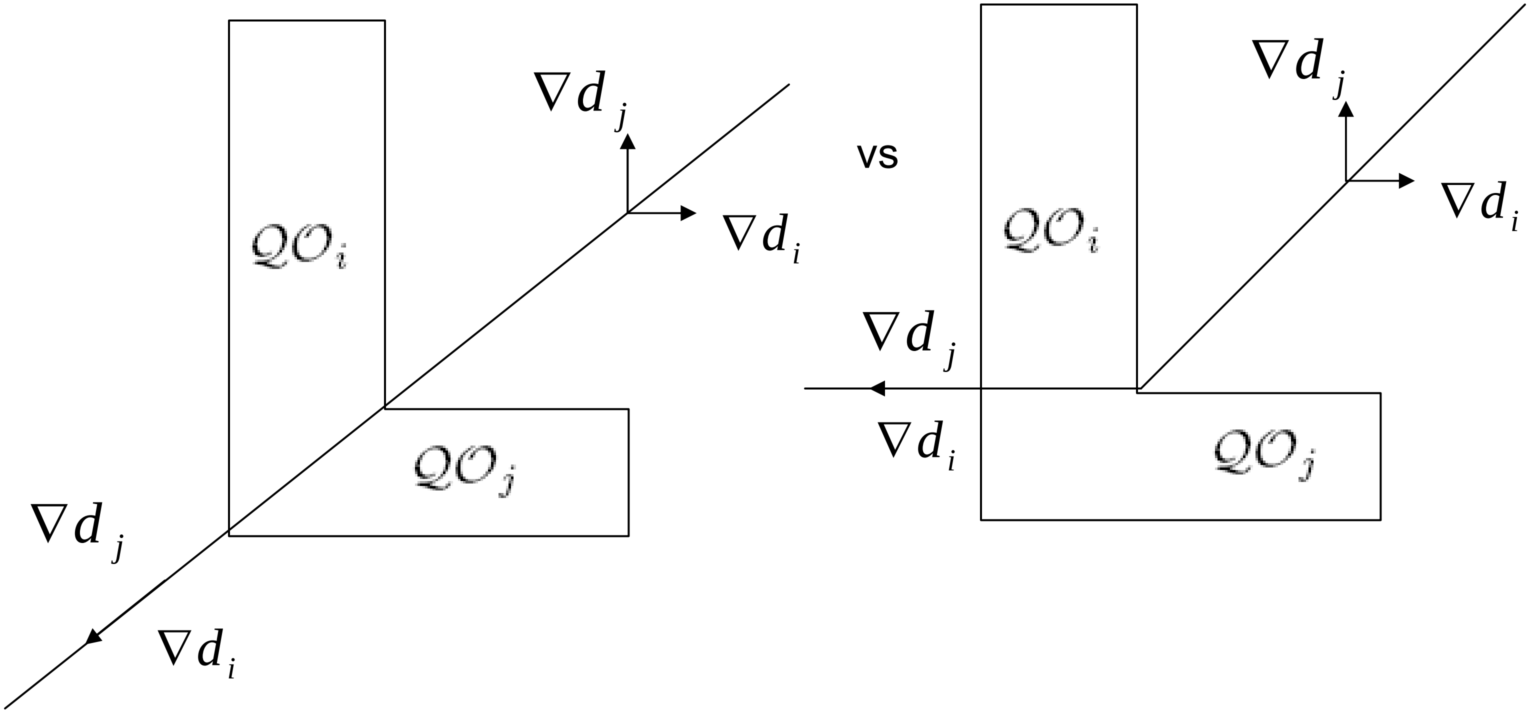
vs



What about concave obstacles?



What about concave obstacles?



16-735, Howie Choset, with significant copying from G.D. Hager who
loosely based his notes on notes by Nancy Amato

Two-Equidistant

- *Two-equidistant surface*

$$S_{ij} = \{x \in Q_{\text{free}} : d_i(x) - d_j(x) = 0\}$$

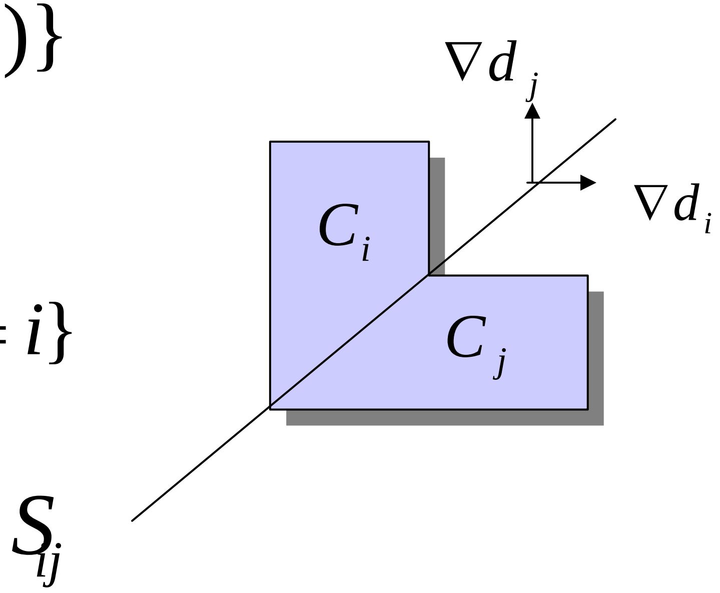
Two-equidistant surjective surface

$$SS_{ij} = \{x \in S_{ij} : \nabla d_i(x) \neq \nabla d_j(x)\}$$

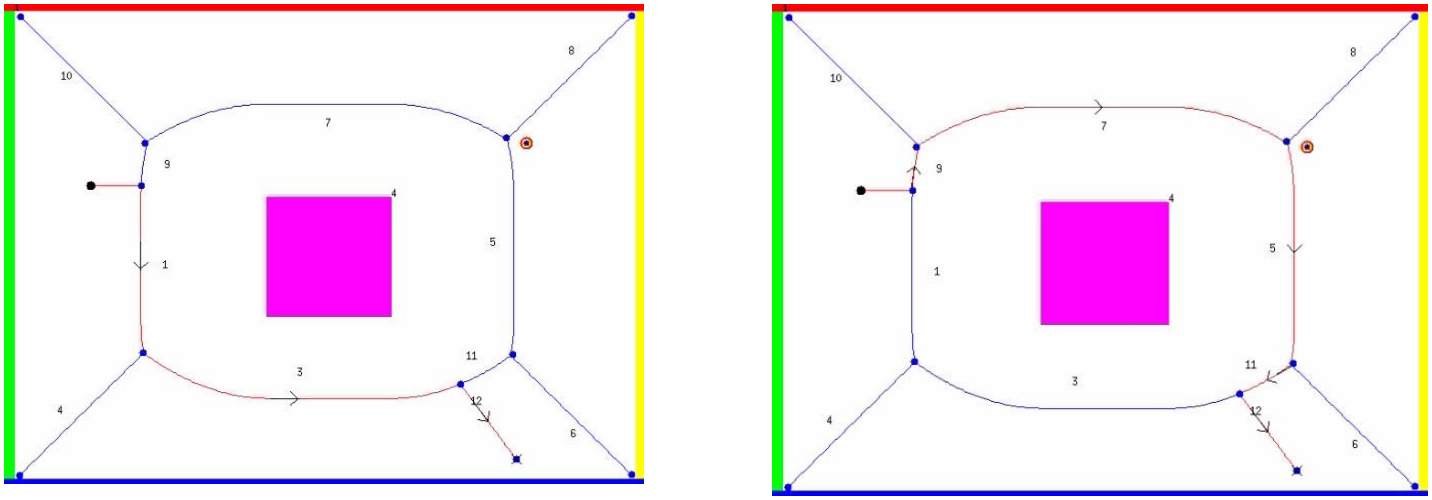
Two-equidistant Face

$$F_{ij} = \{x \in SS_{ij} : d_i(x) \leq d_h(x), \forall h \neq i\}$$

$$\text{GVD} = \bigcup_{i=1}^{n-1} \bigcup_{j=i+1}^{n} F_{ij}$$



Curve Optimization Approach: Homotopy Classes



$$[c] = \{\bar{c} \in C^0 \mid \bar{c} \sim c\}$$

Pre-Image Theorem

$$f : R^m \to R^n$$

$$f^{-1}(c) = \{x \in R^m : f(x) = c\}$$

e.g. $f(x, y) = x^2 + y^2 \qquad f : R^2 \to R$

$f^{-1}(9)$ is a circle with radius 3

if $\forall x \in f^{-1}(c), Df(x)$ is full rank,

then $f^{-1}(c)$ is a manifold of dimension $m - n$

Proof for GVD Dimension

$$f(x) = d_i(x) - d_j(x), \quad f : R^m \to R, \quad c = 0$$

$$(d_i - d_j)^{-1}(0) \text{ s.t. } D(d_i - d_j) \text{ is full rank}$$

$$\Rightarrow \dim((d_i - d_j)^{-1}(0)) = m - n = m - 1$$

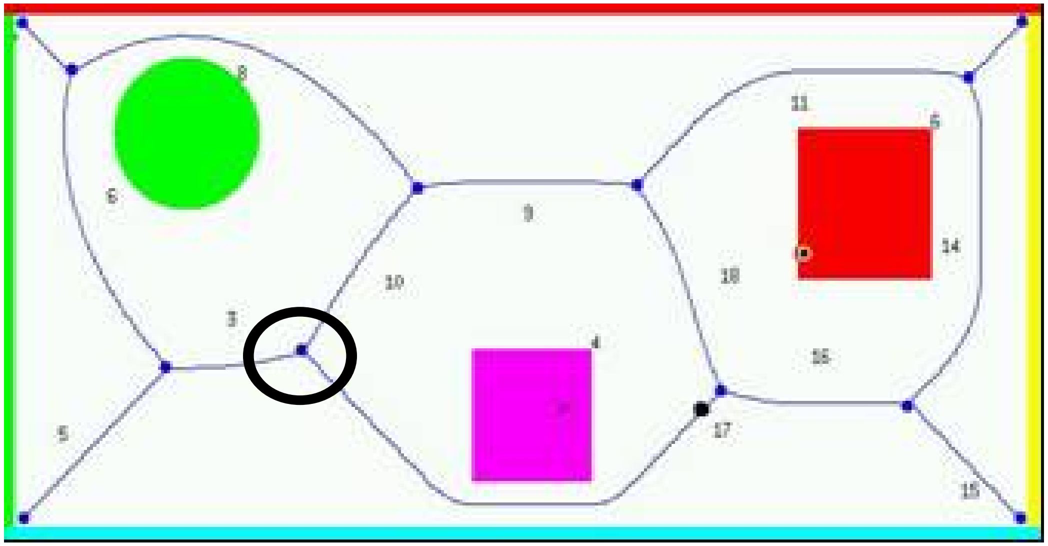
$$\text{Show } D(d_i - d_j) \text{ is full rank} \Leftrightarrow$$

$$D(d_i - d_j) \text{ is not a 0 row vector}$$

$$\nabla d_i \neq \nabla d_j \Leftrightarrow \nabla d_i - \nabla d_j \neq 0 \Leftrightarrow \nabla(d_i - d_j) \neq 0 \Leftrightarrow D(d_i - d_j)$$

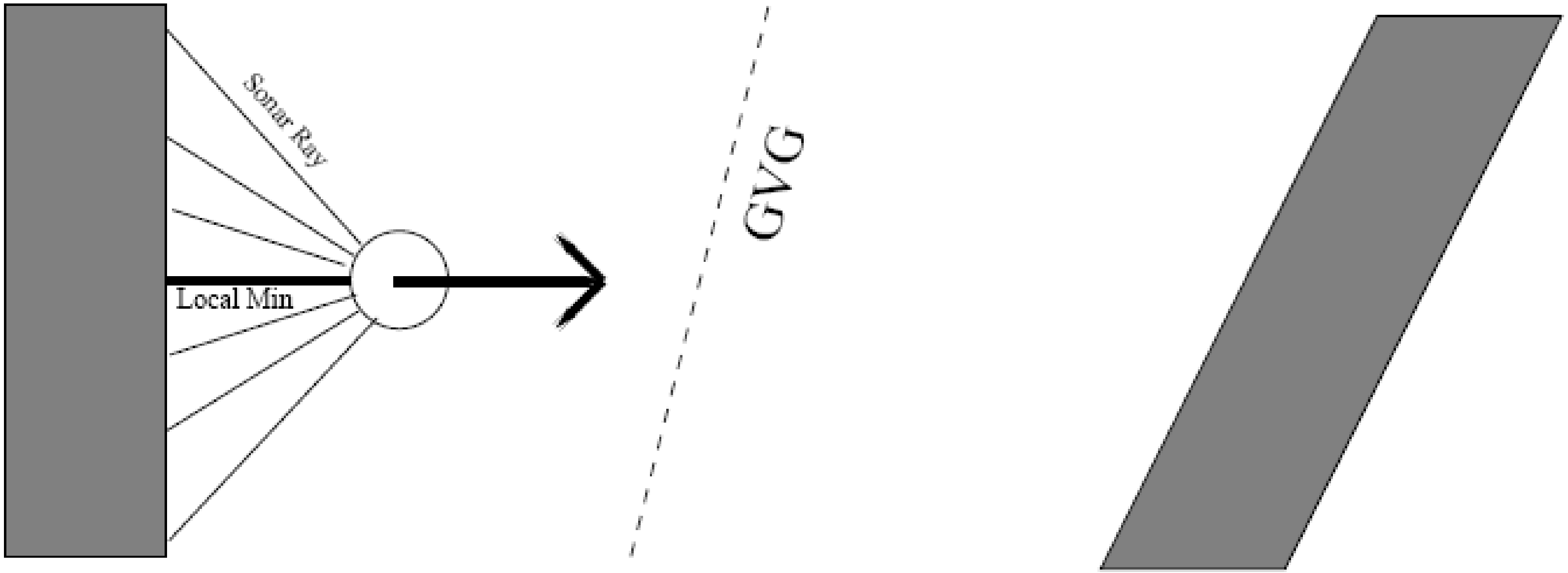
More on GVD (cont.)

Is the GVD a 1-Dimension manifold?

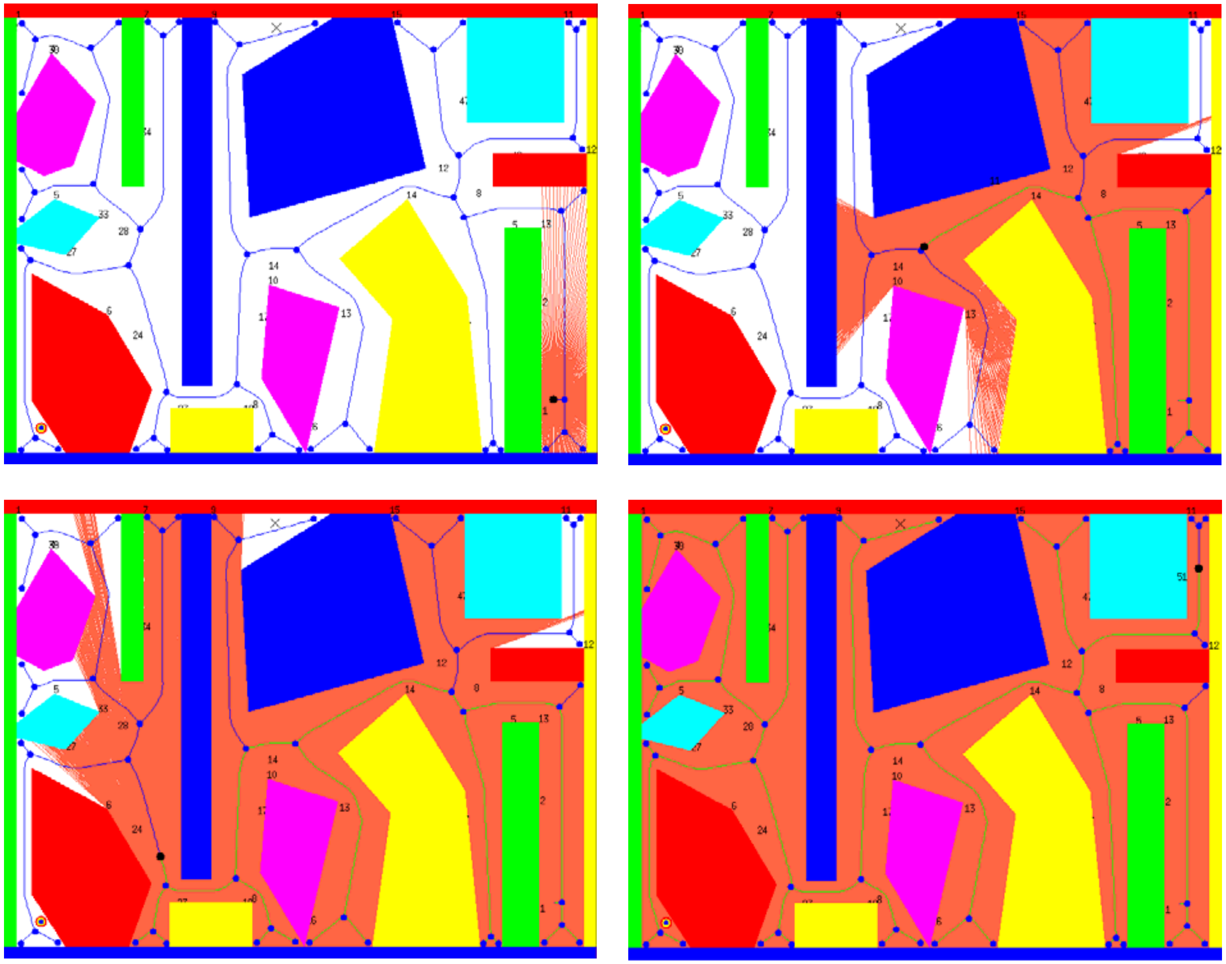


No, but it’s the union of 1D manifolds

Accessibility (in the Plane)



Departability



$$\forall\ \ q \in Q_{\text{free}},\ \ \exists q' \in\ \text{GVD such that}\ \ sq + (1-s)q' \in Q_{\text{free}}\ \ \forall s \in [0,1].$$

GVD Connected?

Proof:

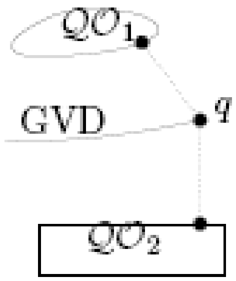
$$\mathrm{Im} : Q_{\mathrm{free}} \rightarrow GVD$$

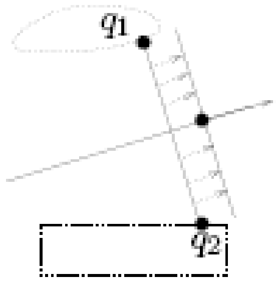
- Im is continuous (Prof. Yap, NYU)
- Im of a connected set, under a continuous map, is a connected set

$$\because \text{for each connected component of } Q_{\mathrm{free}},$$
$$GVD \text{ is connected.}$$

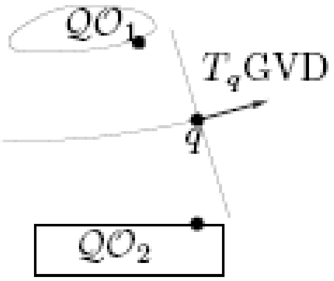
Traceability in the Plane

Tangent





Pass a line through two closest points on two closest obstacles

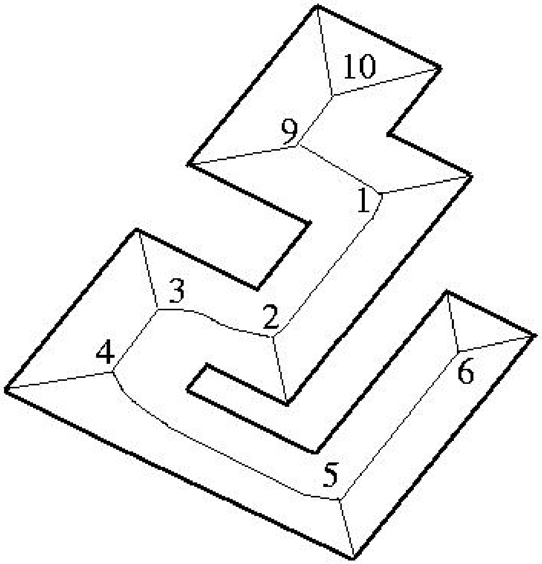


Orthogonal is tangent

Correction

$$y^{k+1} = y^k - (\nabla_y G)^{-1} G(y^k, \lambda^k)$$

Control Laws



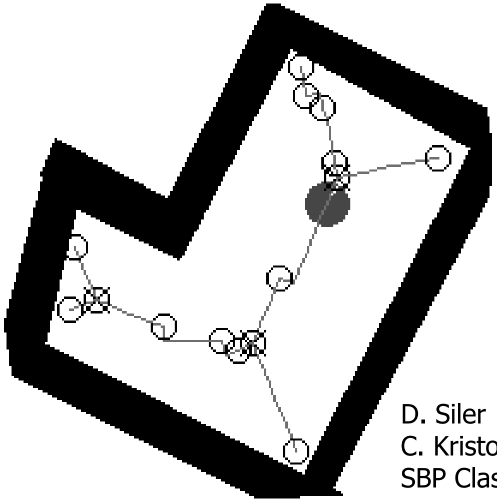
$$\underline{\text{Edge}}\ :\ G(x) = 0 = d_i(x) - d_j(x)$$

$$\dot{x} = \alpha\, \mathrm{Null}(\nabla G(x)) + \beta(\nabla G(x))^{\dagger} G(x)$$

where

- α and β are scalar gains
- $\mathrm{Null}(\nabla G(x))$ is the null space of $\nabla G(x)$
- $(\nabla G(x))^{\dagger}$ is the Penrose pseudo inverse of $\nabla G(x)$, i.e.,

$$(\nabla G(x))^{\dagger} = (\nabla G(x))^{T}(\nabla G(x)(\nabla G(x))^{T})^{-1}$$



$$\underline{\text{Meet Point}}\ :\ G(x) = 0 = \begin{Bmatrix} d_i(x) - d_j(x) \\ d_i(x) - d_k(x) \end{Bmatrix}$$

$$\dot{x} = \alpha\, \mathrm{Null}(\nabla G(x)) + \beta(\nabla G(x))^{\dagger} G(x)$$

D. Siler
C. Kristoff
SBP Class

Algorithm for exploration

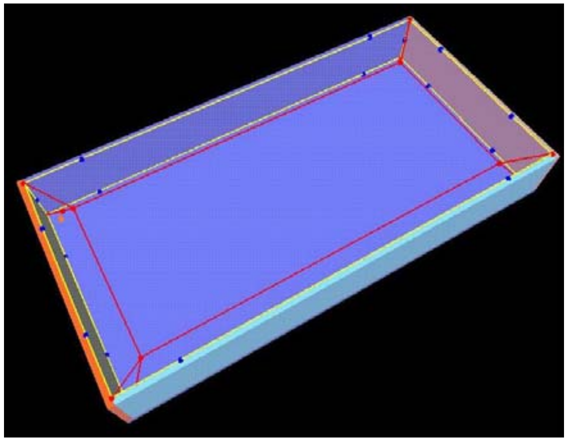
- Trace an edge until reach a meet point or a boundary point
- If a boundary point, return to the previous meet point, otherwise pick a new edge to trace
- If all edges from meet point are already traced, search the graph for a meet point with untraced edges
- When all meet points have no untraced edges, complete.

Demo

General Voronoi Graph

- In 3-Dimensions

$$F_{ijk} = F_{ij} \cap F_{ik} \cap F_{jk}$$



- In *m*-Dimensions

$$F_{ijk\ldots m} = F_{ij} \cap F_{ik} \ldots \cap F_{im}$$

$$= F_{ij\ldots m-1} \cap F_{im}$$

GVD vs. GVG

	Equidistant (#obs)	Dim	Codim
GVD	2	*m-1*	1
GVG	*m*	1	*m-1*

Proofs by Pre-Image Theorem to come

Proof for GVG Dimension

- For 3-Dimensions

$$f = \begin{pmatrix} d_i - d_j \\ d_i - d_k \end{pmatrix}, \quad f : R^3 \to R^2$$

$$f^{-1}\begin{pmatrix} 0 \\ 0 \end{pmatrix} = f^{-1}(0)$$

$$D\begin{pmatrix} d_i - d_j \\ d_i - d_k \end{pmatrix} \neq 0, \text{ since} \nabla d_i \neq \nabla d_j, \nabla d_i \neq \nabla d_k$$

Proof for GVG (cont.)

- For *m*-Dimensions

$$f:\begin{pmatrix} d_{i_1}-d_{i_2} \\ . \\ . \\ . \\ d_{i_1}-d_{i_m} \end{pmatrix}, \text{where } f:R^m \rightarrow R^{m-1}$$

$$\text{By Pre - Image Theorem, } \dim(f^{-1}) = m-(m-1)=1$$

Traceability in m dimensions

- *x* is a point on the GVG
  - *normal slice plane*
  - "*sweep" coordinate*
- Define

Pass a hyperplane through the m closest points on the m closest obstacles

$$y = (z_2, \ldots z_m)$$

$$\lambda = z_1$$

Tangent is orthogonal to this hyperplane

$$G(y,\lambda) = \begin{bmatrix} (d_1 - d_2)(y,\lambda) \\ (d_1 - d_3)(y,\lambda) \\ . \\ . \\ . \\ (d_1 - d_m)(y,\lambda) \end{bmatrix}$$

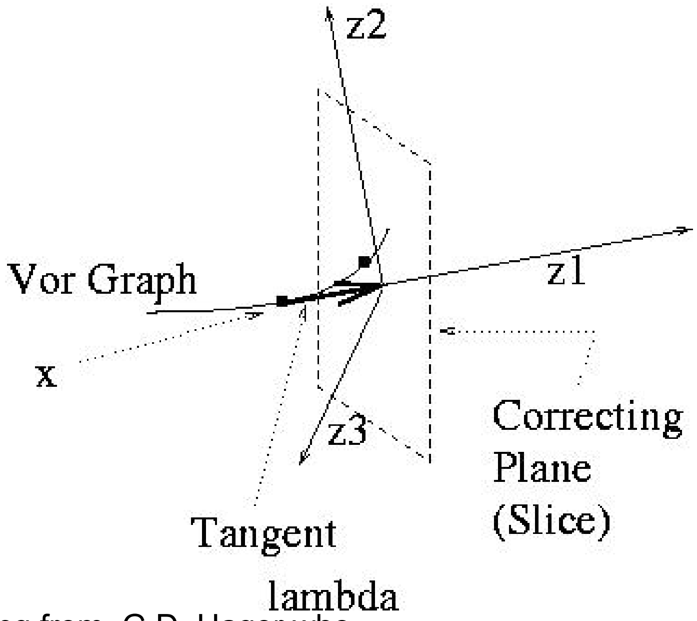
$$\text{where } G : R^{m-1} \times R \to R^{m-1}$$

Traceability (cont.)

Predictor-corrector scheme

- Take small step, $\Delta\lambda$ in z_1 direction (tangent).
- Correct using iterative Newton’s Method

$$y^{k+1} = y^k - (\nabla_y G)^{-1} G(y^k, \lambda^k)$$



Accessibility

Gradient Ascent: Cascading Sequence of Gradient Ascent Operations

- Move until F_{ij}
- Maintain 2-way equidistant while $\uparrow D$

$$\prod_{T_x F_{ij}} \nabla D = \prod_{T_x F_{ij}} \nabla d_i = \prod_{T_x F_{ij}} \nabla d_j$$

GVG Connected?

$$F_{ijk} \subseteq \partial F_{ij}$$

$$F_{ijk} \subseteq \partial F_{ik}$$

$$F_{ijk} \subseteq \partial F_{jk}$$

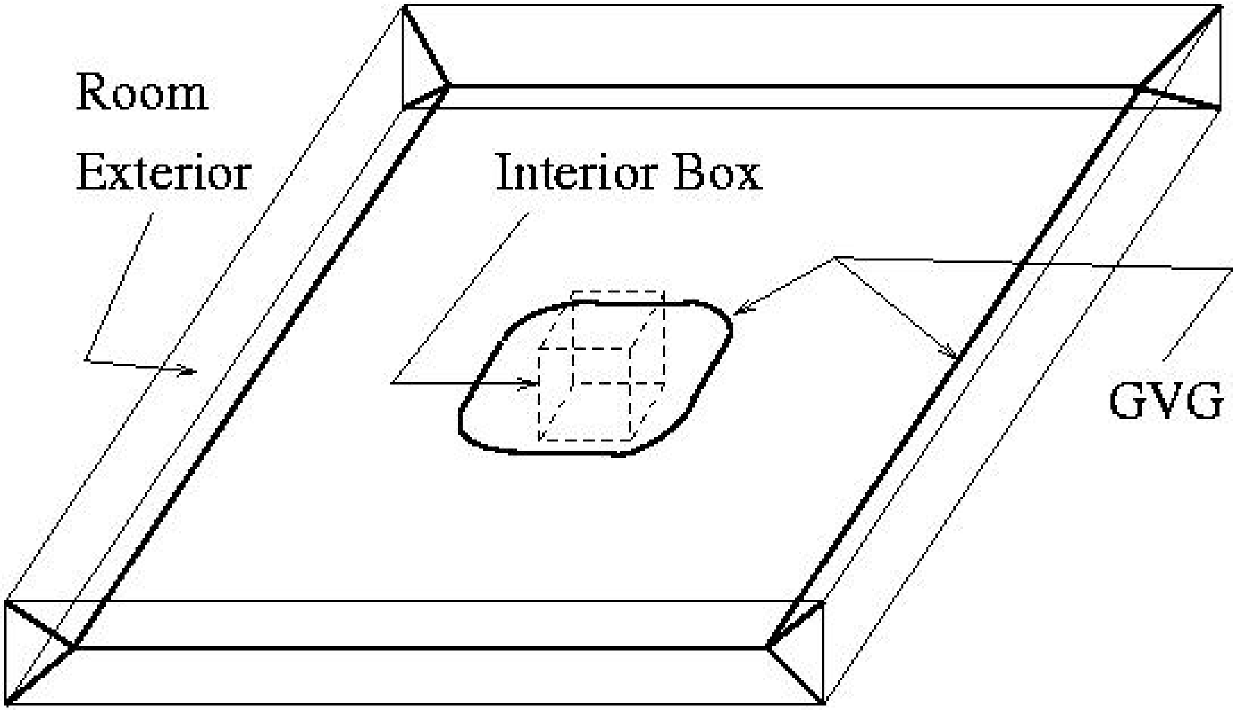
Assuming ∂F_{ij} is connected $\forall F_{ij}$

Is GVG connected?

GVG Connected?

is not connected

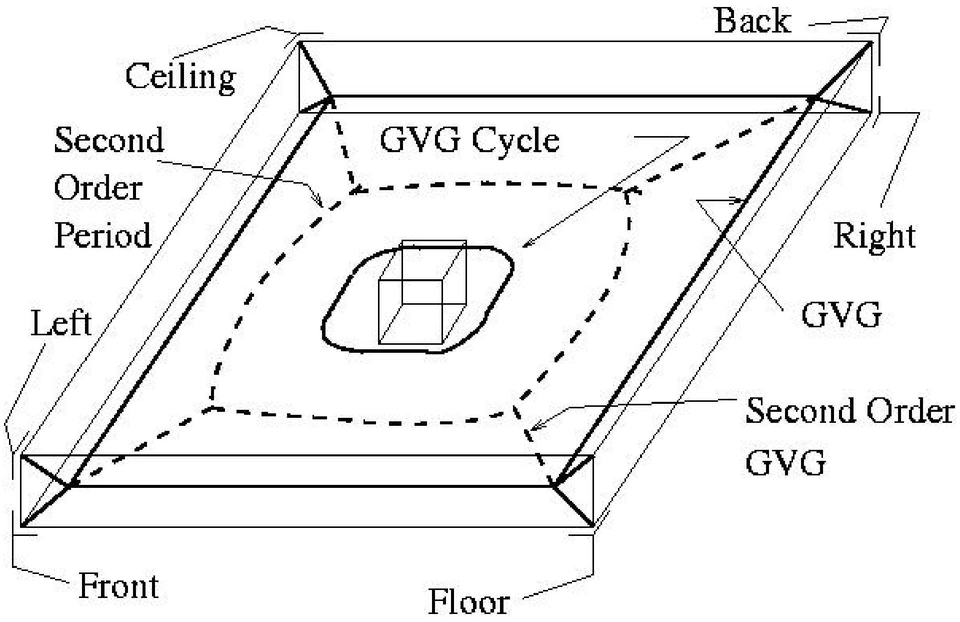
∂F_{ij}



GVG^2

Second-order two-equidistant surface

$$F_k |_{F_{ij}} = \{x \in F_{ij} : \forall h \neq i, j, k,$$
$$d_h(x) > d_k(x) > d_i(x) = d_j(x) > 0$$
$$\text{and } \nabla d_i \neq \nabla d_j\}$$



Linking to GVG Cycle

- Detect GVG Cycle
- Gradient Descent

$$-\prod_{T_x F_{ij}} \nabla d_k$$

- ∇d_k increases distance to C_k
- $-\nabla d_k$ decreases distance to C_k
- $\prod$ projection
- $T_x F_{ij}$ tangent space of
- $-\prod_{T_x F_{ij}}$ projection onto the tangent space

From GVG^2 to GVG Cycle

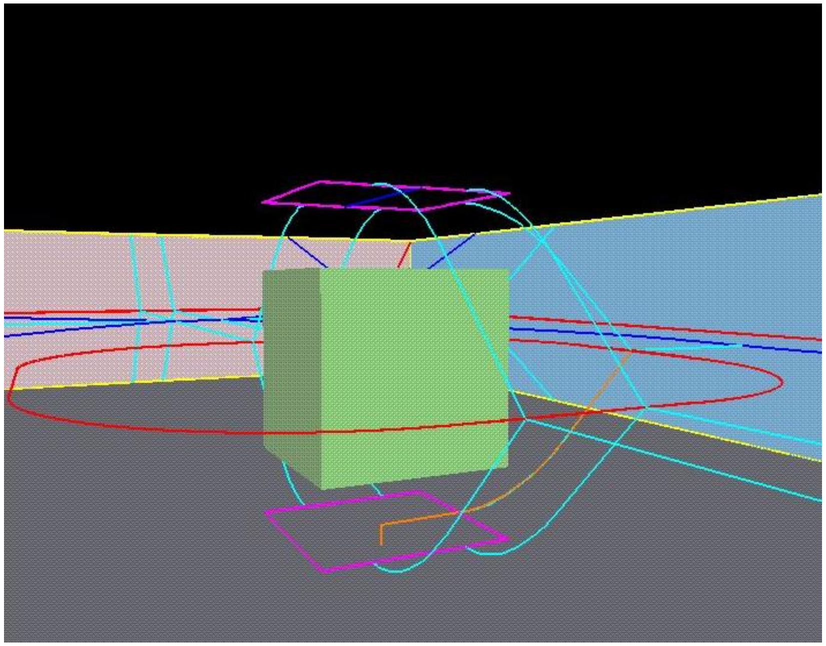
$$\dot{c}(x) = -\prod\nolimits_{T_x F_{ij}} \nabla d_k c(t)$$

Assuming $-\prod_{T_x F_{ij}} \nabla d_k$ is never 0.

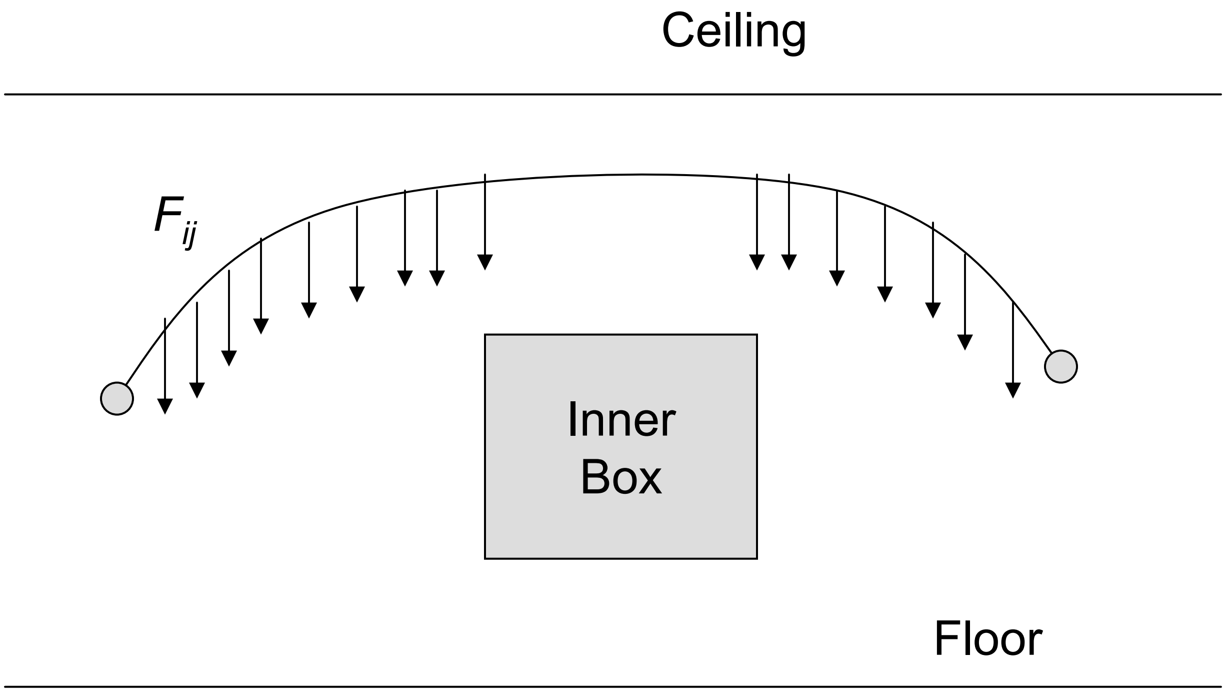
$$\begin{tikzcd}
 & d_i = d_j < d_k \arrow[dl] \arrow[dr] & \\
(d_i = d_j)\downarrow, d_k \Downarrow \arrow[dr] & & (d_i = d_j)\uparrow, d_k \downarrow \arrow[dl] \\
 & d_i = d_j = d_k &
\end{tikzcd}$$

Two Problems

- Gradient goes to 0?
- Going on top of the box
    - Define occluding edges



Finding Occluding Edges



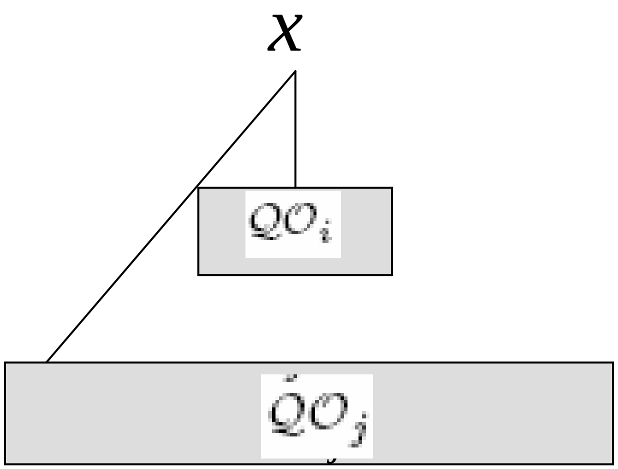
Visible Distance Revisited

- *Single-object*

$$d_i(x) = \begin{cases} \text{distance to } QO_i & \text{if } c_i \in \tilde{C}_i(x) \\ \infty & \text{otherwise} \end{cases}$$

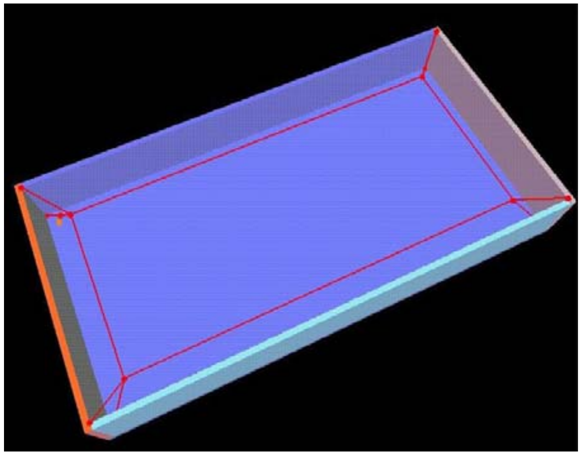
- *Multi-object*

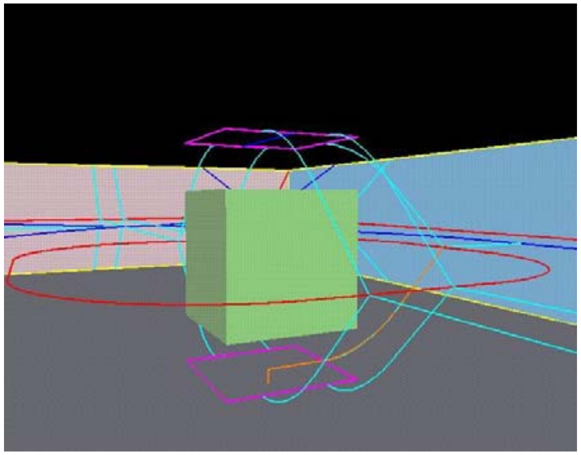
$$D(x) = \min_i d_i(x)$$



Occluding Edges (cont.)

- Change in second closest object
  - GVG two-equidistant edges (continuous)
  - Occluding edges (not continuous)
- Questions?
  - When to link?
  - Do we have all possible edges?



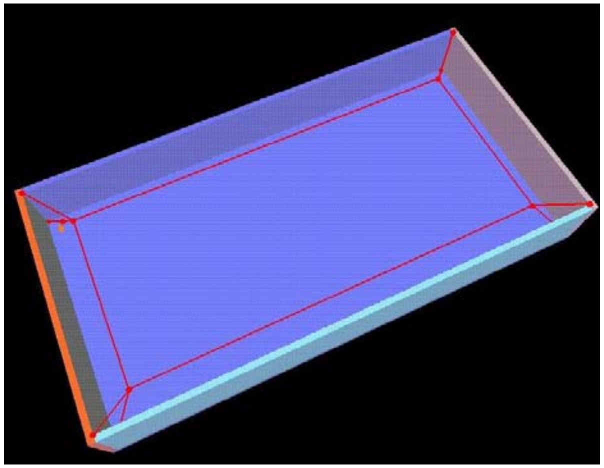


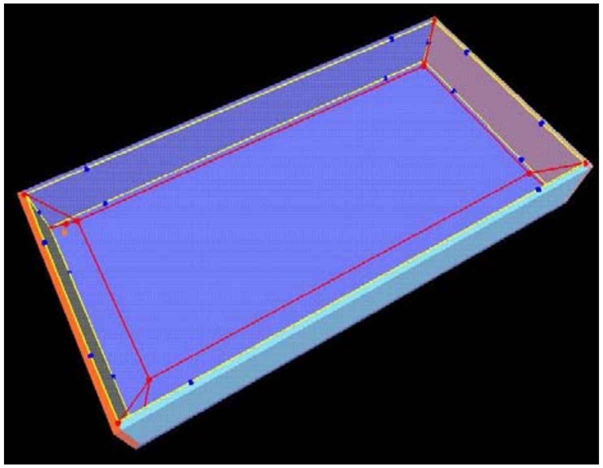
More Linking

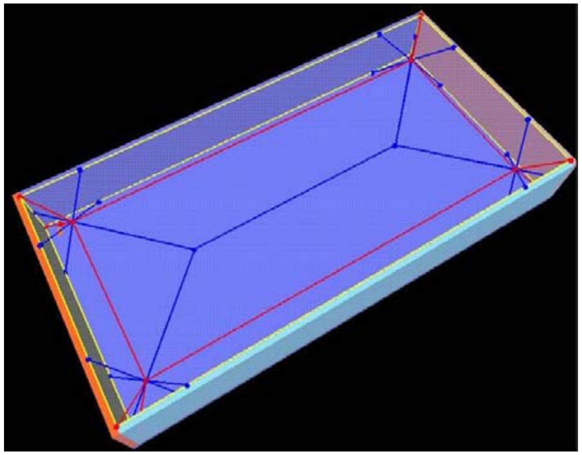
$$F_k \,|_{F_{ij}} = \{x \in F_{ij} : \forall h \neq i, j, k,$$
$$d_h(x) > d_k(x) > d_i(x) = d_j(x) > 0$$
$$\text{and}\, \nabla d_i \neq \nabla d_j\}$$

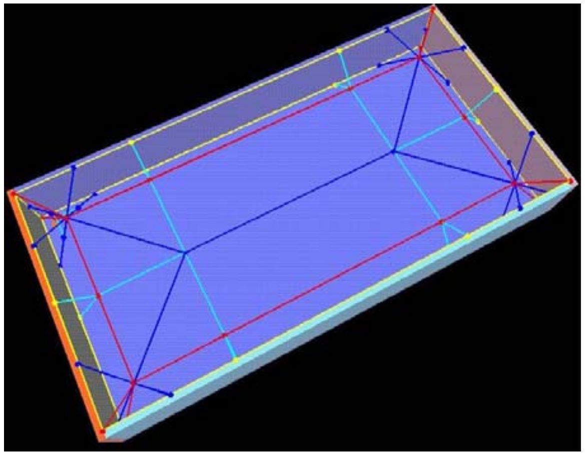
- GVG^2 $\quad d_h = d_k > d_i = d_j$
- Occluding edges $\quad d_h \enclose{circle}{>} d_k > d_i = d_j$
- GVG Edge $\quad d_h > d_k = d_i = d_j > 0$
- Boundary Edge $\quad d_h > d_k > d_i = d_j = 0$
- Floating boundary edge $\quad \nabla d_i = \nabla d_j$

Basic Links

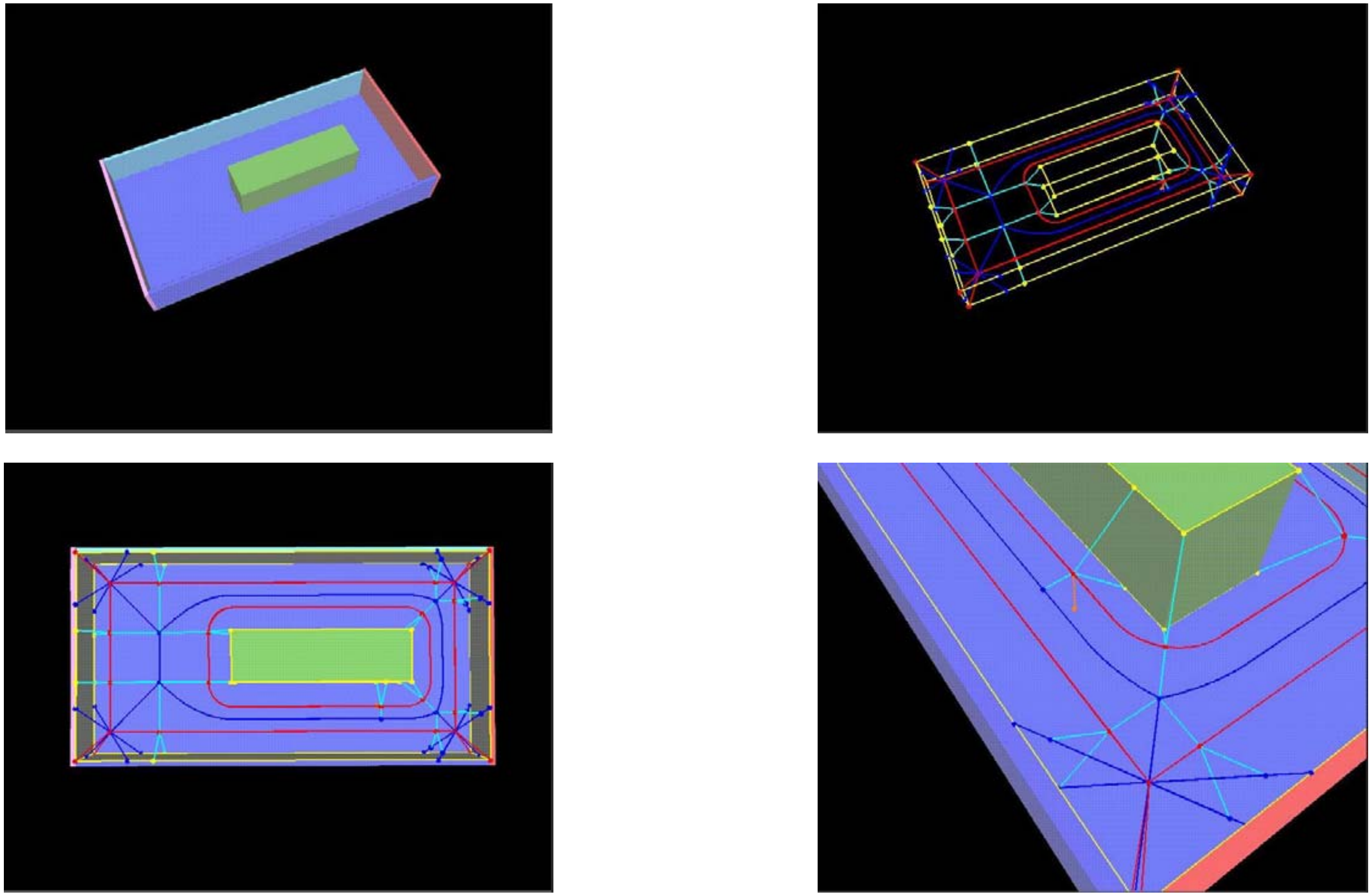




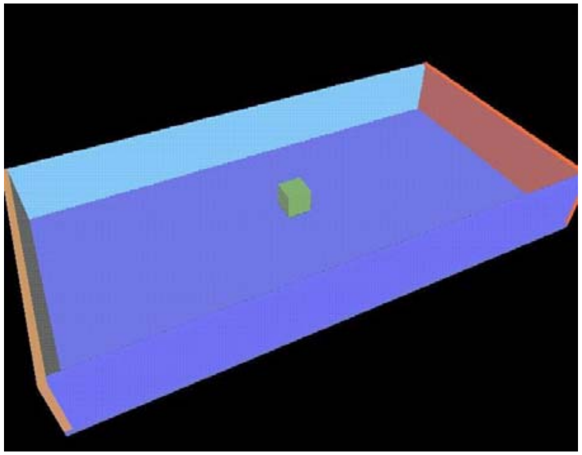


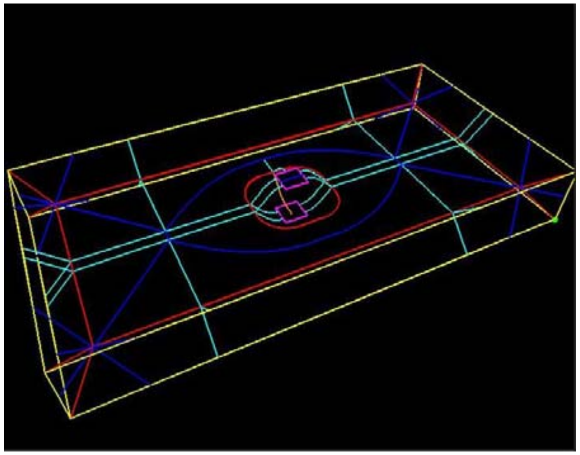


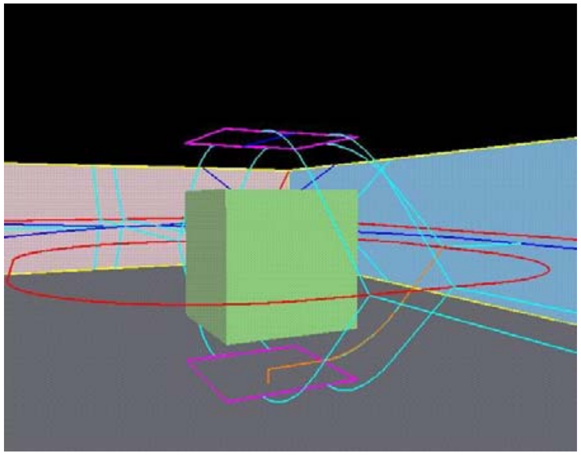
Room with Box

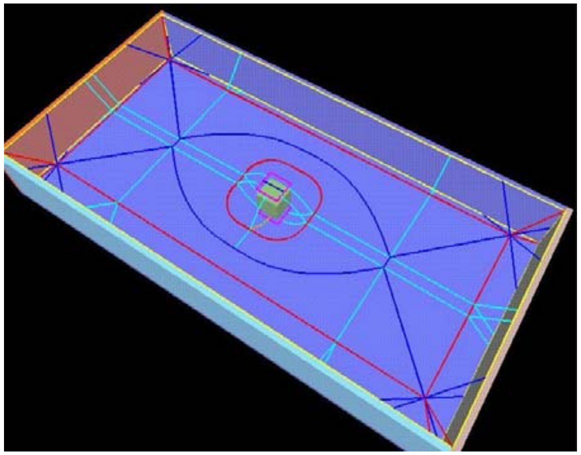


Floating Box

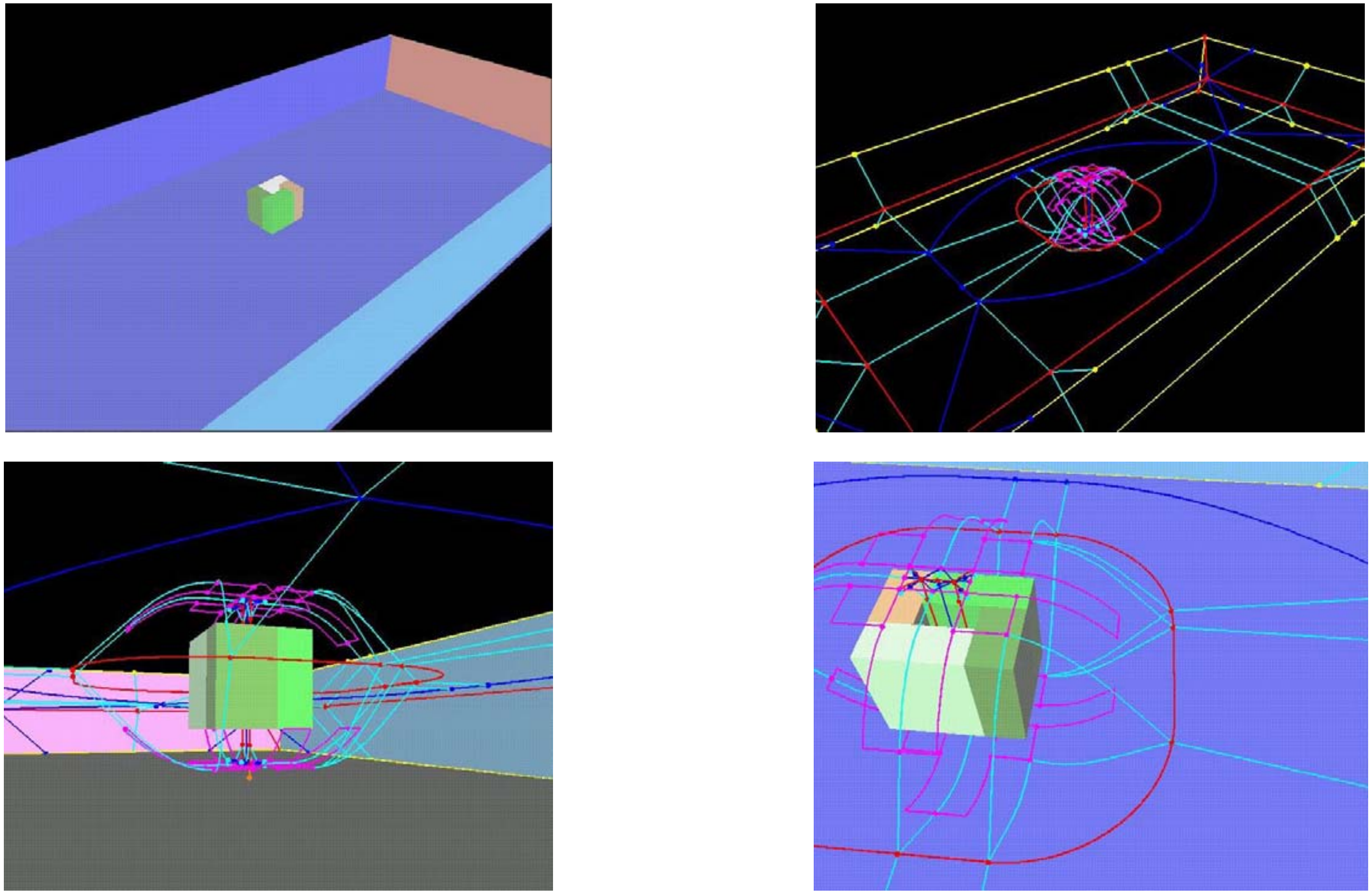




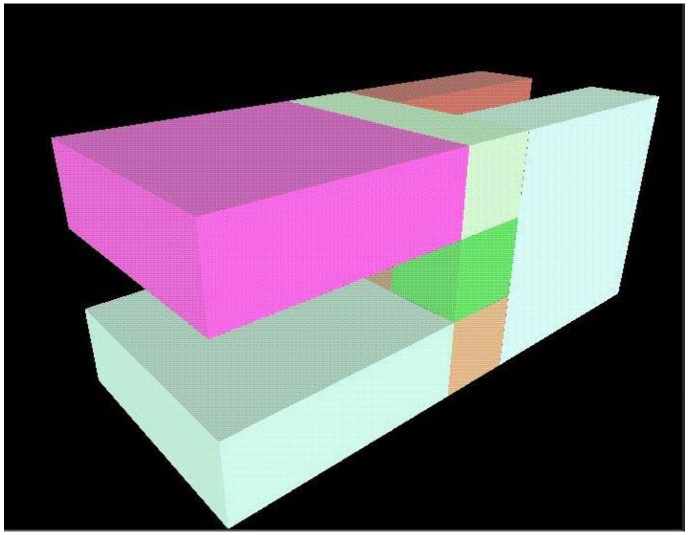


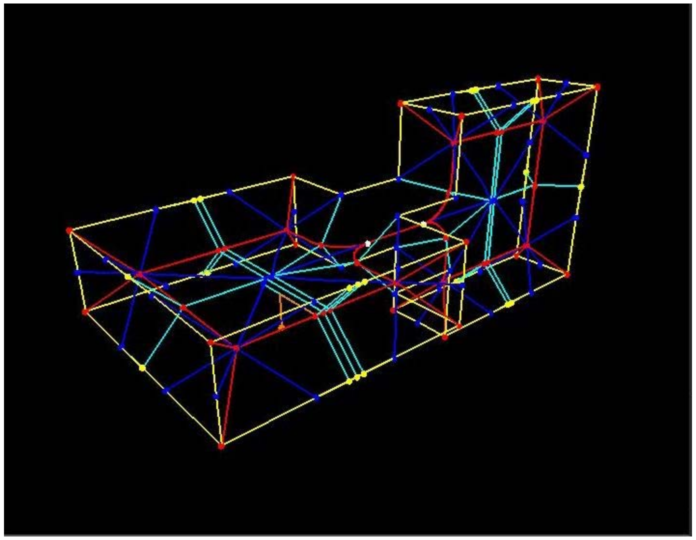


Box with Opening



This is COMPLICATED!





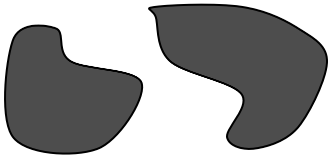
Topological Maps (Kuipers)

- Topological map represents spatial properties of actions and of places and paths in the environment. Topological map is defined as the minimal models of an axiomatic theory describing the relationship between the different sources of information explained by map (Remolina and **Kuipers**, Artificial Intelligence, 2003)
- Topological maps represent the world as a graph of places with the arcs of the graph representing movements between places (Kortenkamp & Weymouth, AAAI-94)
- Topological maps represent the robot environment as graphs, where nodes corresponds to distinct places, and arcs represent adjacency. A key advantage of topological representations is their compactness (Thrun, et at. 1999)
- Topological localization uses a graph representation that captures the connectivity of a set of features in the environment (Radhakrishnan & Nourkbash, IROS 1999)

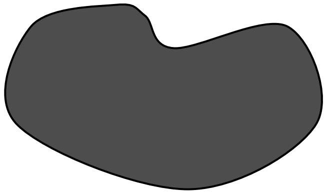
Topology (Really, Connectivity)

- A ***topology*** is a collection ***T*** of subsets of ***X***
  - $\emptyset, X \in T, \quad a_1 \cup a_2 \cup \ldots \in T, \quad a_1 \cap a_2 \cap \ldots \cap a_n \in T$
  - *What is the relevance?*

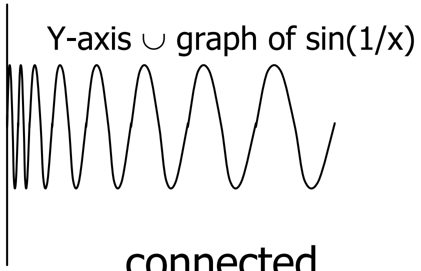
- ***path connected***



not connected

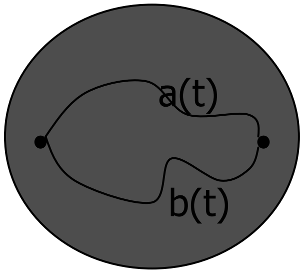


Connected,
path connected

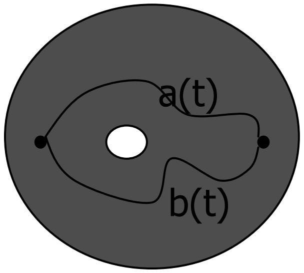


connected,
but not path connected

- ***simply connected (contractible)***



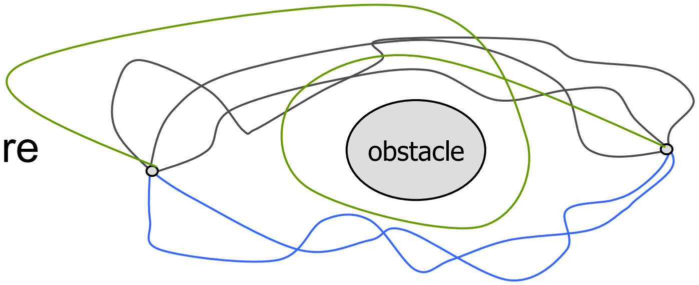
simply connected



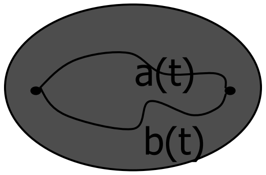
not simply
connected

Homotopy

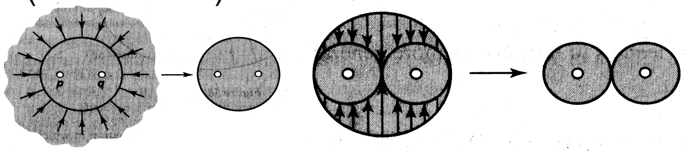
- Two paths $f, f' : [0,1] \to X$, are ***path-homotopic***
  $F(s,0)=f(s)$ and $F(s,1)=f'(s)$;
  $F(0,t)=x_0$ and $F(1,t)=x_1$ $\forall \in$s [0,1], $\forall \in$t [0,1]



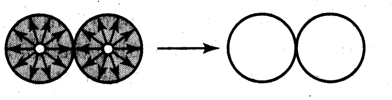
- ***Path-Homotopy class [f]*** set of the mappings that are path-homotopic to f.



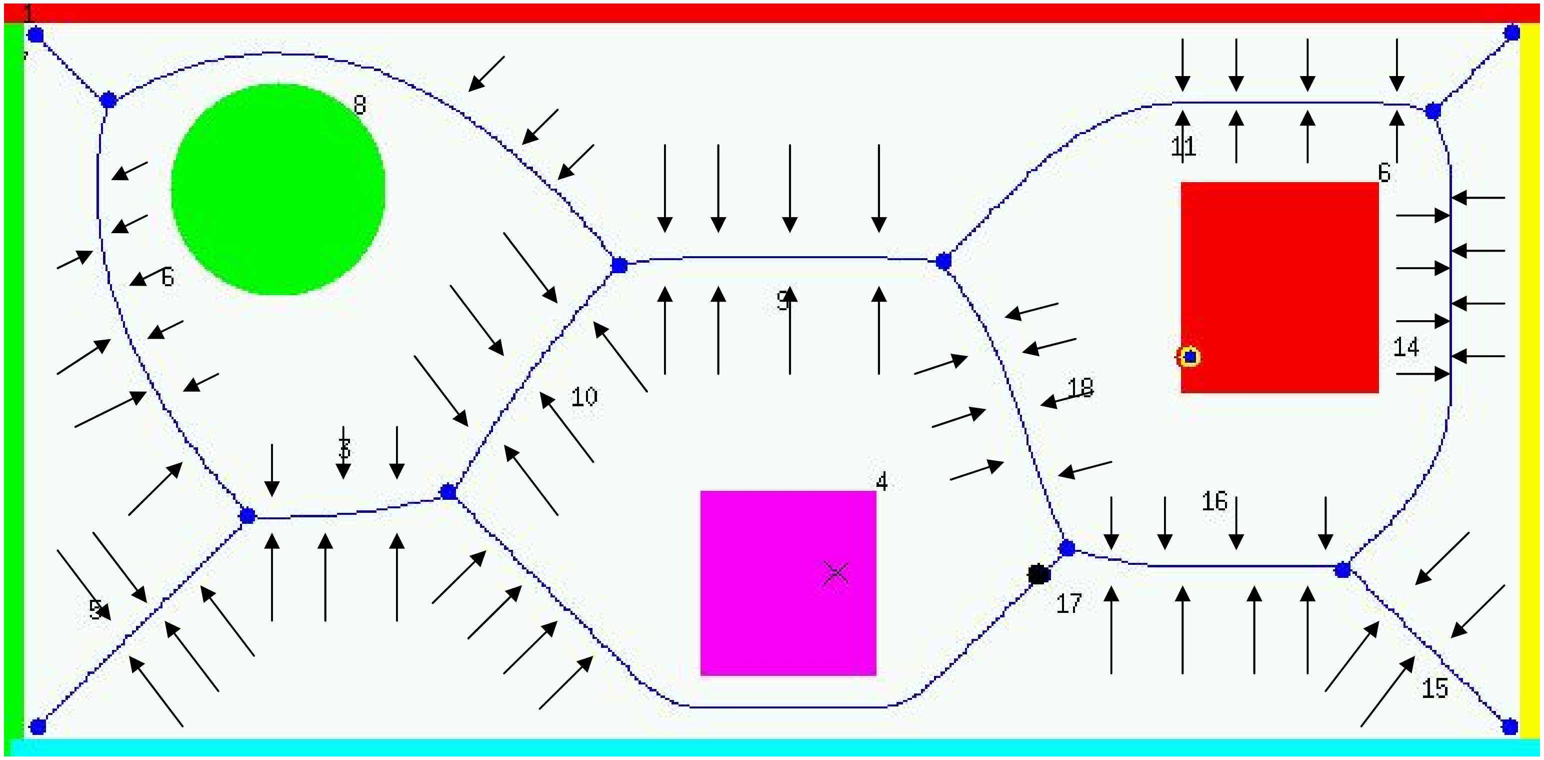
- ***Fundamental group*** π_1 : set of path-homotopy classes
  - X is **simply connected** if π_1 is the trivial (one-element)



- ***A*** is a ***deformation retract*** iff $H: X \times [0,1] \to X$,
  $H(x,0)=x$ and $H(x,1) \in A$ $\forall x \in X$, and $H(a,t)=a$ $\forall a \in A$, $t \in [0,1]$.
  - **H** is called a **deformation retraction**



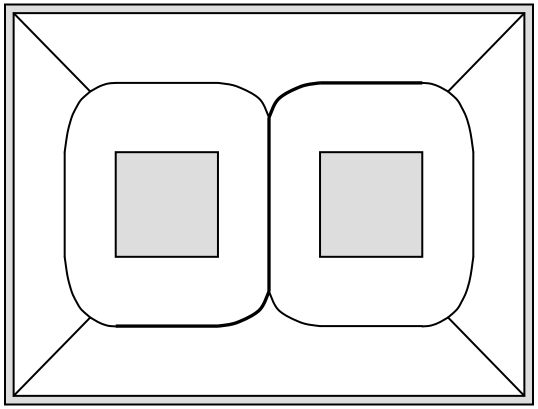
Deformation Retraction: GVG in Plane



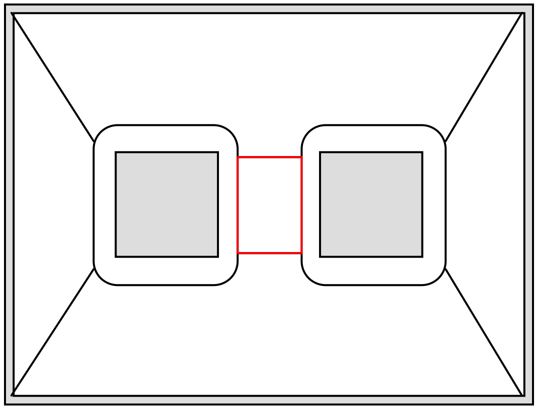
16-735, Howie Choset, with significant copying from G.D. Hager who loosely based his notes on notes by Nancy Amato

Topological Map: Good and Bad

- Topological Map: *For each homotopy class in free space, there is a corresponding homotopy class in the map.*



- Good Topological map : the first fundamental groups have the same cardinality

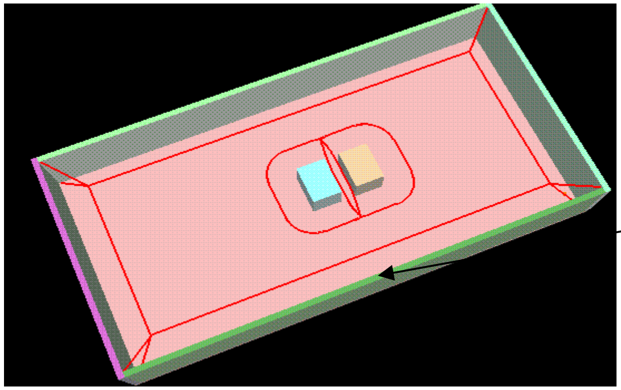


- Bad Topological map : redundant homotopy classes in the map

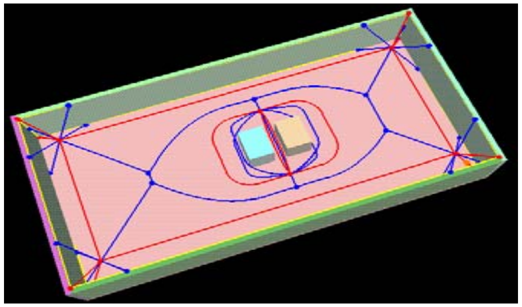
Bad Topological Maps in Higher Dims

- In general, there cannot be a one-dimensional deformation retract in a space with dimension greater than two

  → **There can not be "good" one-dimensional topological maps for $\Re^3$**



loop



- GVG in $\Re^3$: Not a good topological map

HGVG in R^3

Application of Topological Maps: Sensor Based Planning for a Rod Robot:

- Challenges
  - Three and Five dimensional Space
  - Non-Euclidean
  - Sensor Based Approach
  - Workspace -> Configuration Space

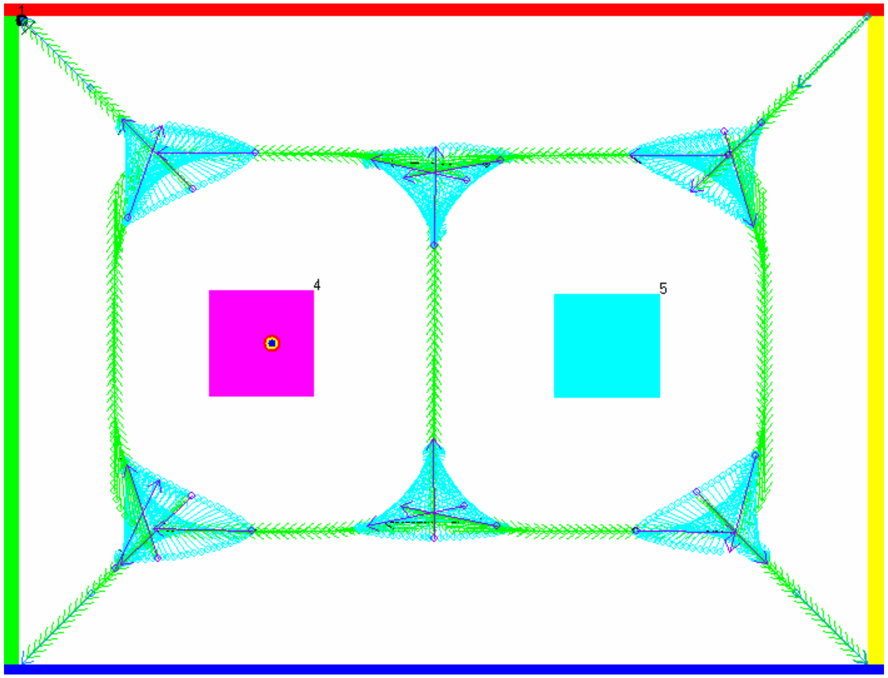
- Piecewise Retract of $R^2 \times S^1$ and $R^3 \times S^2$

$$q = (x, y, \theta)$$

$$D_i(q) = \min_{c \in C_i,\, r \in R(q)} \|c - r\|$$

C_i
Rod
Snake

Rod-HGVG

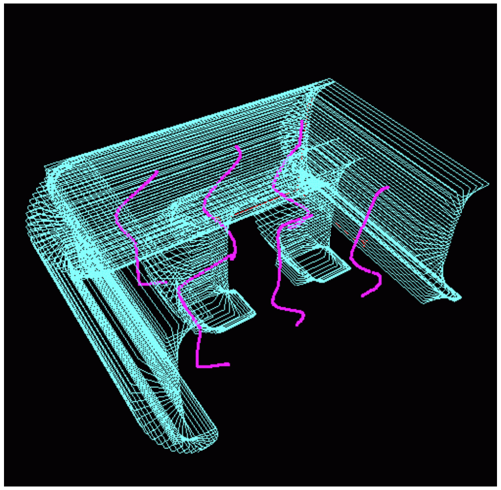


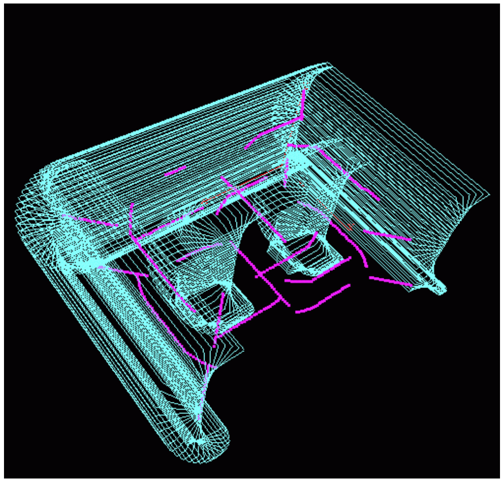
Piece-wise retract

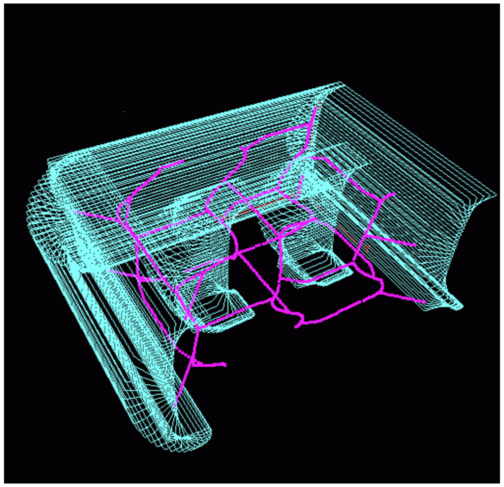
- Retract in Cspace Cells
- *Point-GVG connects retracts*

Rod-GVG's $D_i(q) = D_j(q) = D_k(q)$
(Diffeo to S^1 if rod is small enough)

1tan edges, Tan to Pt GVG
$D_i(q) = D_j(q)$



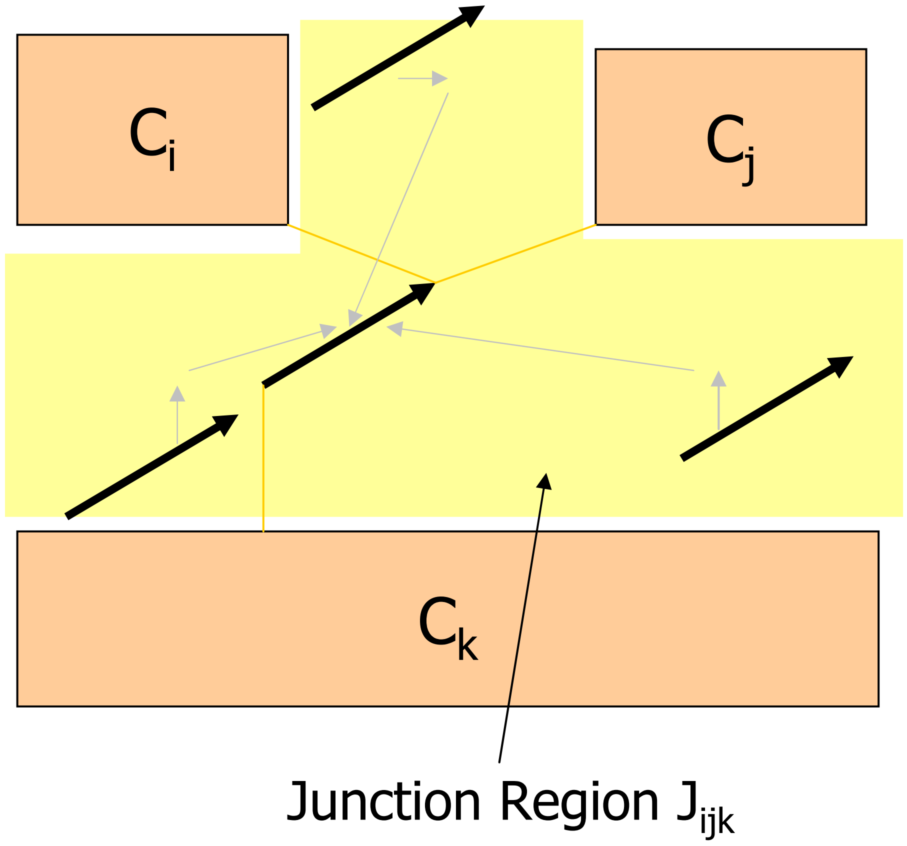




Piece-wise Retract:

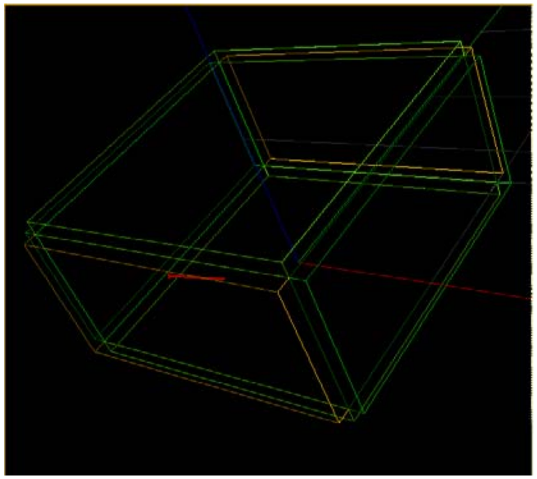
A Bad Topological Map Containing Good Topological Submaps

- Accessibilty: path between any configuration and roadmap
- Deformation Retraction:
  - $H(q, t) : SE(2) \times [0,1] \rightarrow CF_{ijk}$
  - $H(q,0) = q$
  - $H(q,1)$ = a configuration on the roadmap
  - □ $\theta(H(q,1)) = \theta(q)$
- $H(q, t)$ is continuous in Junction Region
- ***Sensor-Based Implementation***

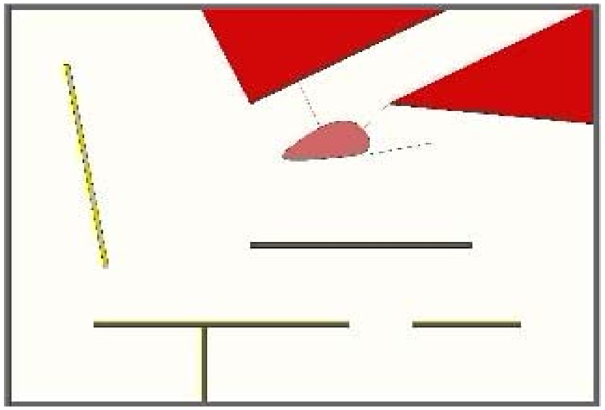


More Piece-wise Retracts

- Rod-HGVG in Three-Dimension
  - Rod-GVG edges
  - 1-tan edges
  - 2-tan edges



- Convex GVG in the plane
  - Convex GVG edges
  - Fat 1-tan edges



Maps Comprising Good Sub Maps

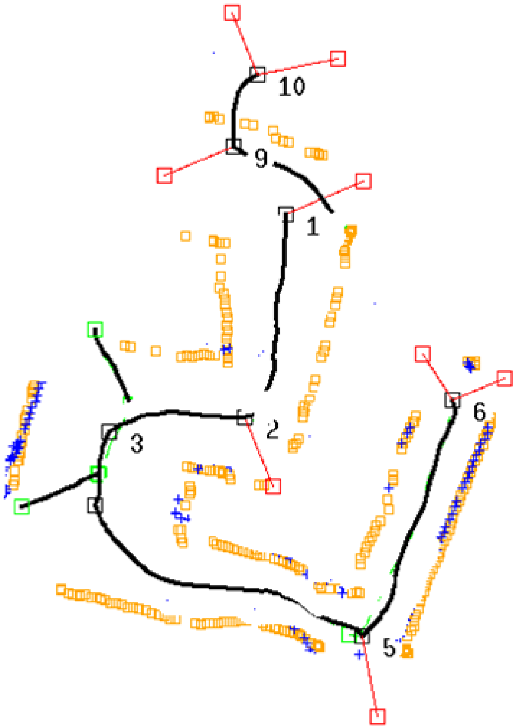
Sensor-Based Deployment

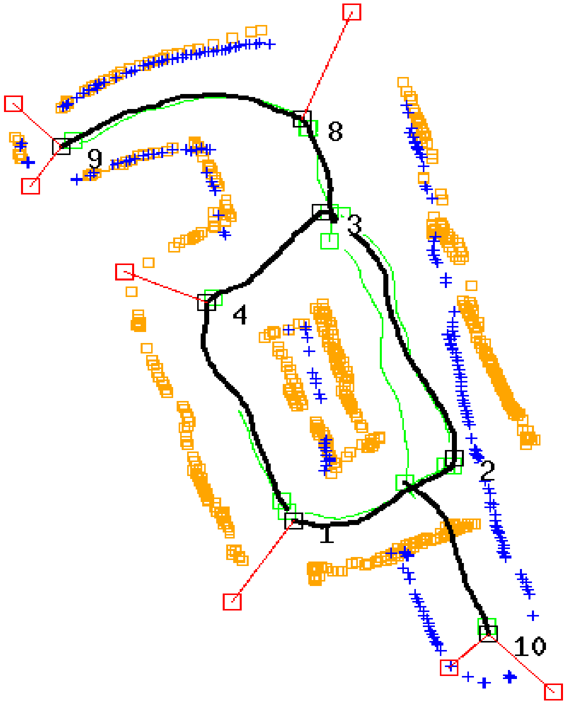
16-735, Howie Choset, with significant copying from G.D. Hager who loosely based his notes on notes by Nancy Amato

Implication for complexity??

- Base and Fiber Variables (Ostrowski and Burdick)
- Internal Shape and Position Variables
- Purely configuration and Workspace Variables

Topological Simultaneous Localization & Mapping





Node

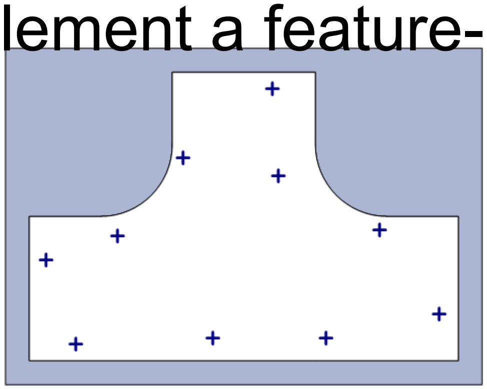
Distance to nearby obstacles
Number of emanating edges
Departure angles

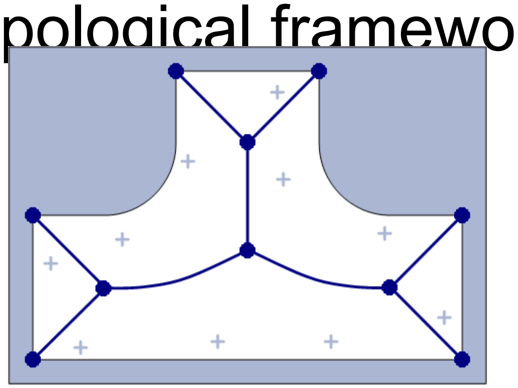
Edge

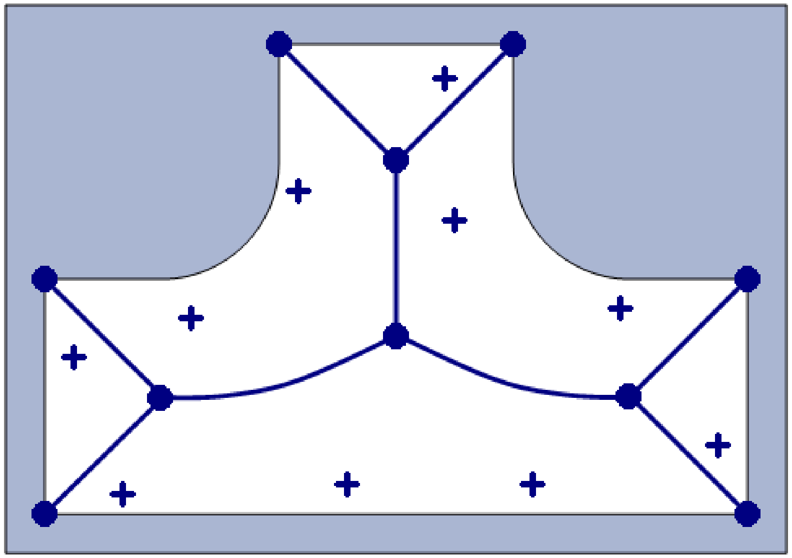
Path Length
"Correspondence"

Application: Hierarchical SLAM

Implement a feature-based technique in a topological framework





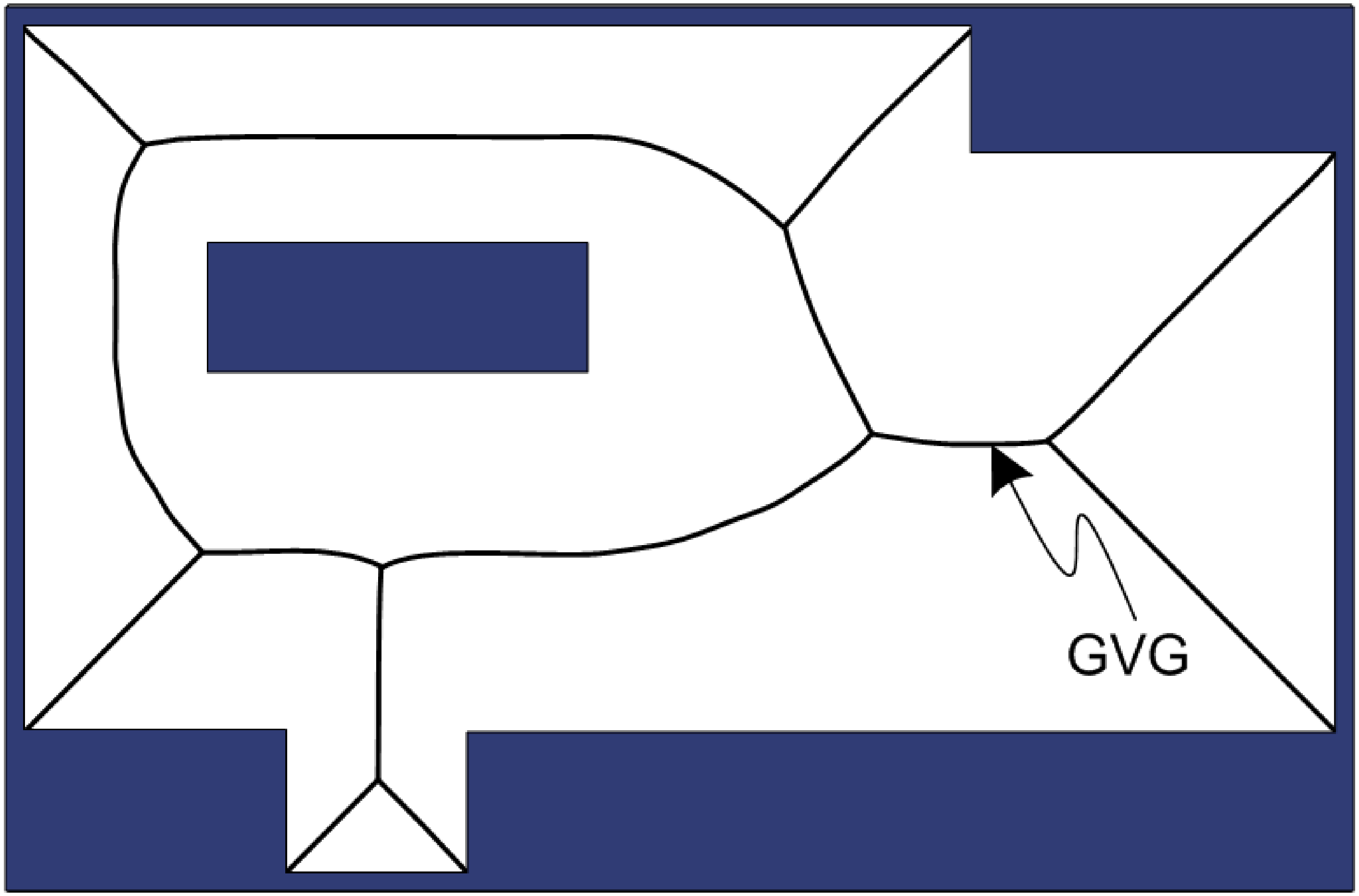


<u>Submap</u>
Chong

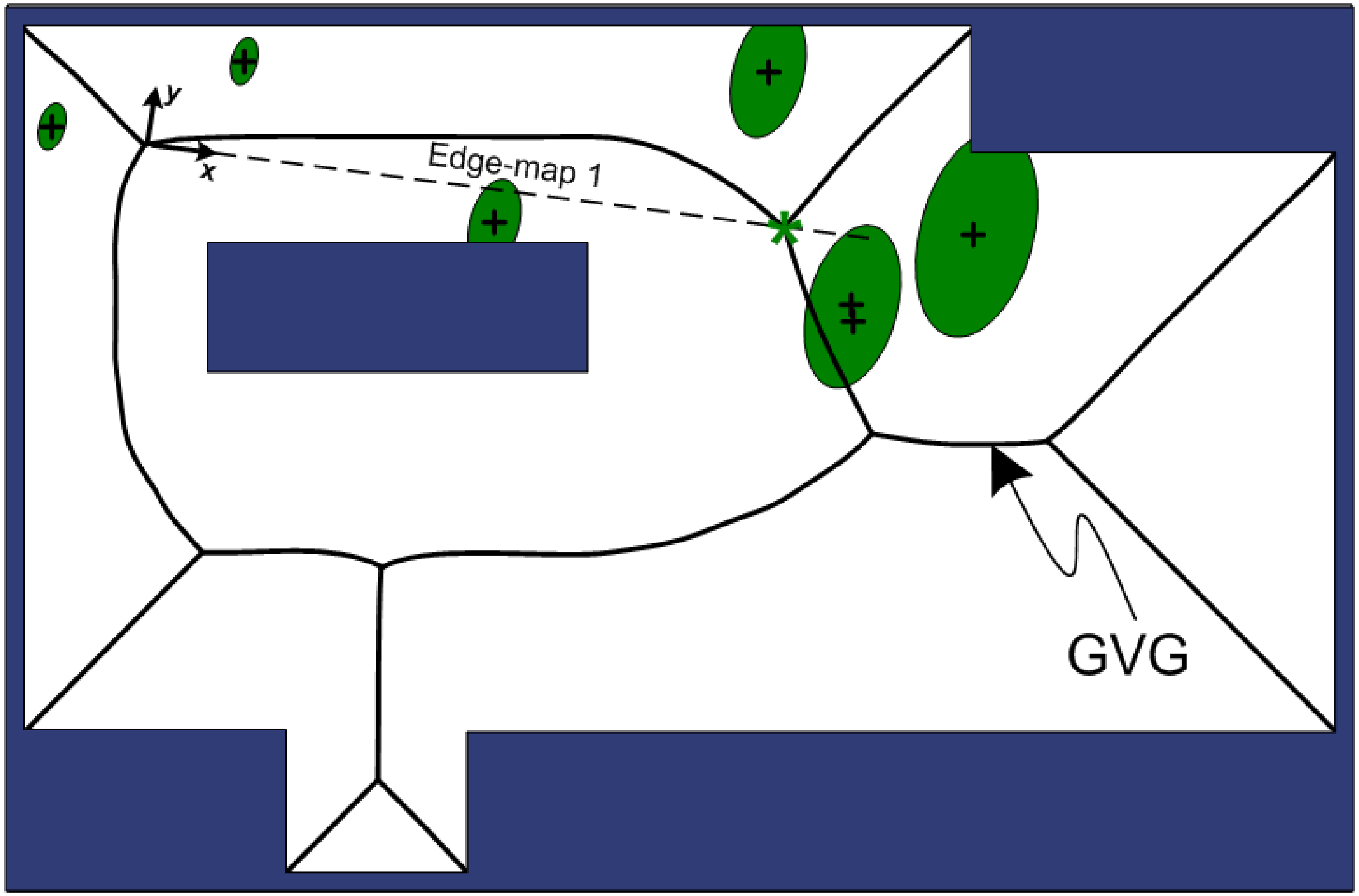
<u>Atlas</u>
Bosse & Leonard

Work with Kantor

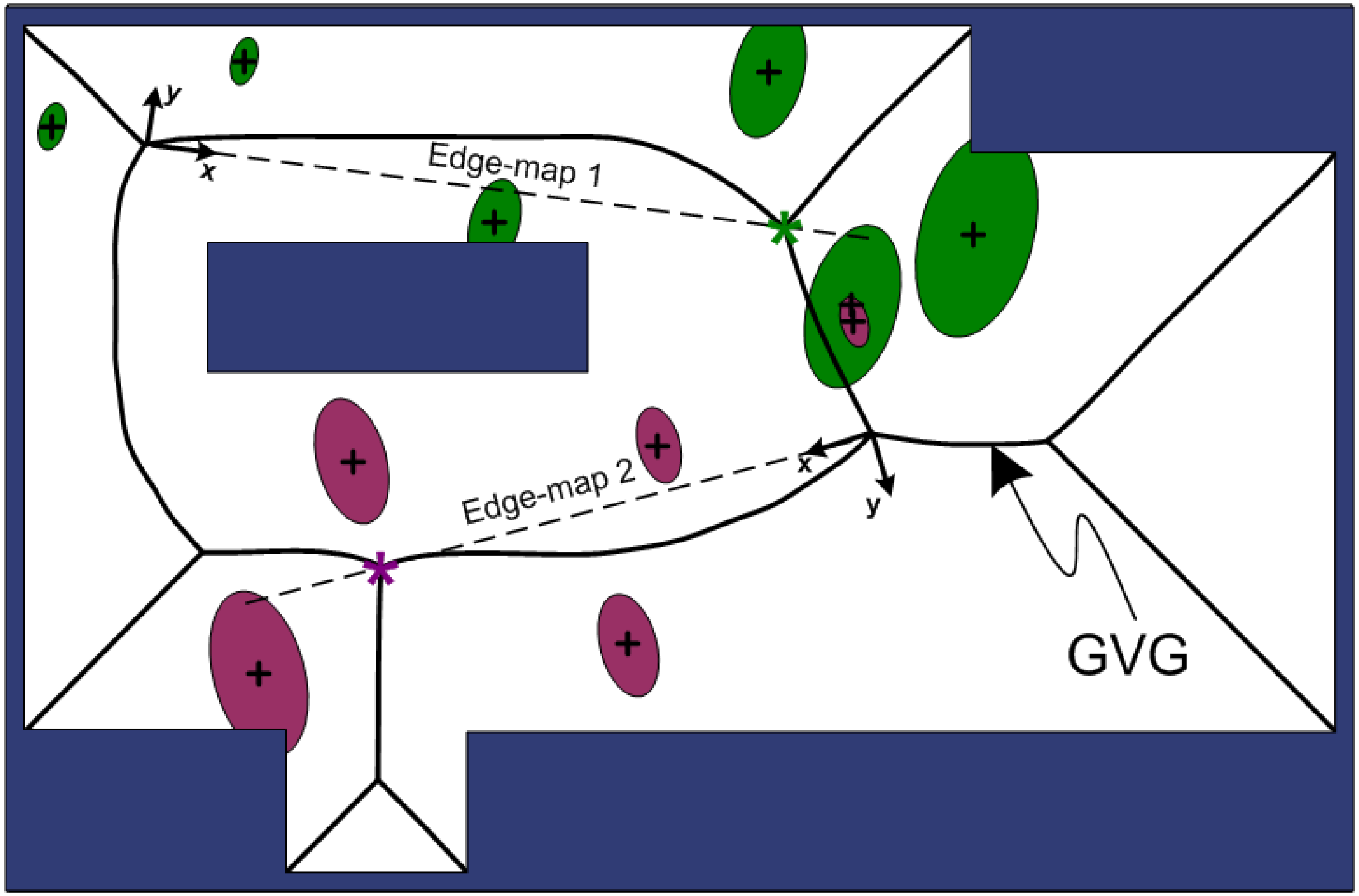
Embedded H-SLAM Map



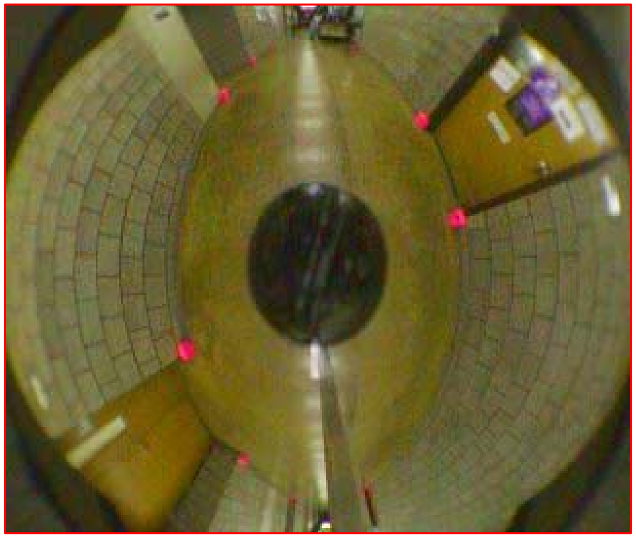
Embedded H-SLAM Map

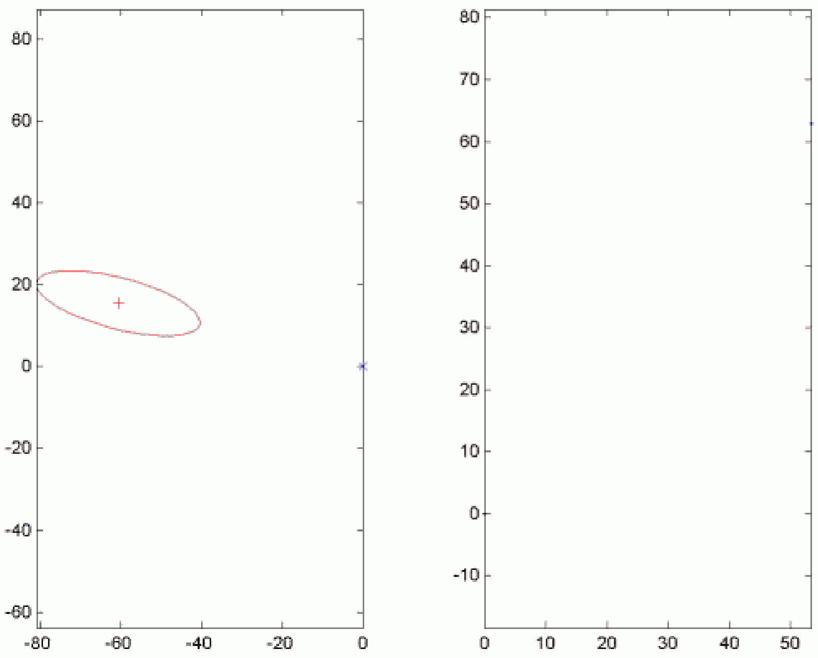


Embedded H-SLAM Map

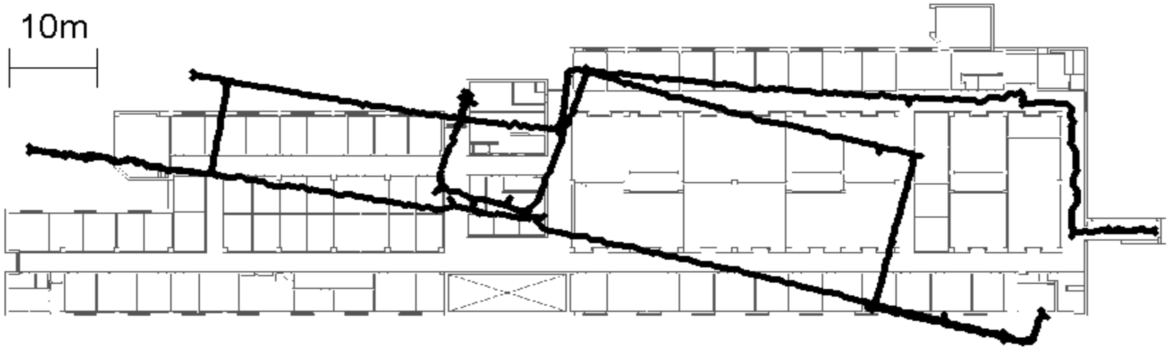


Experimental Platform

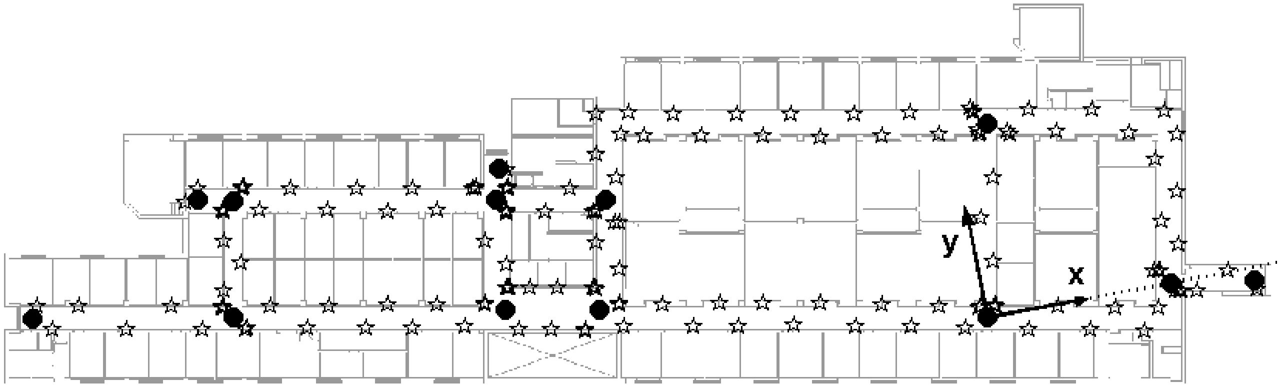




Experimental Results

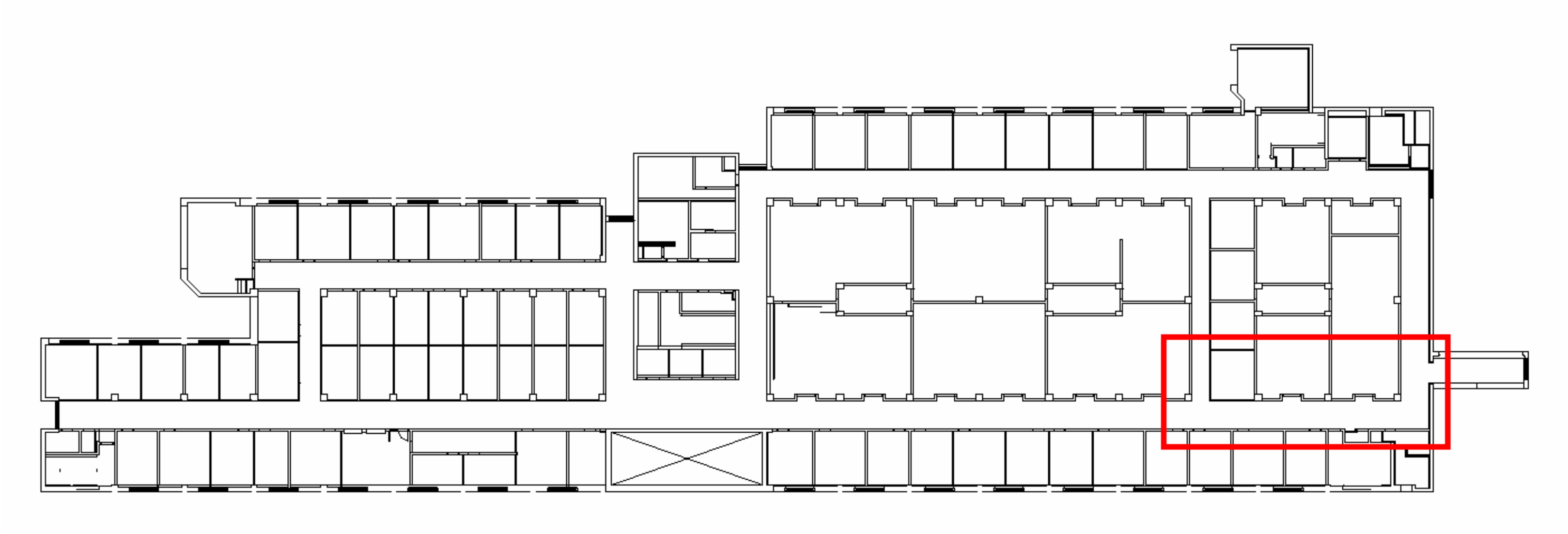


Odometry

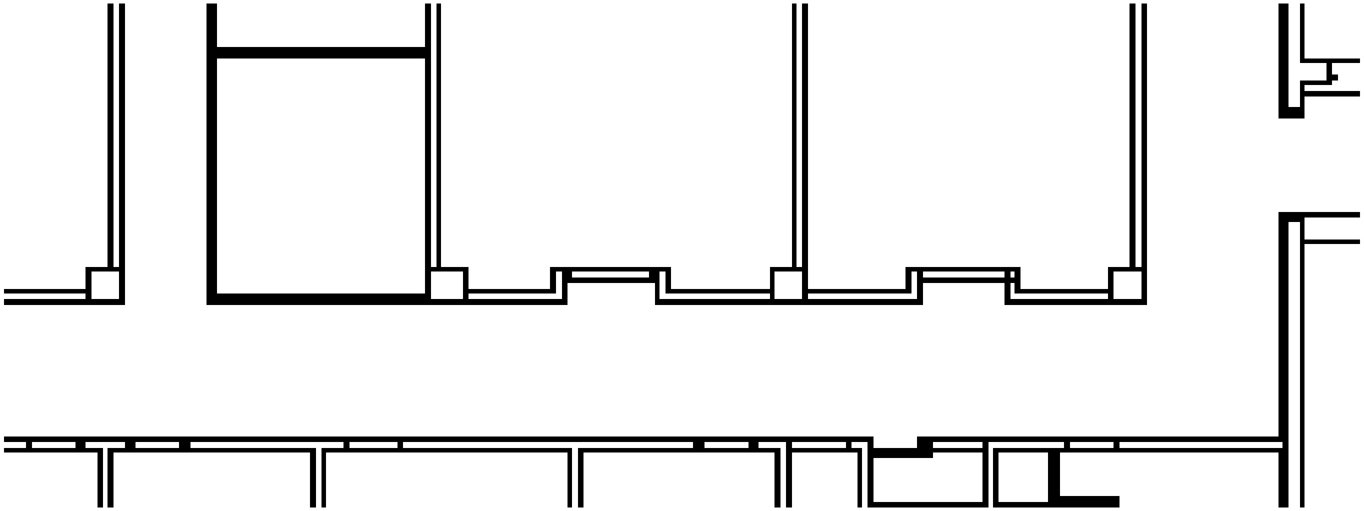


Feature-maps tied to meetpoint locations

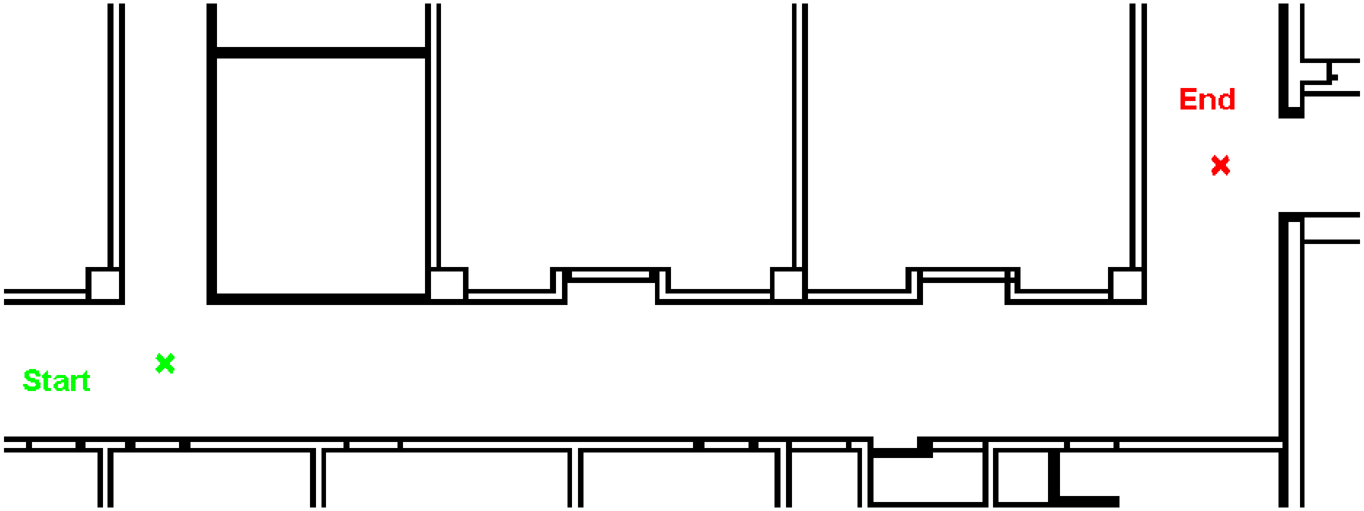
Local Edge-Map Example



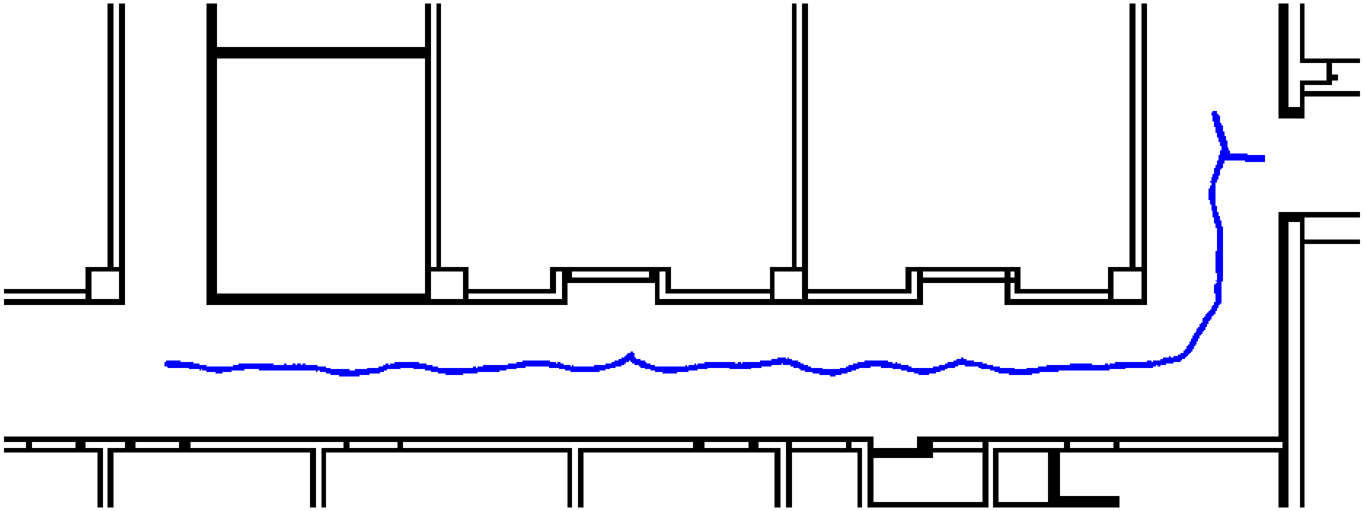
Local Edge-Map Example



Local Edge-Map Example

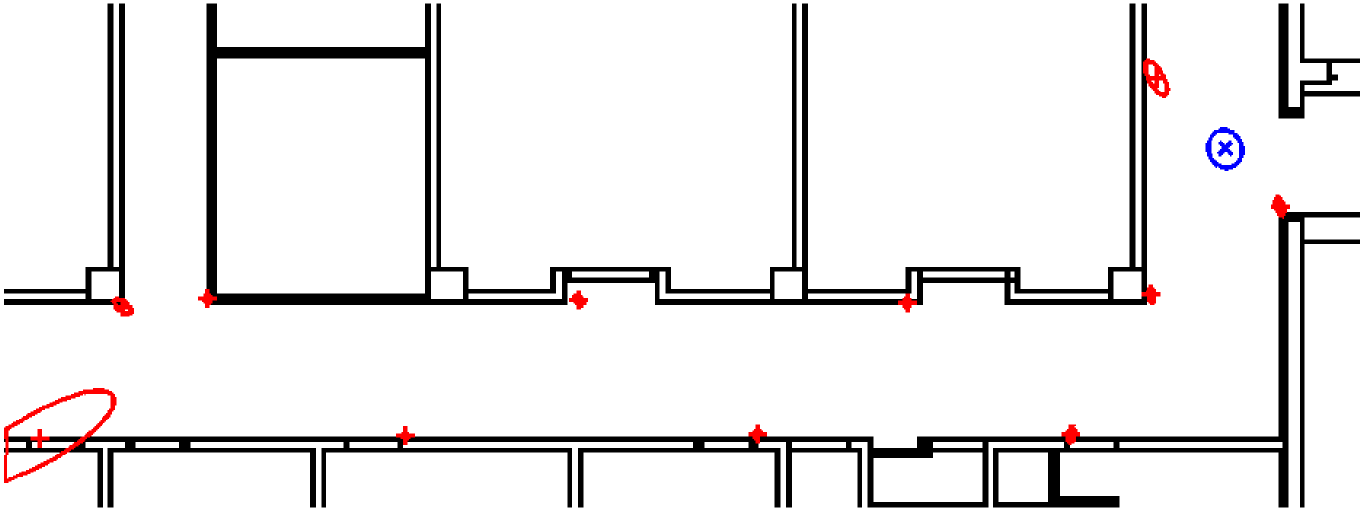


Local Edge-Map Example



16-735, Howie Choset, with significant copying from G.D. Hager who loosely based his notes on notes by Nancy Amato

Local Edge-Map Example



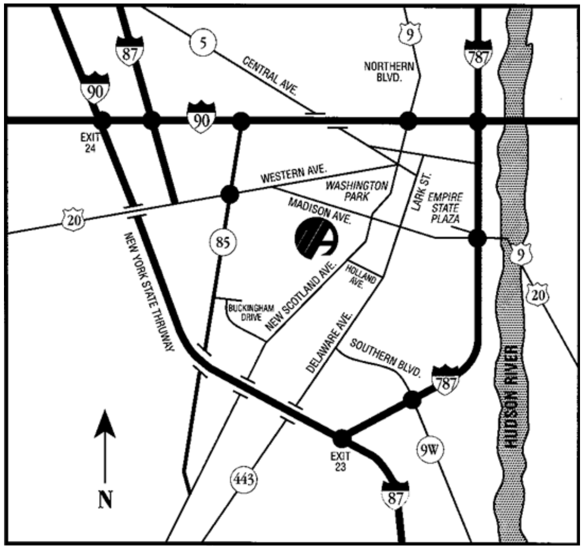
Benefits & Drawbacks of Topological Maps

- Scale
  - dimension
  - geometric size
- Reduce planning problem
  - Graph search
  - Localization along a “line”
- Induces a hierarchy of maps for SLAM
  - T: Topological (Kuipers, Choset)
  - F: Feature-based (Leonard, Durrant-Whyte)
  - L: Local/Pixel-based (Morevac, Elfes, Thrun)
  - D: Dead-reckoning (Borenstein)
- Provides sensor space decomposition useful for control
  - Brooks and other: Behaviors – sense/act
  - Brockett; Manikonda, Krishnaprasad, and Hendler – Motion Description Languages
  - Rizzi, Burridge, Koditschek – Hybrid Controls
  - Kuipers and Choset – Topological Maps

- Cannot position in arbitrary locations
- Fails when environments is not topologically “rich”
  - Hyper-symmetric
  - Large open spaces

Silhouette Method

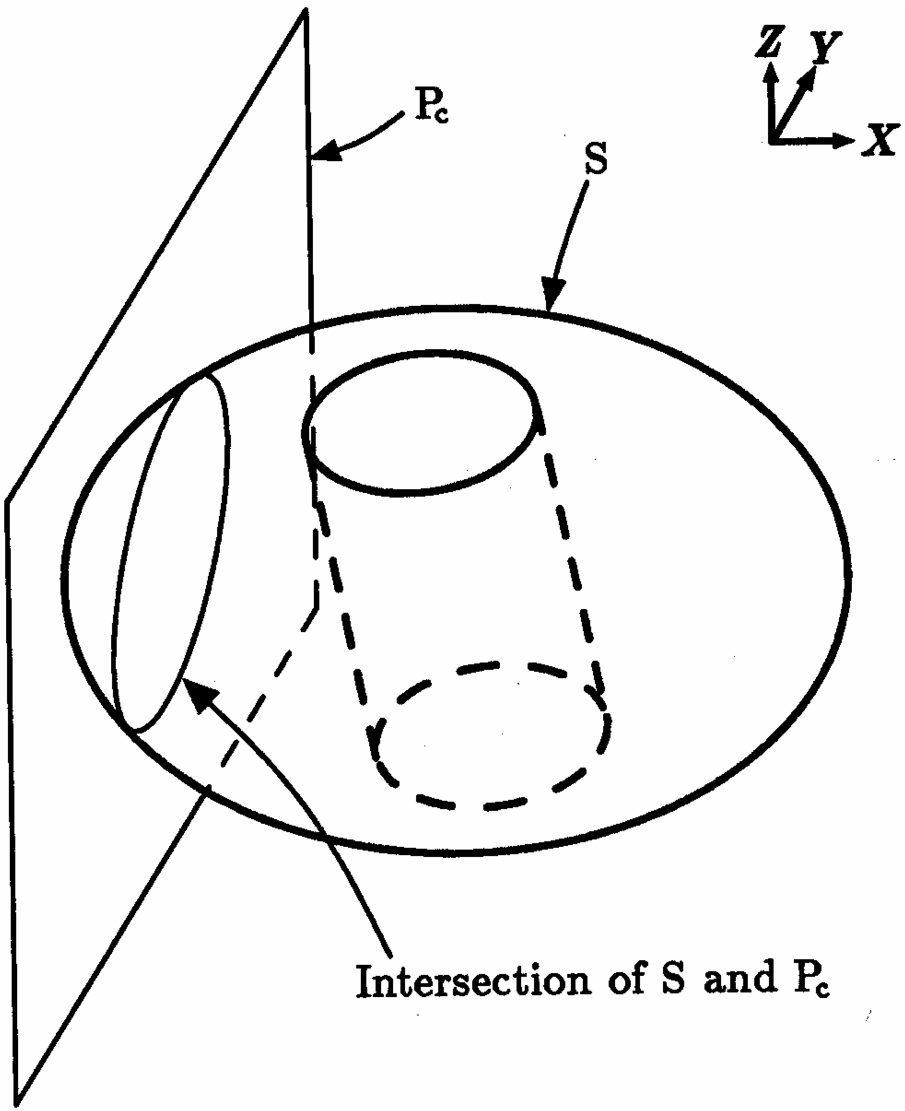
Canny’s Roadmap Algorithm
The Opportunistic Path Planner



16-735, Howie Choset, with significant copying from G.D. Hager who loosely based his notes on notes by Nancy Amato

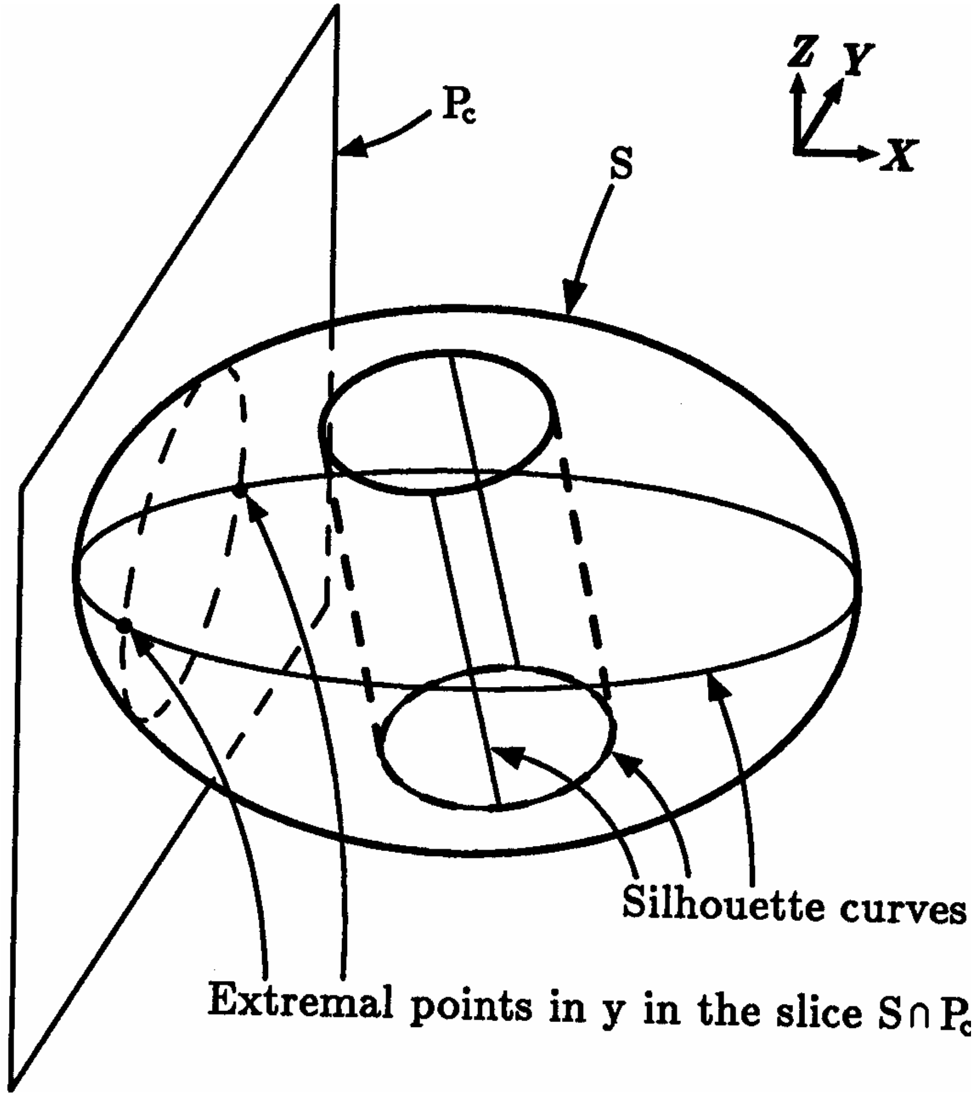
Illustrative Example (1)

Let *S* be the ellipsoid with a through hole. P_c is a hyperplane of codimension 1 (x = c) which will be swept through *S* in the X direction.



Illustrative Example (2)

At each point the slice travels along X we'll find the extrema in S$\cap$P$_c$ in the Y direction. If we trace these out we get **silhouette curves**.



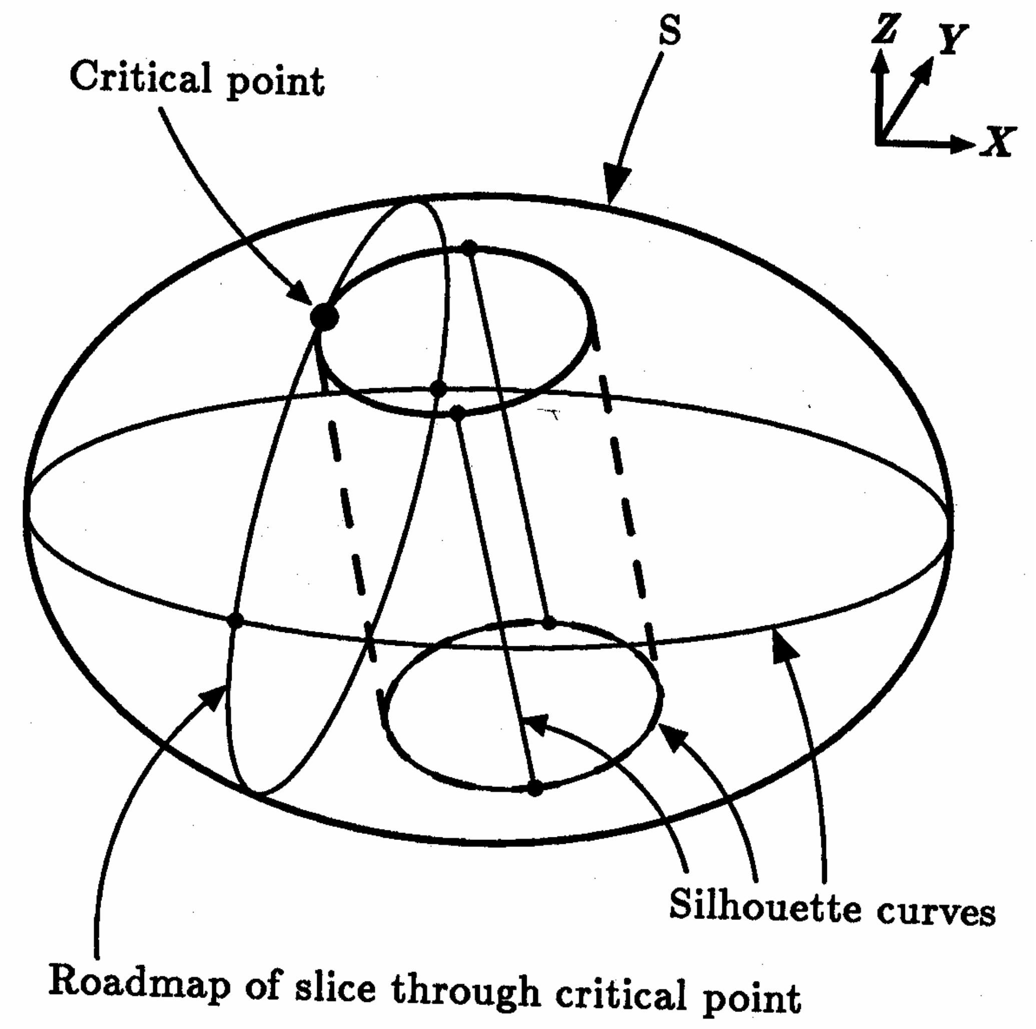
Illustrative Example (3)

Observations:

- The silhouette curves are one-dimensional.
- This is not a roadmap, it’s not connected.
- There are points where extrema disappear and reappear, these will be called **critical points** and the slices that go through these points are **critical slices**.
- A point on a silhouette curve is a critical point if the tangent to the curve at the point lies in P_c.

Illustrative Example (4)

We'll connect a critical point to the rest of the silhouette curve with a path that lies within $S \cap P_c$. This can be done by running the algorithm recursively. Each time, we increase the codimension of the hyperplane by 1.

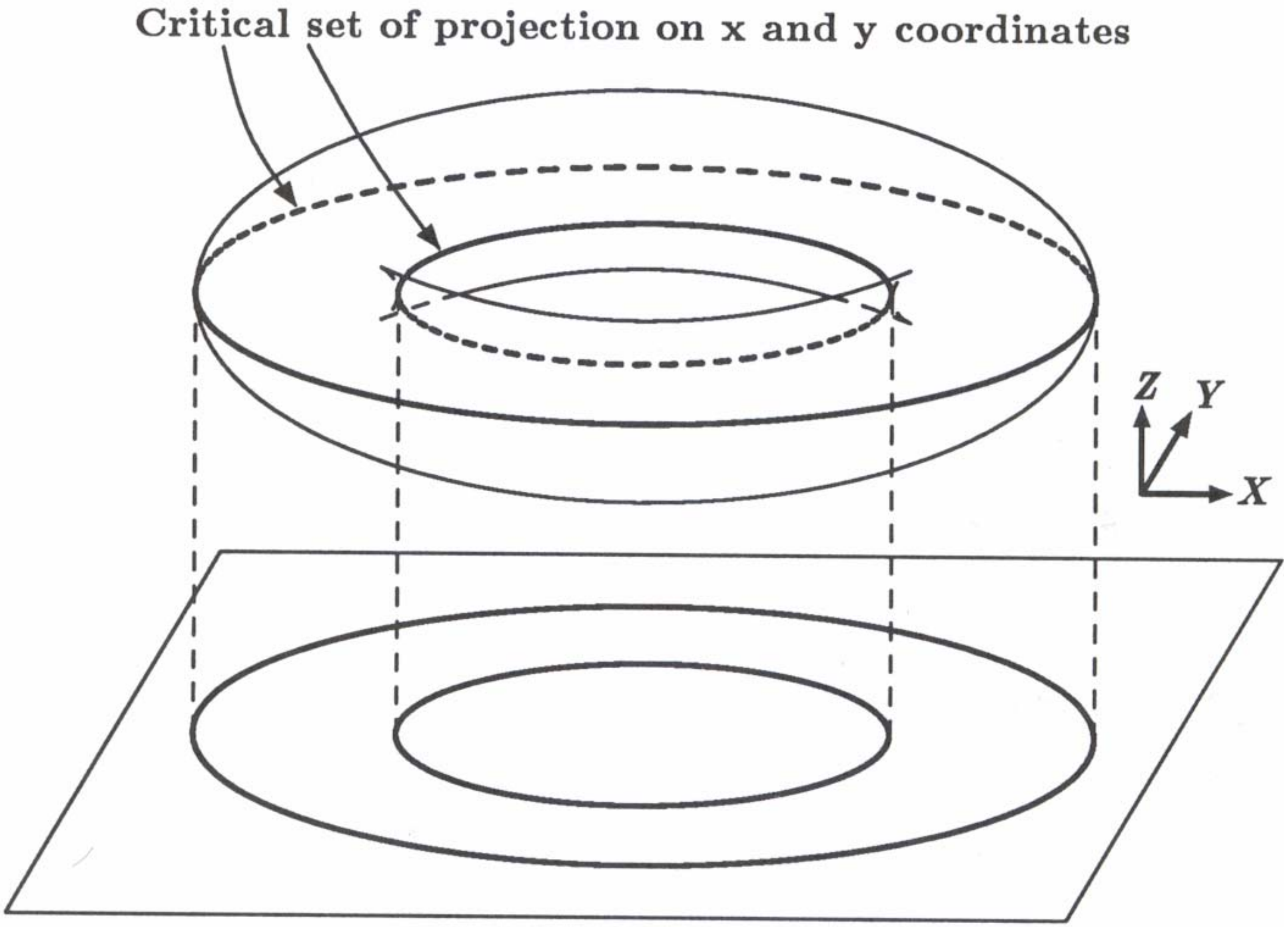


Illustrative Example (5)

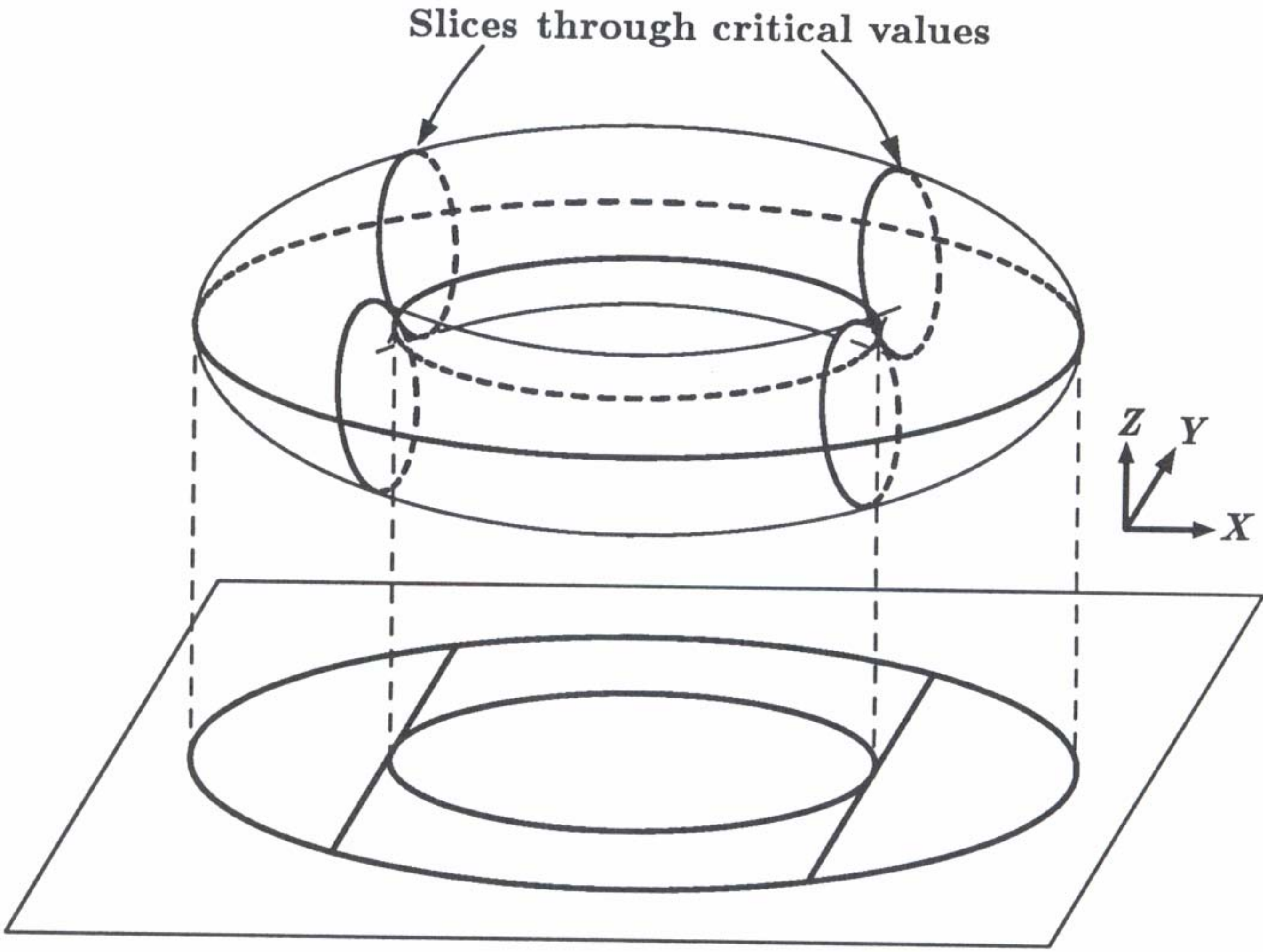
Final points

- The recursion is repeated until there are no more critical points or the critical slice has dimension 1(it is its own roadmap)
- The roadmap is the union of all silhouette curves

Another Example (1)



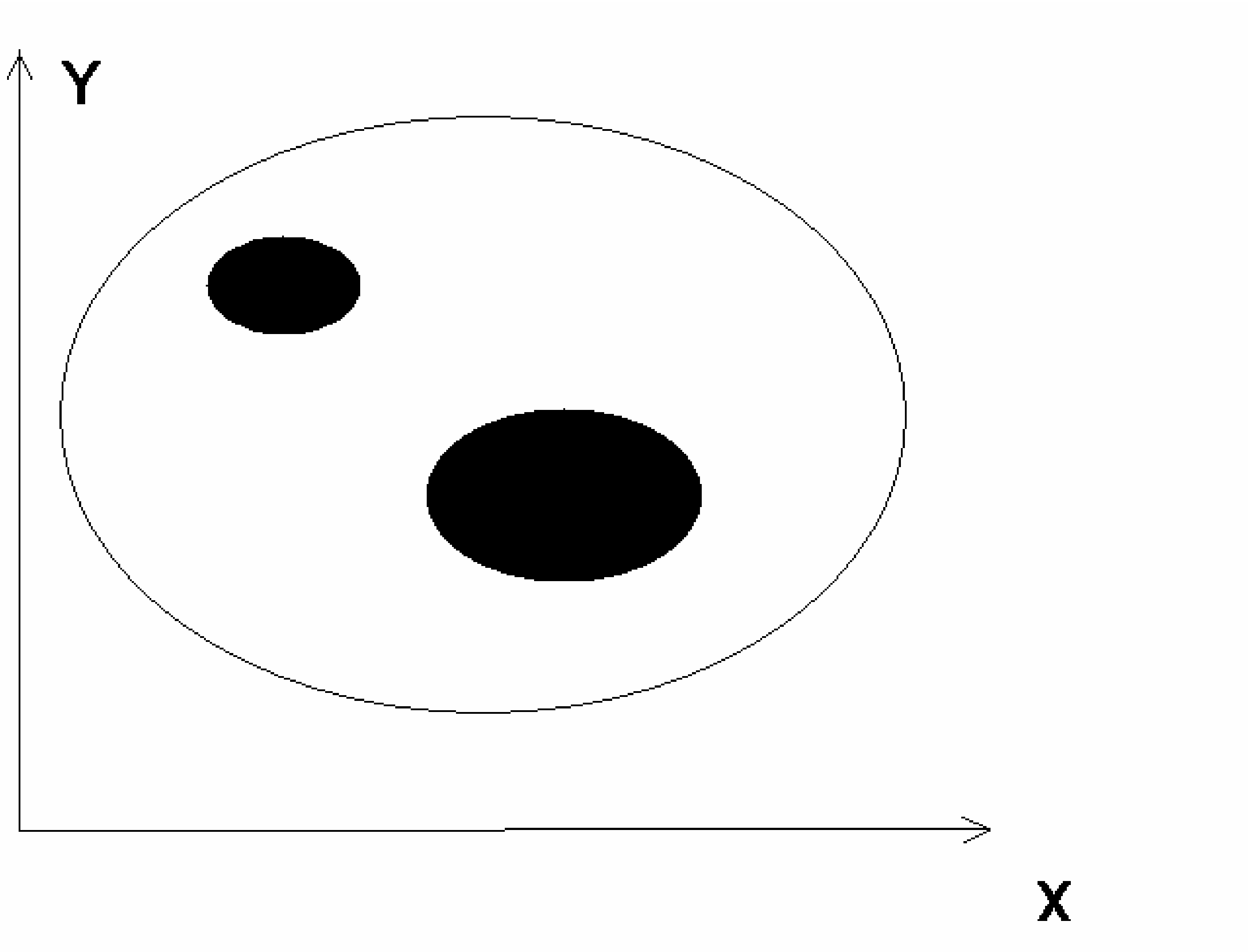
Another Example (2)



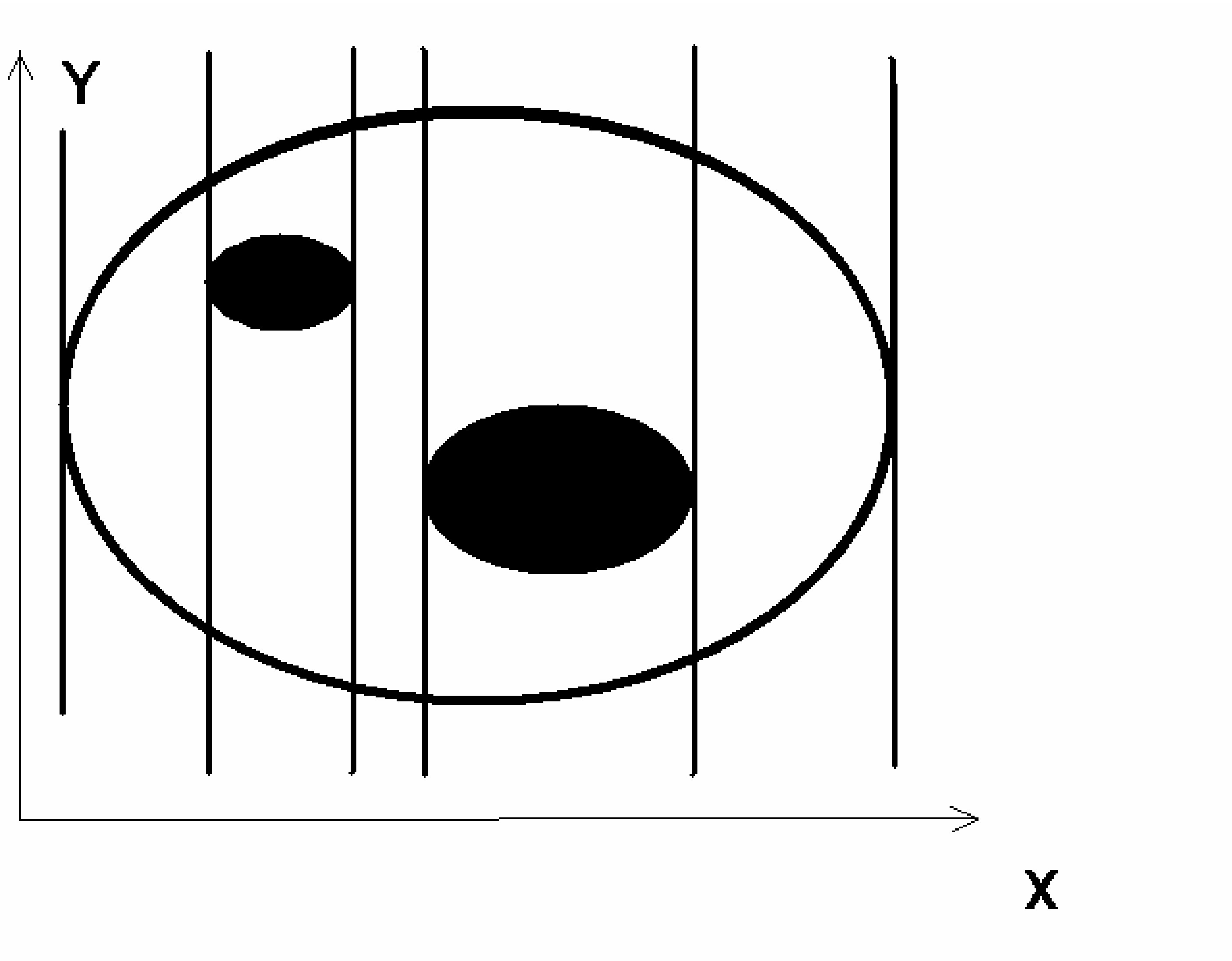
Accessibility and Departability (1)

In order to access and depart the roadmap we treat the slices which contain q_s and q_g as critical slices and run the algorithm the same way.

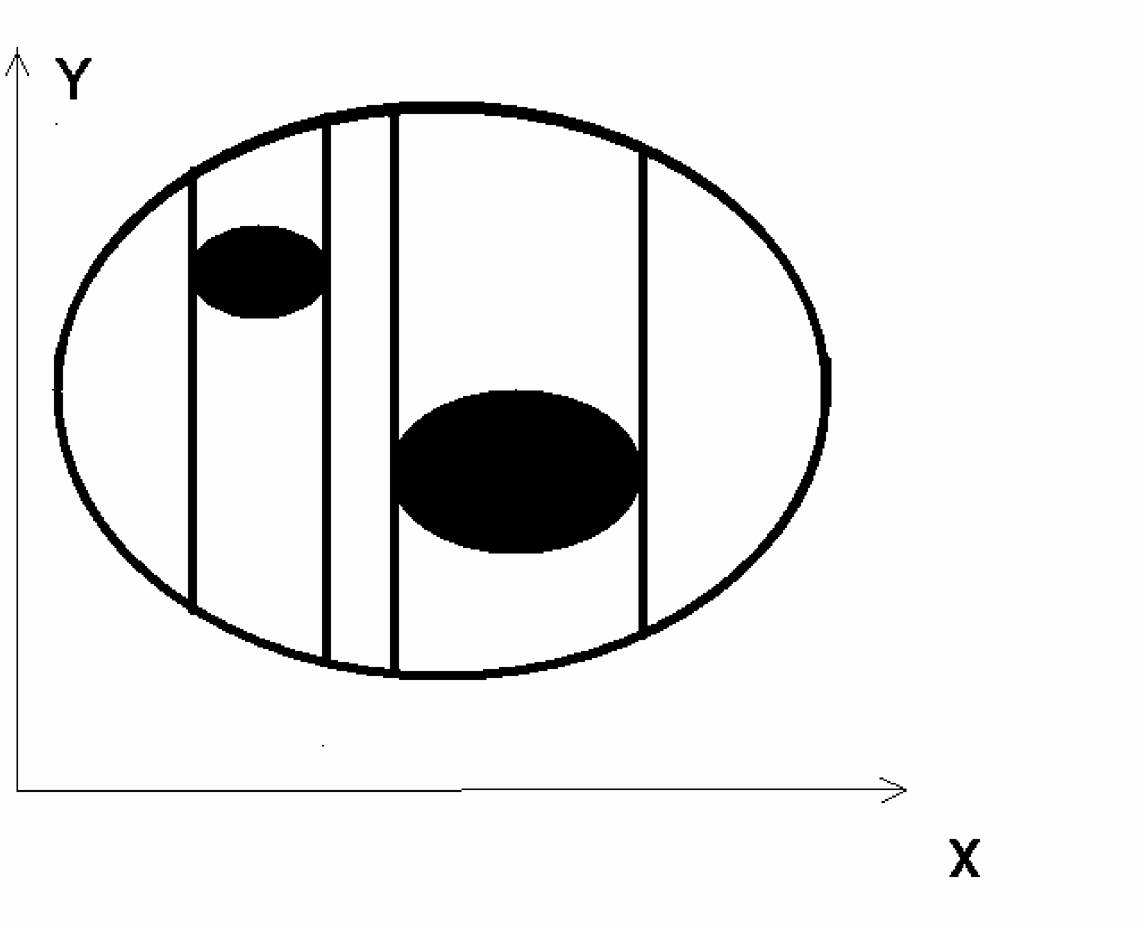
Accessibility and Departability (2)



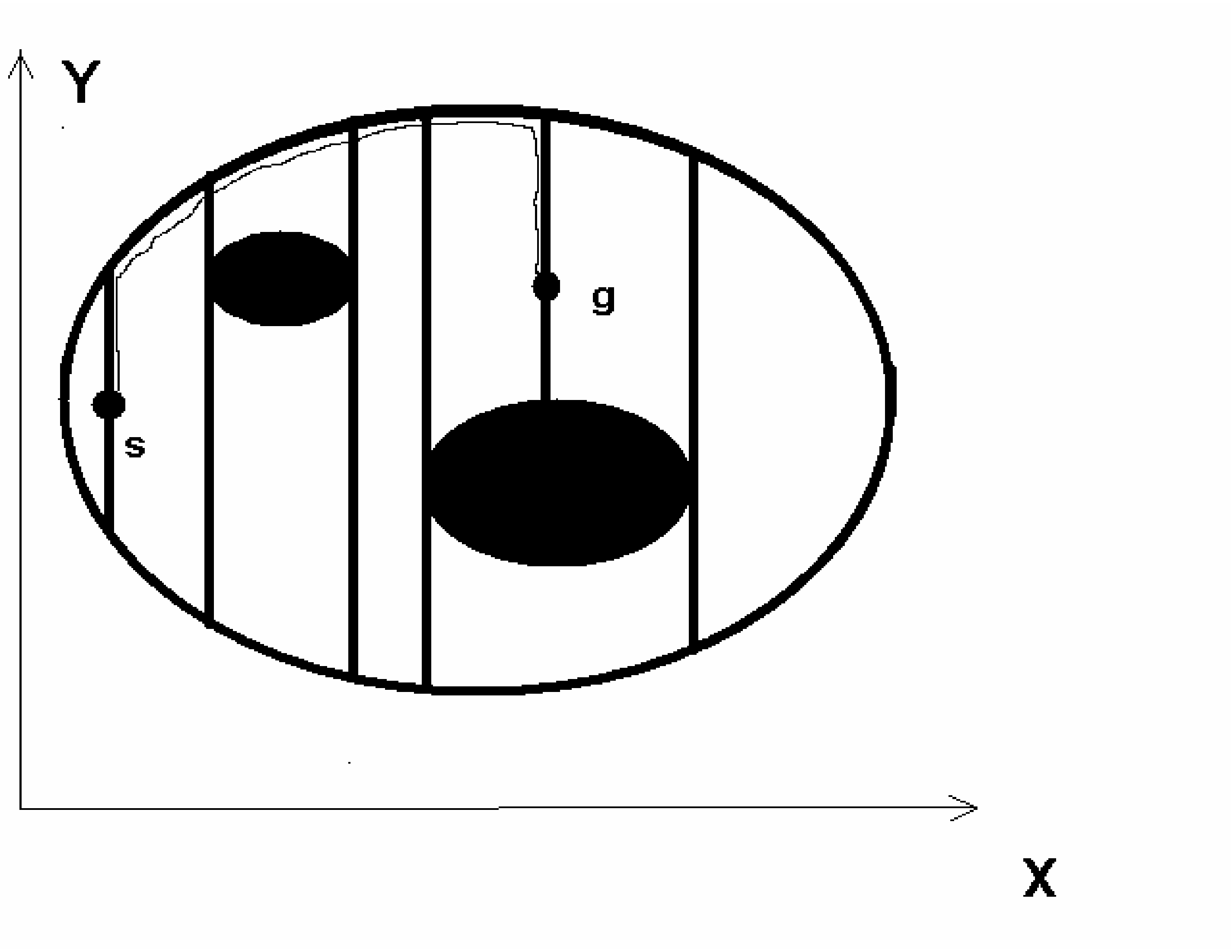
Accessibility and Departability (3)



Accessibility and Departability (4)



Accessibility and Departability (5)



looselv based his notes on notes bv Nancv Amato

Building the Roadmap

Given that the algorithm is now clear conceptually, let’s establish the mathematical machinery to actually construct the roadmap. We must define

- The sets
- The slices
- How to find extrema
- How to find critical points

The Sets

The S which this algorithm deals with are ***semi-algebraic sets*** that are closed and compact.

Def: A *semi-algebraic set* $S \subseteq \Re^r$ defined by the polynomials $F_1,..,F_n \in Q_r$ is a set derived from the sets

$$S_i = \{x \in \Re^r \mid F_i(x) > 0\}$$

by finite union, intersection and complement.

Ex: $(x^2+y^2 \leq 1) \wedge (z \leq 1) \wedge (z \geq -1)$

The Slices

Given that the algorithm is now clear conceptually, let’s establish the mathematical machinery to actually construct the roadmap.

The slices are the intersection of a hyperplane and *S*

$$S_c = S \cap P_c = \{x \in S : \pi_1 = c\}$$

$$\bigcup_c S_c = S$$

where π_1 is the projection on to the first coordinate

$$\pi_k(x_1, x_2, ..., x_n) = x_k$$

How To Find Extrema (1)

When constructing the silhouette curves, we look for extrema of $\pi_2|S_c$, the extrema of the projection of S_c in a second direction.

In order to find the extrema on a manifold we will refer to the **Lagrange Multiplier Theorem**.

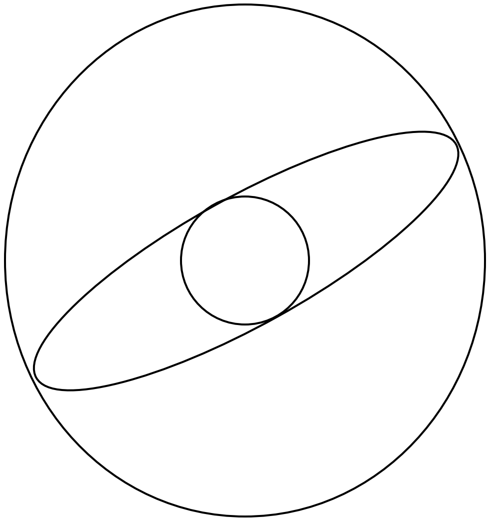
How To Find Extrema (2)

Lagrange Multiplier Theorem:

Let S be an n-surface in $\Re^{n+1}$, $S=f^{-1}(c)$ where f:$U\rightarrow\Re$ is such that $\nabla f(q)\neq 0\ \forall q\in S$.

Suppose h:$U\rightarrow\Re$ is a smooth function and $p\in S$ is a extremum point of h on S.

Then $\exists\lambda\in\Re$ s.t. $\nabla h(p)= \lambda\nabla\ f(p)$ (they are parallel)



How To Find Extrema

Example:

Consider $S=f^{-1}(0)$ where $f=x^2+y^2+z^2-1$ (a solid unit sphere). Extrema of $h=\pi_1(x,y,z)=(x)$.

$$d(f,h)=\begin{bmatrix} 2x & 2y & 2z \\ 1 & 0 & 0 \\ & & \end{bmatrix}$$

y = z = 0 (y-z plane) and only points on sphere is x = 1, x = -1, left most and right most

How To Find Extrema (3)

Canny's Generalization of the Lagrange Multiplier Theorem:

Suppose that *U* is an open subset of the kernel of some map f:$\Re^r \rightarrow \Re^n$, and let *f* be transversal to {0}. Let *g*:$\Re^r \rightarrow \Re^m$ be a map, then *x*$\in$*U* is an extremum of *g*|*U* iff the following matrix is not full rank.

$$d(f,g)_x = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \cdots & \frac{\partial f_1}{\partial xr}(x) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(x) & & \frac{\partial f_n}{\partial x_r}(x) \\ \frac{\partial g_1}{\partial x_1}(x) & \cdots & \frac{\partial g_1}{\partial x_r}(x) \\ \vdots & & \vdots \\ \frac{\partial g_m}{\partial x_1}(x) & \cdots & \frac{\partial g_m}{\partial x_r}(x) \end{bmatrix}$$

How To Find Extrema (4)

Canny's Slice Lemma:

The set of critical points of $\pi_{12}|S$, $\Sigma(\pi_{12}|S)$,

is the union of the critical points of $\pi_2|S_c$ where we sweep in the 1 direction.

$$\Sigma(\pi_{12}|_S) = \bigcup_{\lambda} \Sigma(\pi_2|_{\pi_1^{-1}(\lambda)}).$$

How To Find Extrema (5)

Example:

Consider $S=f^{-1}(0)$ where $f=x^2+y^2+z^2-1$ (a solid unit sphere). If we sweep in the *x* direction and extremize in the *y* direction $h=\pi_{12}(x,y,z)=(x,y)$.

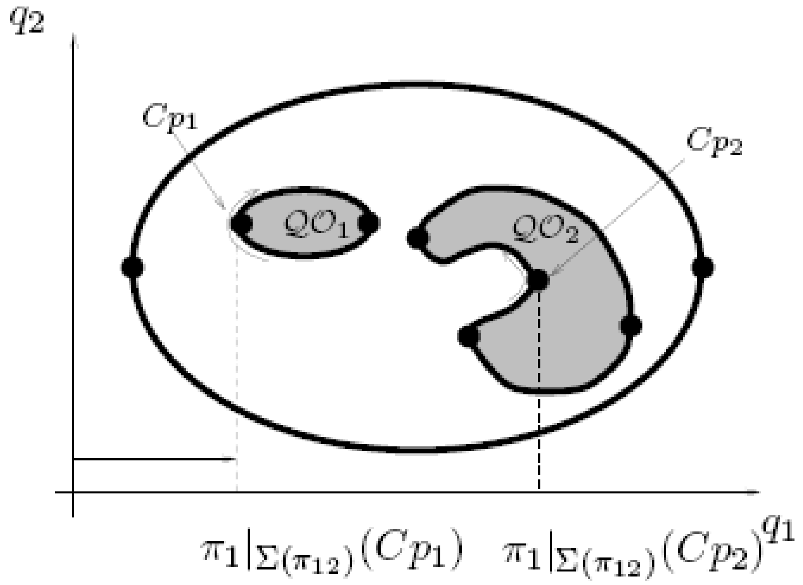
$$d(f,h)=\begin{bmatrix} 2x & 2y & 2z \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

So the silhouette curve is the unit circle on the *x*-*y* plane

Finding Critical Points

The critical points which denote changes in connectivity of the silhouette curves also follow from Canny's Generalization. They are the extrema of the projection on to the sweeping direction of the silhouette curves. Simply

$$\Sigma(\pi_{1|\Sigma(\pi_{12})})$$



q_2
Cp_1
Cp_2
QO_1
QO_2
q_1
$\pi_1|_{\Sigma(\pi_{12})}(Cp_1)$
$\pi_1|_{\Sigma(\pi_{12})}(Cp_2)$

$$\pi_1(q)$$

Can be viewed as the distance to the y axis from a point

Critical point is where roadmap tangent is parallel to slice

Finding Critical Points in Higher Dims

$D(f, \pi)(q)$ loses rank.

Define roadmap as the pre-image of f, but cannot do so. Df, however is a m-1 x m matrix.

This matrix forms the top m-1 rows of $D(f, \pi)$

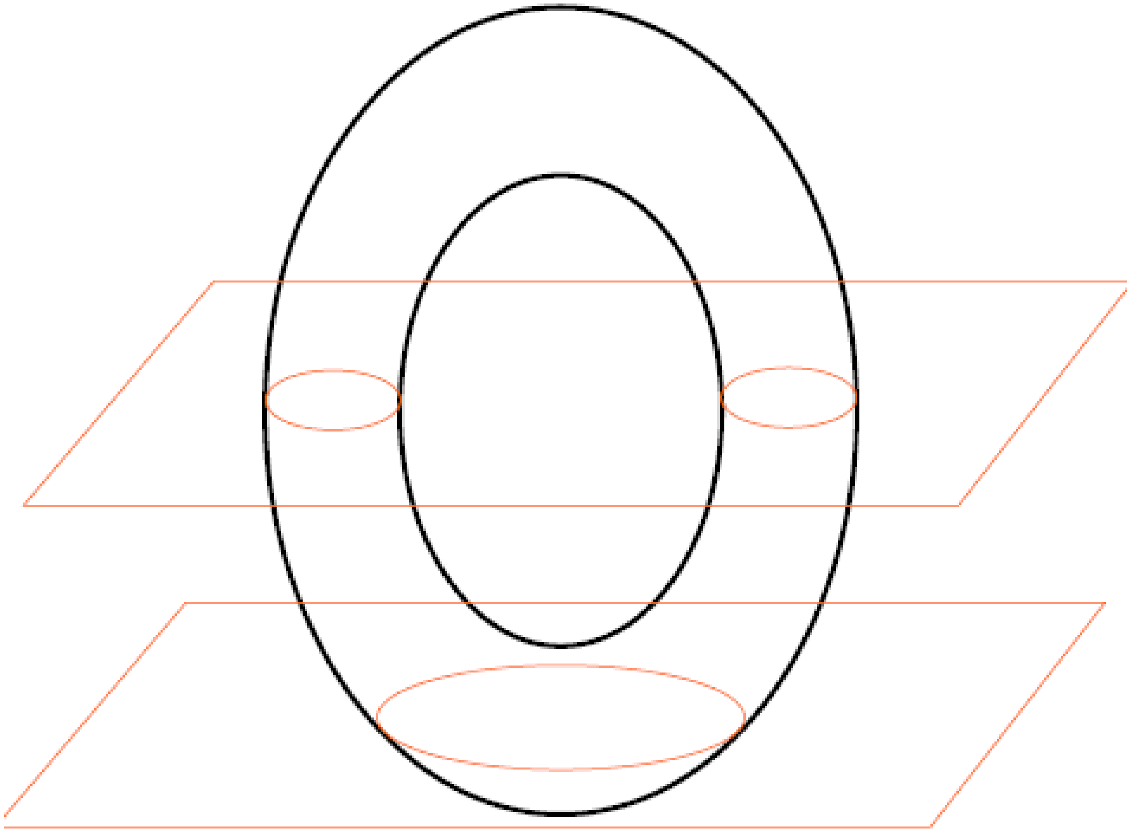
Null of Df is tangent to roadmap, so m-1 row vectors of Df form a plane orthogonal to roadmap tangent $T^\perp$

Slice function π_1 has gradient $[1, 0, \ldots, 0]^T$ which forms the bottom row of $D(f, \pi)$

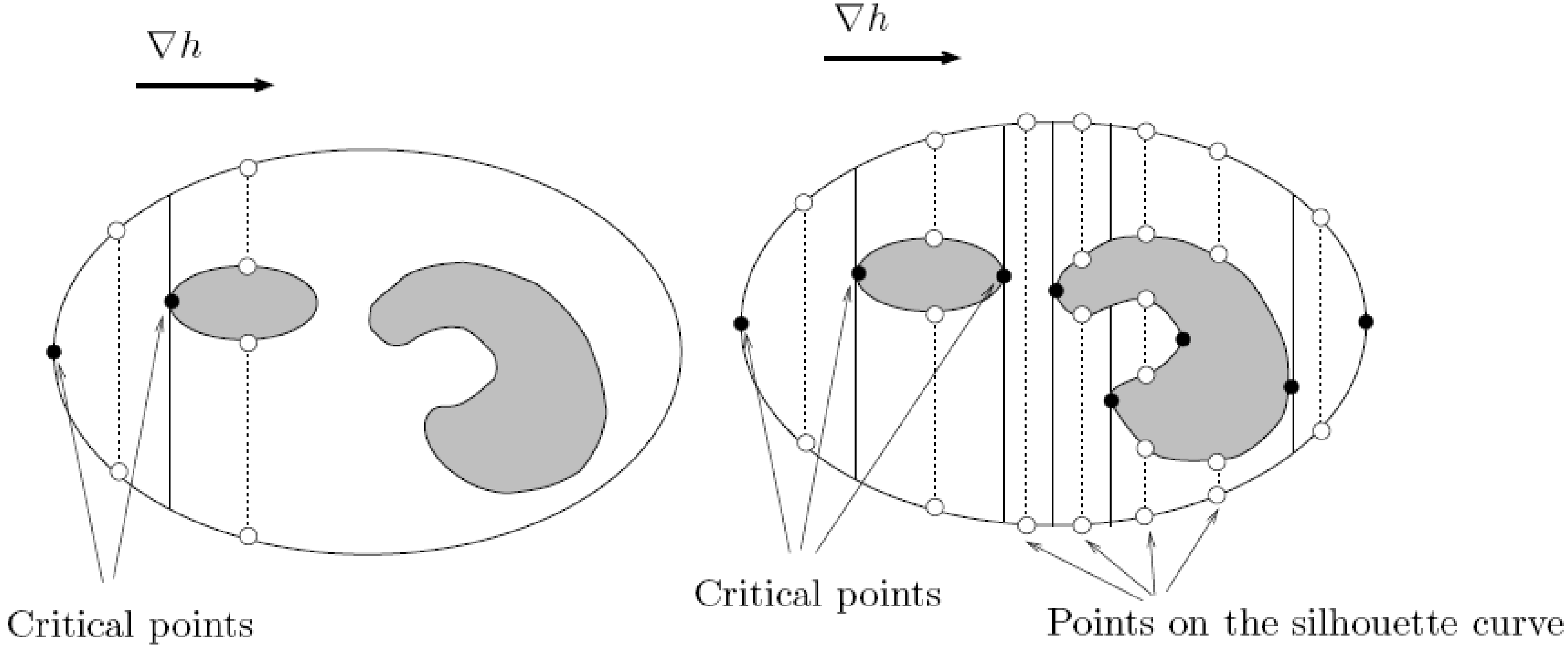
When roadmap tangent lies in slice plane, this means that and slice plane are orthogonal to each other $T^\perp$

$\nabla \pi_1(q)$ lies in $T^\perp$ $D(f, \pi)$ looses rank

Connectivity change at Critical Points

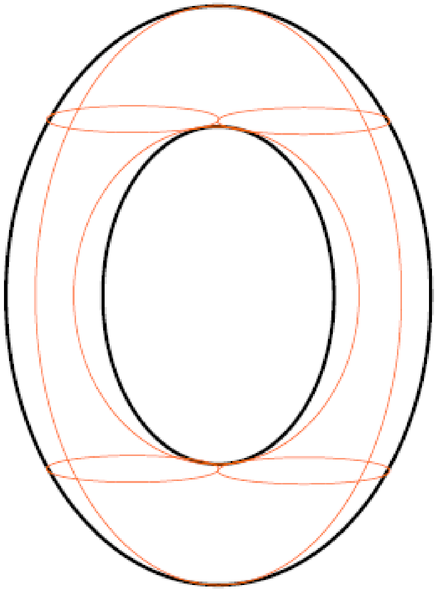


Between Critical Points



Building the Roadmap (Conclusion)

- We can now find the extrema necessary to build the silhouette curves
- We can find the critical points where linking is necessary
- We can run the algorithm recursively to construct the whole roadmap



Proof of Connectivity (1)

First, let $S|\leq c$ denote the set $S\cap(x\leq c)$. We are claiming that $R(S)\ |\leq c$ is connected within each component of $S|\leq c$.

Base case: c is small enough such that $S|\leq c$ is empty and the claim is vacuously true

Induction hypothesis: The claim is true for some $c=c_0$

Proof of Connectivity (2)

Inductive step:

It remains true as c is increased until we come upon another critical value c_1 associated with a critical point p.

If the algorithm works on slice $S \cap Pc_1$, then if a new component of silhouette appears or if several components of S come together at p, they will be joined recursively by the algorithm. Therefore the claim is true for c_1 and all other critical values.

The Opportunistic Path Planner

The Opportunistic Path Planner is similar to Canny’s Roadmap but differs in the following ways

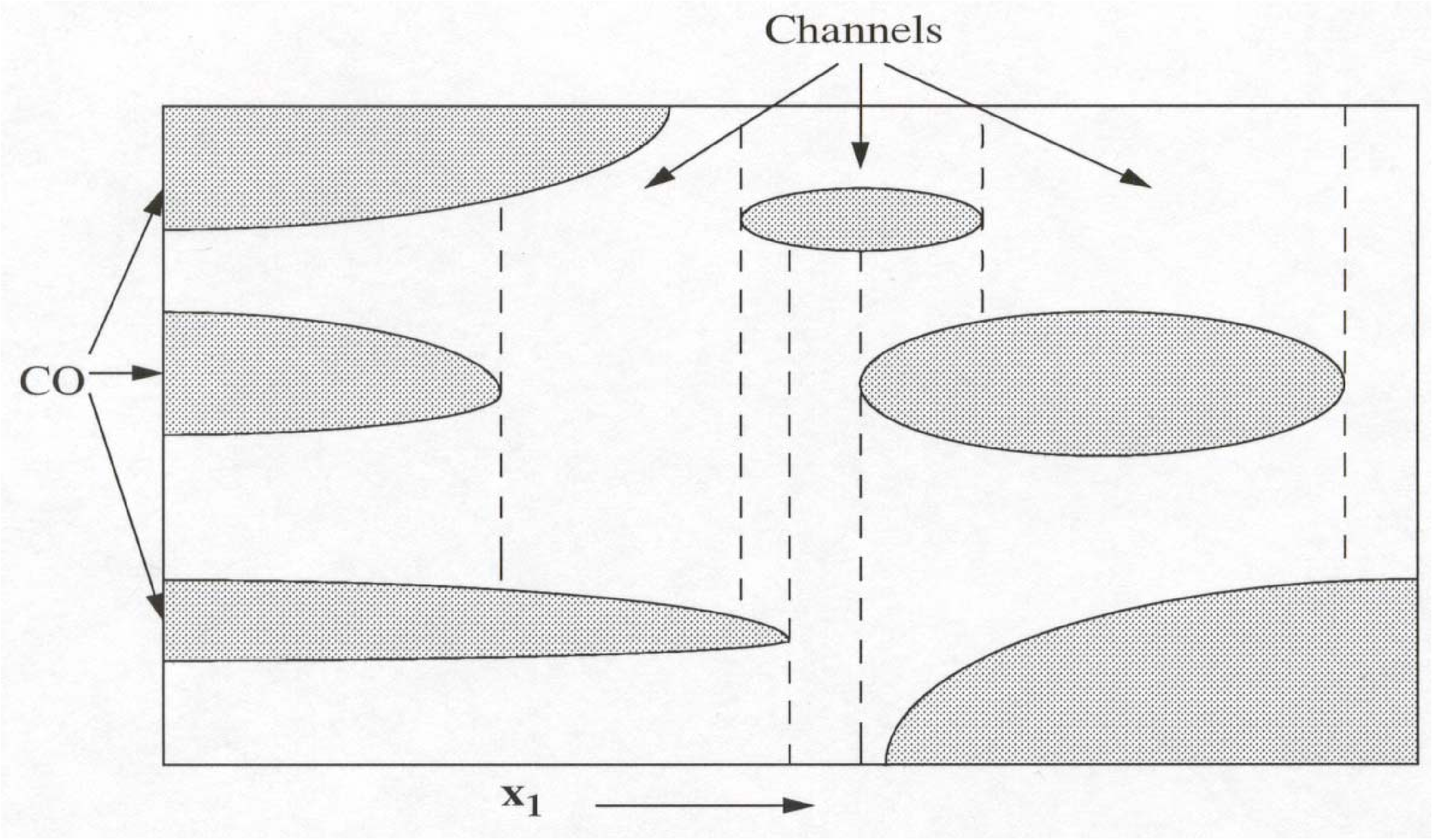
- Silhouette curves are now called *freeways* and are constructed slightly differently
- Linking curves are now called *bridges*
- It does not always construct the whole roadmap
- The algorithm is not recursive

Channels (1)

Def: A *channel slice* is a slice at which the connectivity of the intersection with the sweeping hyperplane and the freespace changes.

Def: A *channel* is a subset of the freespace which is bounded by channel slices and configuration space obstacles

Channels (2)



Interesting Critical Points and Inflection Points

Def: An *interesting critical point* is a critical point that corresponds to the joining or splitting of the intersection of the sweeping hyperplane and the freespace

Def: An *inflection point* is a point where the tangent to the freeway curve becomes orthogonal to the sweep direction

Freeways

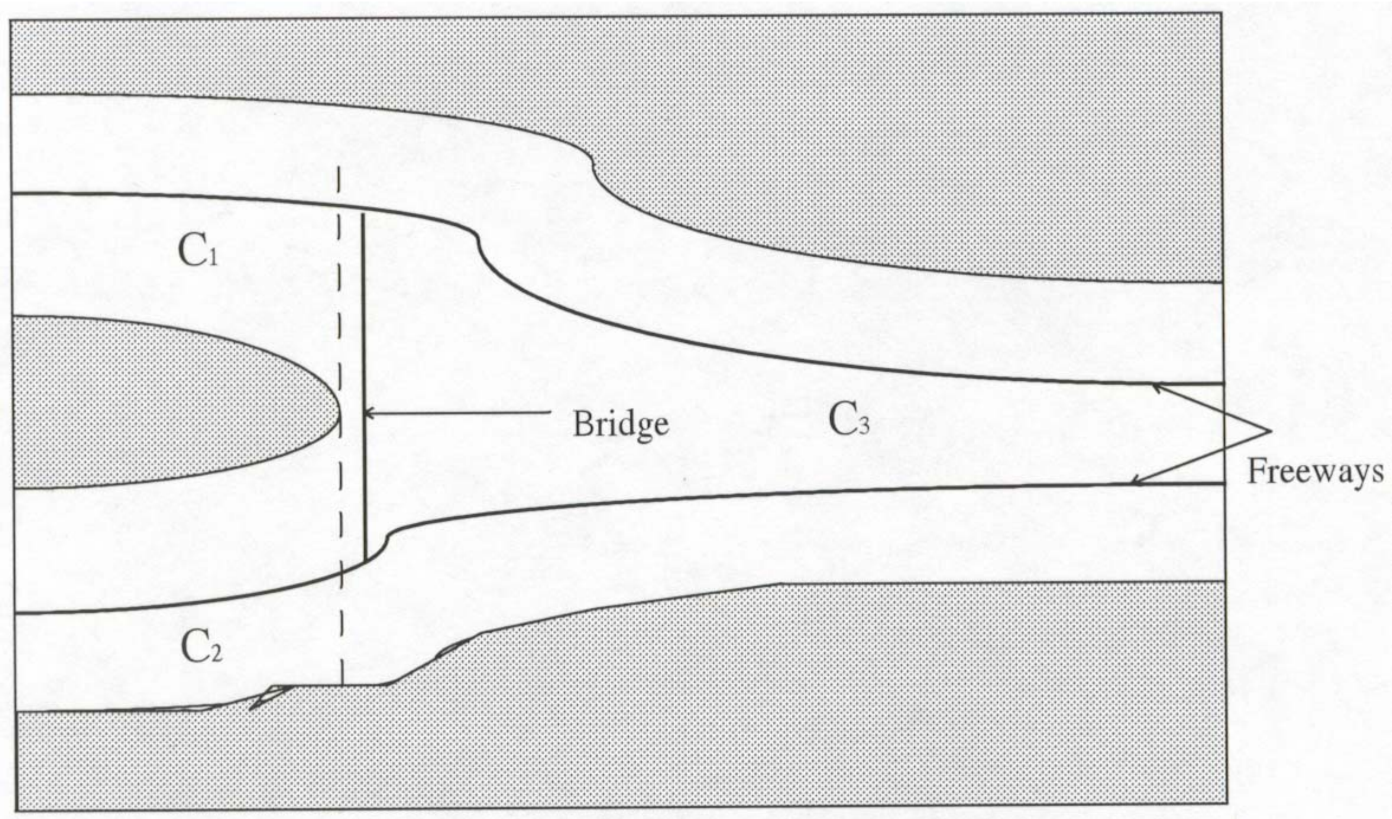
Freeways are defined by the following artificial potential field which induces an artificial repulsion from the surface of obstacles

$$U_{art}(x) = D(x)$$

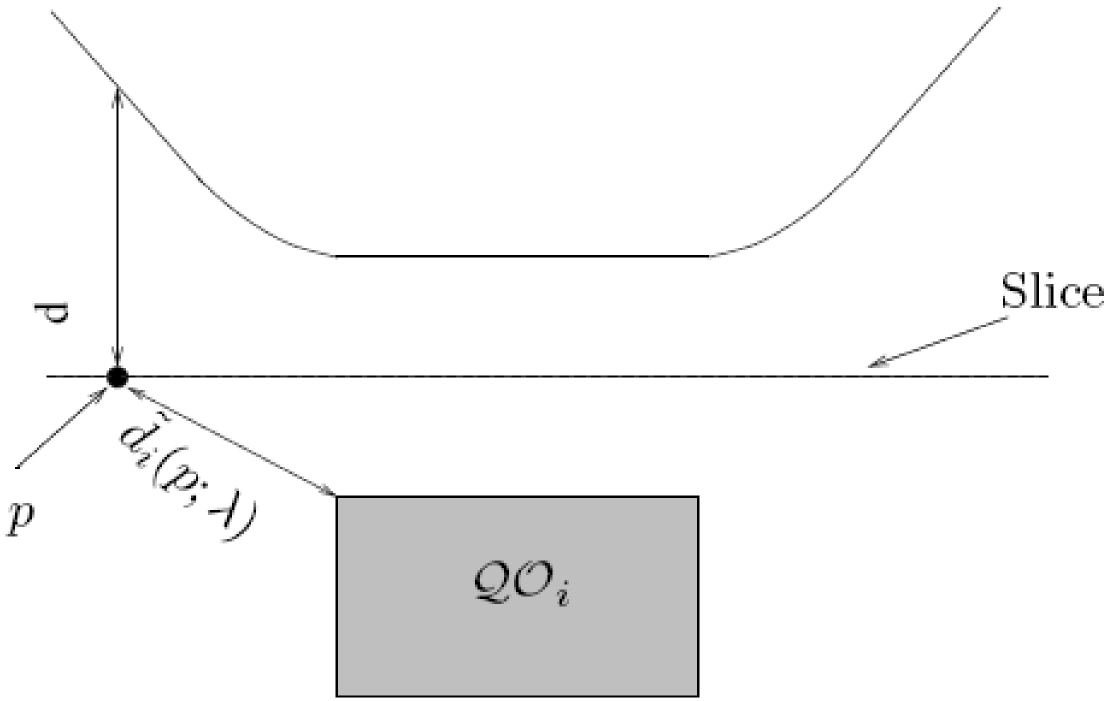
A freeway is the locus of the maxima of $U_{art}(x)$ as you sweep through the configuration space

Bridges

Def: A *bridge* is a one-dimensional set which links freeways from channels that have just joined or are about to split (as you sweep across)



Freeway Tracing



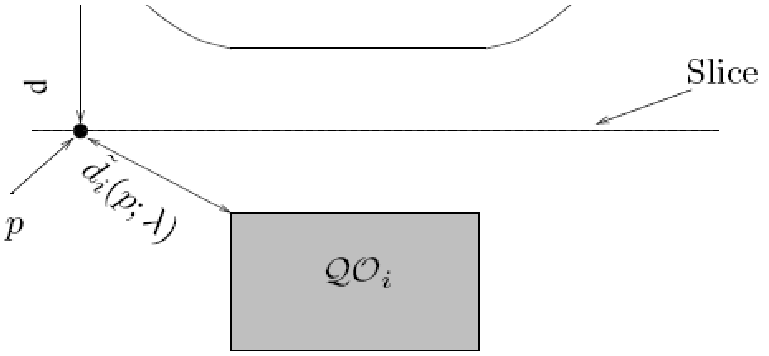
Freeway Tracing

Freeway tracing is done by tracking the locus of the maxima of the artificial potential field and terminates when

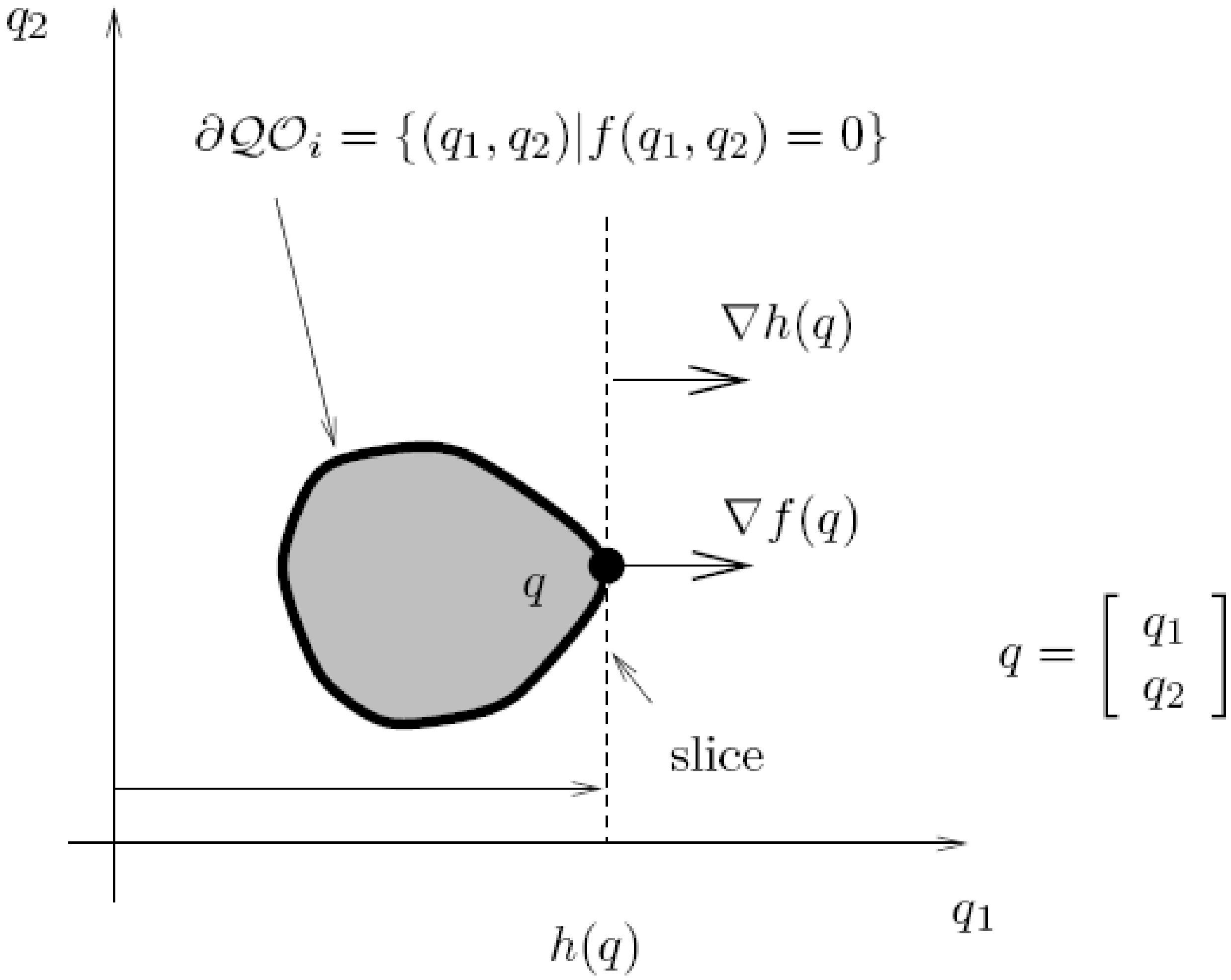
(1) The freeway runs into an inflection point where you create a bridge

(2) The freeway runs into an obstacle where it ends

$$\begin{aligned} \partial D(q^*) &= \mathrm{Co}\{\nabla d_i(q^*) \mid i \in Z(q^*)\} \\ &= \textstyle\sum_{i \in Z(q^*)} \mu_i \nabla d_i(q^*) \text{ where } \textstyle\sum_{i \in Z(q^*)} \mu_i = 1 \text{ and } \mu_i > 0, \end{aligned}$$



Also create bridges at interesting critical points



q_2
$\partial \mathcal{QO}_i = \{(q_1, q_2) | f(q_1, q_2) = 0\}$
$\nabla h(q)$
$\nabla f(q)$
q
slice
$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$
q_1
$h(q)$

Accessibility and Departability

The roadmap is accessed and departed by connecting q_s and q_g to a local maximum on the slice which they reside (which is part of a freeway).

This is referred to as *hill-climbing* and is the same procedure we use when creating bridges except in the case of bridges we hill-climb in two directions.

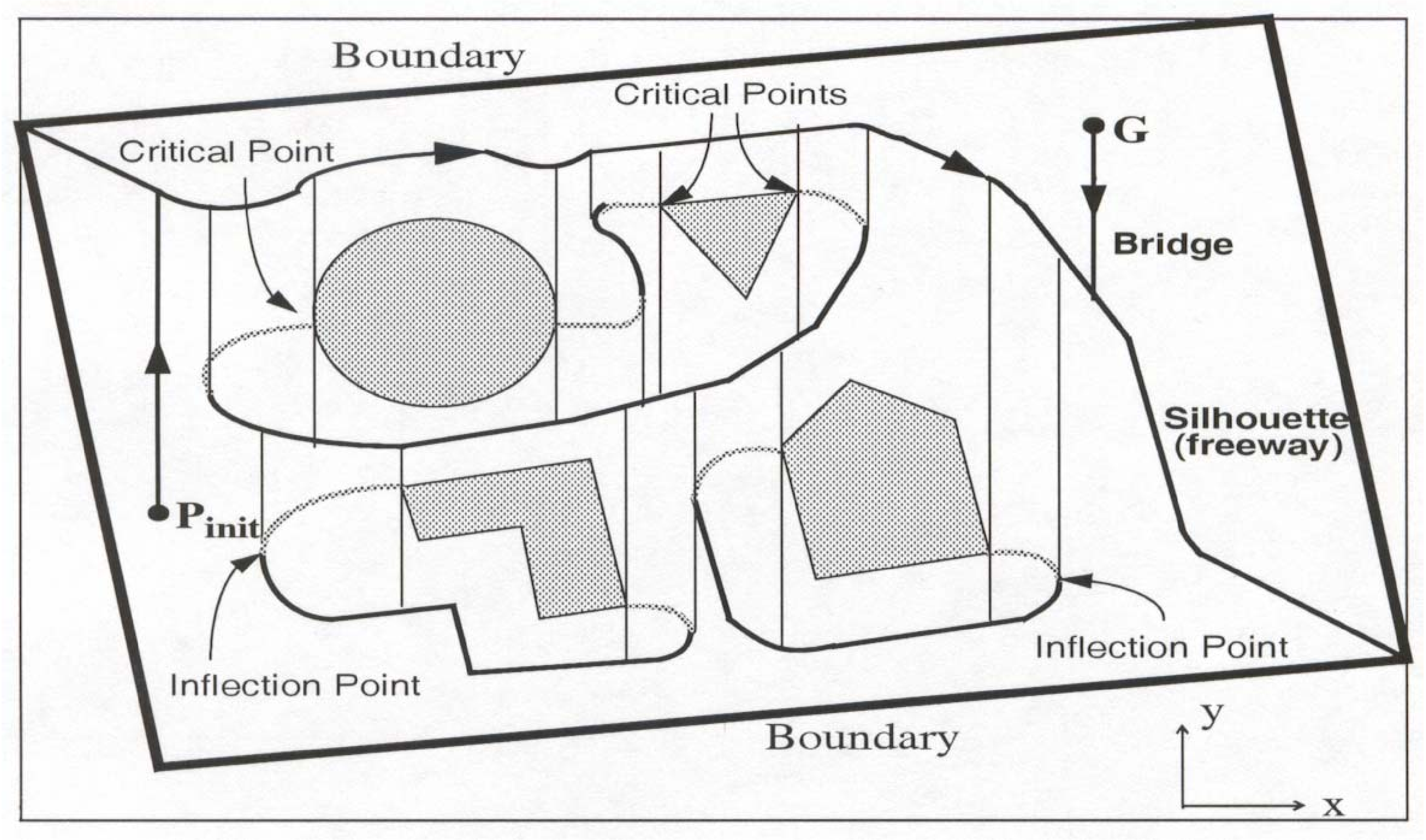
Building the Roadmap

(1) Hill-climb from both q_s and q_g. Then trace freeway curves from both start and goal

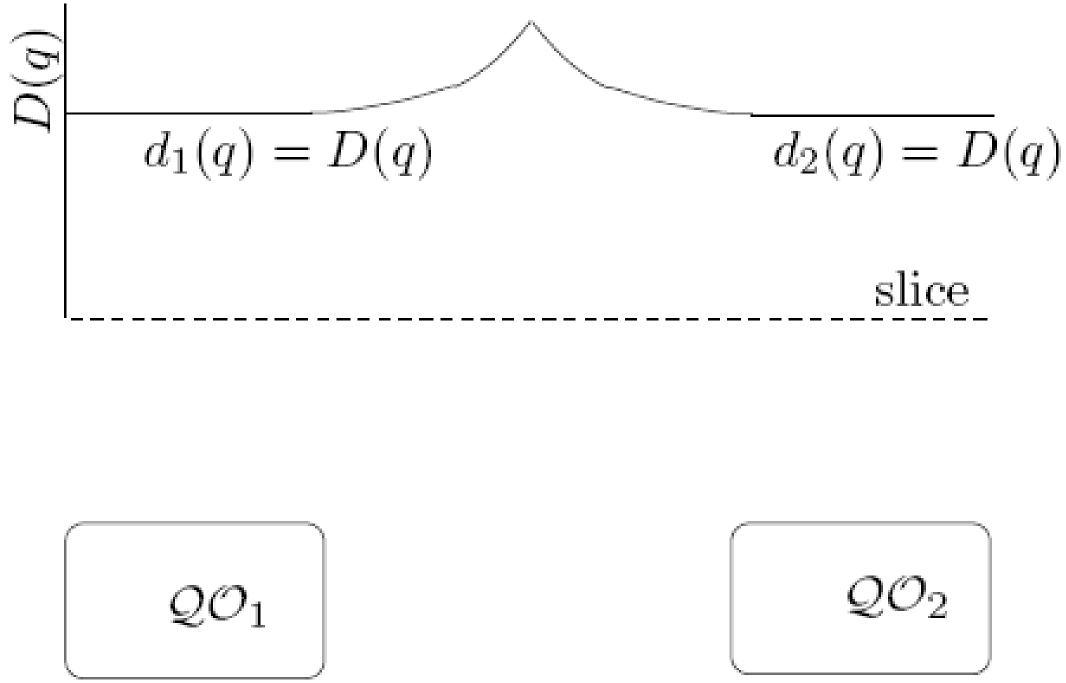
(2) If the curves leading from start and goal are not connected enumerate a split point or join point and add a bridge curve near the point. Else stop.

(3) Find all points on the bridge curve that lie on other freeways and trace from these freeways. Go to step 2.

Example


Boundary
Critical Points
Critical Point
G
Bridge
Silhouette (freeway)
P_init
Inflection Point
Inflection Point
Boundary
y
x

Nonsmooth Analysis



$$\begin{aligned}\partial D(q^*) &= \mathrm{Co}\{\nabla d_i(q^*) \mid i \in Z(q^*)\} \\ &= \textstyle\sum_{i \in Z(q^*)} \mu_i \nabla d_i(q^*) \text{ where } \textstyle\sum_{i \in Z(q^*)} \mu_i = 1 \text{ and } \mu_i > 0,\end{aligned}$$

Proof of Connectivity (1)

Let A be the set of all x_1-coordinates (sweeping direction) of critical points which are order in ascending order $A = \{a_1,a_2,\ldots..,a_m\}$

Base case: $x_1=a_1$, $S|\leq a_1$ should consist of a single point which will be part of the roadmap

Inductive hypothesis: The roadmap condition is satisfied for $x_1\leq a_{i-1}$

Proof of Connectivity (2)

Inductive step:

It is a fact that you can smoothly deform or retract a manifold or union of manifolds in the absence of critical points. So $S|<a_{\mathrm{i}}$ can be smoothly retracted on to $S|\leq a_{\mathrm{i-1}}$ because (a_{i-1},a_i) is free of critical points.

Proof of Connectivity (3)

Inductive step (continued):

Also, $R(S)|<a_i$ can be retracted on to $R(S)|\leq a_{i-1}$. This implies that there are no topological changes in $R(S)$ or S on the interval (a_{i-1},a_i) and if $R(S)|\leq a_{i-1}$ satisfies the roadmap condition so does $R(S)|<a_i$

Proof of Connectivity (4)

Inductive step (continued):

Let p_i be the critical point that corresponds to a_i. As x_i increases to a_i the only way connectivity can be lost is if p_i is an inflection point or a join point. Both of these situations will be handled by the application of hill-climbing which will create linking curves. Therefore the roadmap condition holds for $R(S)|\leq a_\mathrm{i}$ and our inductive step is proven.

Assumptions

- Robot is a point
- Workspace contains only convex obstacles
- Non-convex obstacles are modeled as the union of convex obstacles
- Bounded space

Introduction
Distance Function
GVG and Pre-Image Theorem
Numerical Curve Tracing
Definition of HGVG

Summary

- Roadmap methods create a graph of “roads” that will move you through the space; just get on and get off again

  The visibility graph is one method of doing this for polygonal worlds

  Voronoi diagrams are a second form of roadmap

  We will see more graphs in the second half of the semester...