# Algorithms for Sensor-Based Robotics: RoadMap Methods

Computer Science 336
http://www.cs.jhu.edu/~hager/Teaching/cs336

Professor Hager
http://www.cs.jhu.edu/~hager

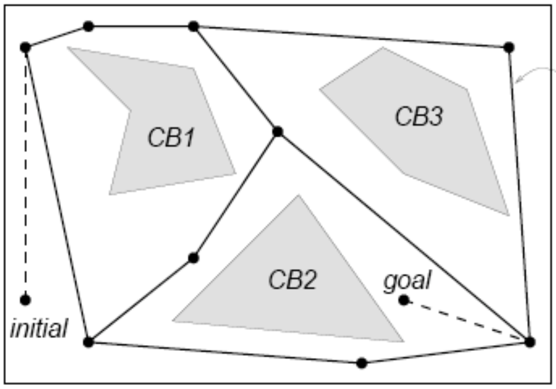
---

# Where Are We Going?

- Chap 2: Bug
  - Complete
  - Non-optimal
  - Planar
- Chap 4: Potential fields
  - Complete in special cases (e.g. nav functions on star worlds)
  - Non-optimal
  - General configuration spaces
  - Scalable to large problems
- Chap 5/6: combinatorial methods
  - Complete
  - Sometimes optimal
  - Sometimes general
  - Intractible in general
- Chap 7: Sampling methods
  - Probabilistically complete
  - Can be made close to optimal
  - General
  - Tractible

# 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''

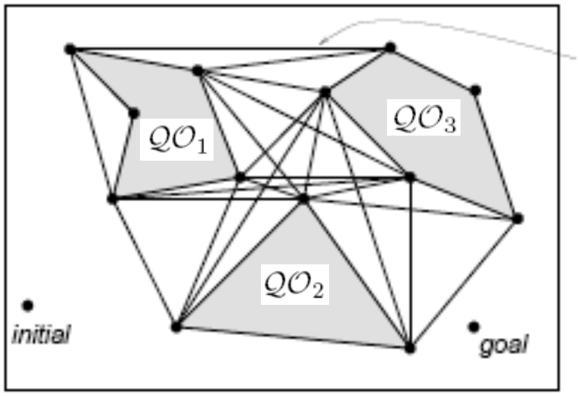
   The result is a path in Q_{free} from start to goal

---

# Overview

- Deterministic methods
    - Some need to represent $Q_{\mathrm{free}}$ 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

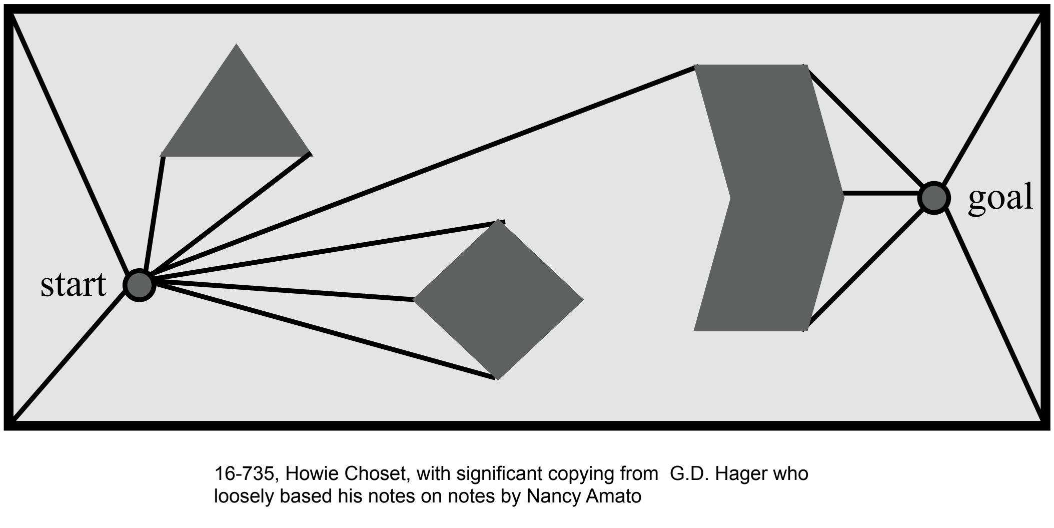
Does this work in 3D?

---

# 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.

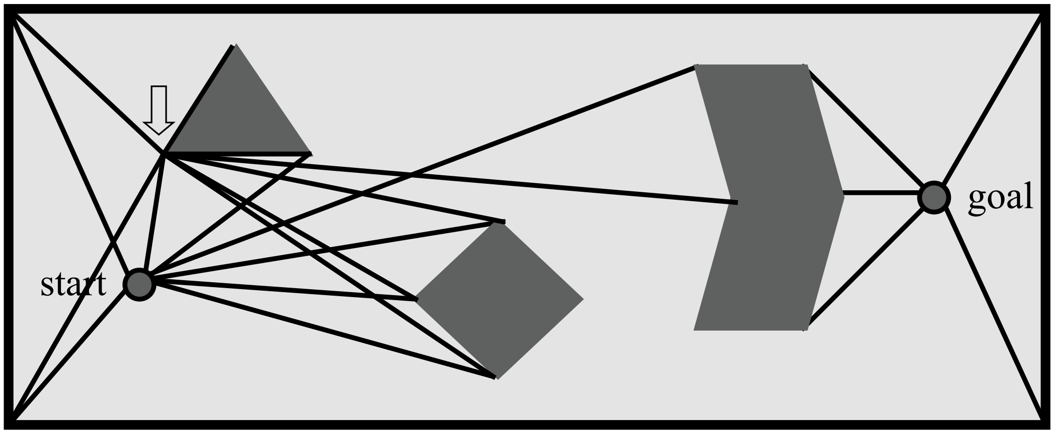
$$e_{ij} \neq \emptyset \iff sv_i + (1-s)v_j \in \mathrm{cl}(\mathcal{Q}_{\text{free}}) \quad \forall s \in (0,1)$$



# 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.

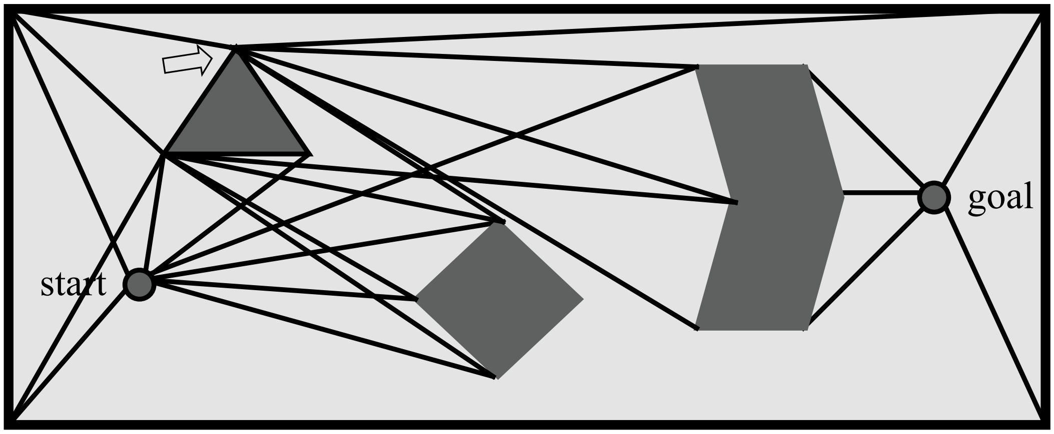
$$e_{ij} \neq \emptyset \iff sv_i + (1-s)v_j \in \mathrm{cl}(\mathcal{Q}_{\mathrm{free}}) \quad \forall s \in (0,1)$$



---

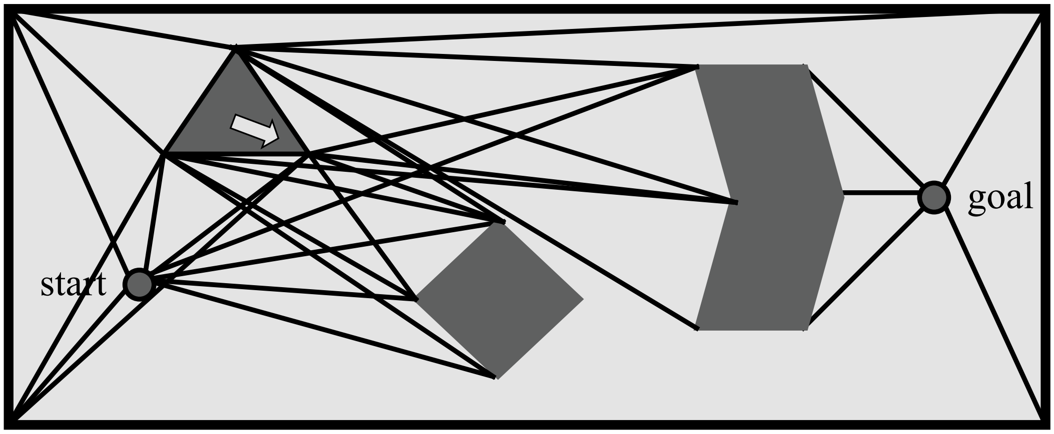
# The Visibility Graph in Action (Part 3)

- 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.

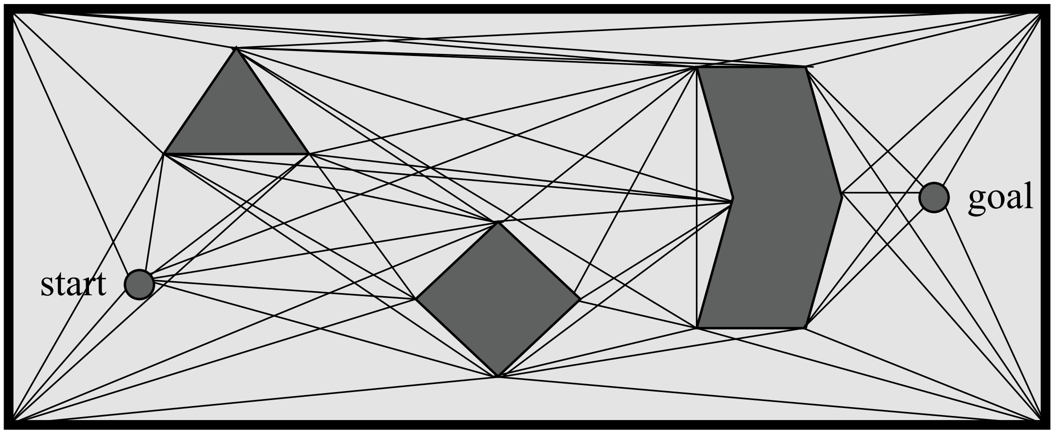


16-735, Howie Choset, with significant copying from G.D. Hager who loosely based his notes on notes by Nancy Amato

---

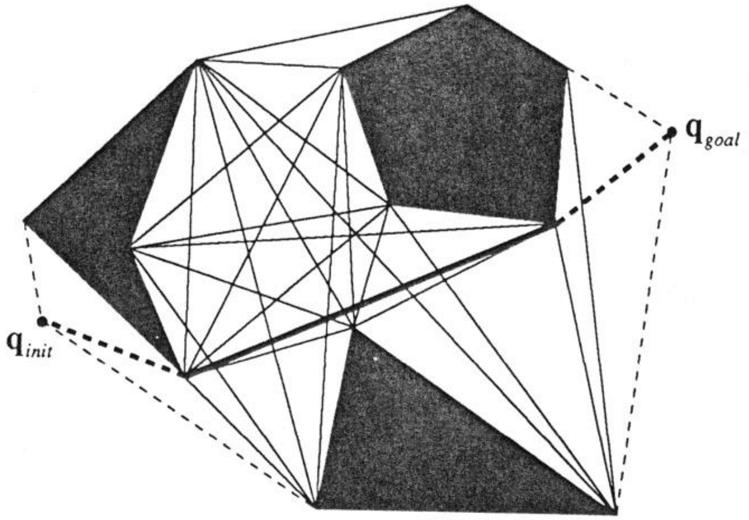
# The Visibility Graph (Done)

- Repeat until you're done.



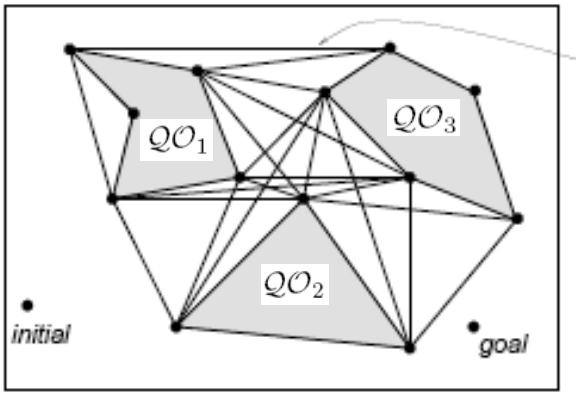
16-735, Howie Choset, with significant copying from G.D. Hager who loosely based his notes on notes by Nancy Amato

# Visibility Graphs



---

# 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

```
For each vertex $v_i$, calculate $\alpha_i$, the angle from the horizontal axis to the line segment $\overline{vv_i}$.
Create the vertex list $\mathcal{E}$, containing the $\alpha_i$'s sorted in increasing order.
Create the active list $\mathcal{S}$, containing the sorted list of edges that intersect the horizontal half-line emanating from $v$.
for all $\alpha_i$ do
  if $v_i$ is visible to $v$ then
    Add the edge $(v, v_i)$ to the visibility graph.
  end if
  if $v_i$ is the beginning of an edge, $E$, not in $\mathcal{S}$ then
    Insert the $E$ into $\mathcal{S}$.
  end if
  if $v_i$ is the end of an edge in $\mathcal{S}$ then
    Delete the edge from $\mathcal{S}$.
  end if
end for
```

$O(n)$

$O(n \log n)$

$O(n \log n)$

$O(n \log n)$ $n$ times (once for each vertex)

$O(\log n)$

If the line segment $\overline{vv_i}$ does not intersect the closest edge in $\mathcal{S}$, and if $l$ does not lie between the two edges incident on $v$ then $v_i$ is visible from $v$.

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

---

# Example

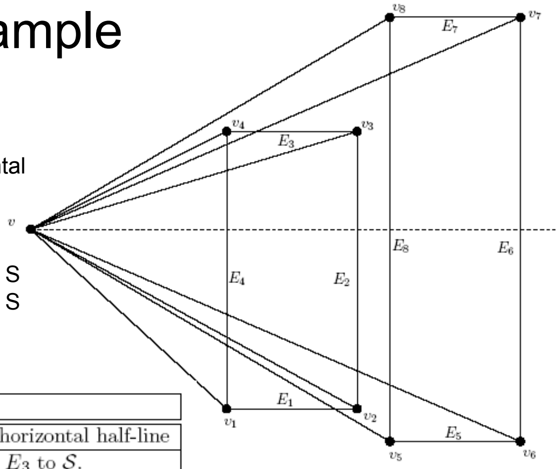
**Algorithm:**

Initially:

- calculate the angle $\alpha_i$ 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 $\alpha_i$

- 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



| Vertex | New $\mathcal{S}$ | Actions |
|---|---|---|
| Initialization | $\{E_4, E_2, E_8, E_6\}$ | Sort edges intersecting horizontal half-line |
| $\alpha_3$ | $\{E_4, E_3, E_8, E_6\}$ | Delete $E_2$ from $\mathcal{S}$. Add $E_3$ to $\mathcal{S}$. |
| $\alpha_7$ | $\{E_4, E_3, E_8, E_7\}$ | Delete $E_6$ from $\mathcal{S}$. Add $E_7$ to $\mathcal{S}$. |
| $\alpha_4$ | $\{E_8, E_7\}$ | Delete $E_3$ from $\mathcal{S}$. Delete $E_4$ from $\mathcal{S}$.<br>ADD $(v, v_4)$ to visibility graph |
| $\alpha_8$ | $\{\}$ | Delete $E_7$ from $\mathcal{S}$. Delete $E_8$ from $\mathcal{S}$.<br>ADD $(v, v_8)$ to visibility graph |
| $\alpha_1$ | $\{E_1, E_4\}$ | Add $E_4$ to $\mathcal{S}$. Add $E_1$ to $\mathcal{S}$.<br>ADD $(v, v_1)$ to visibility graph |
| $\alpha_5$ | $\{E_4, E_1, E_8, E_5\}$ | Add $E_8$ to $\mathcal{S}$. Add $E_5$ to $\mathcal{S}$. |
| $\alpha_2$ | $\{E_4, E_2, E_8, E_5\}$ | Delete $E_1$ from $\mathcal{S}$. Add $E_2$ to $\mathcal{S}$. |
| $\alpha_6$ | $\{E_4, E_2, E_8, E_6\}$ | Delete $E_5$ from $\mathcal{S}$. Add $E_6$ to $\mathcal{S}$. |
| Termination | | |

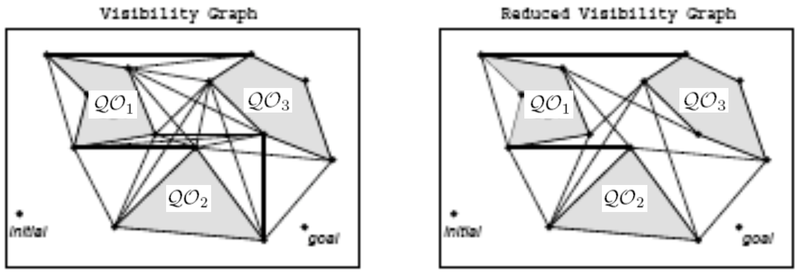
$$\mathcal{E} = \{\alpha_3, \alpha_7, \alpha_4, \alpha_8, \alpha_1, \alpha_5, \alpha_2, \alpha_6,\}$$

$(v, v_4)$, $(v, v_8)$, and $(v, v_1)$

# Reduced Visibility Graphs

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

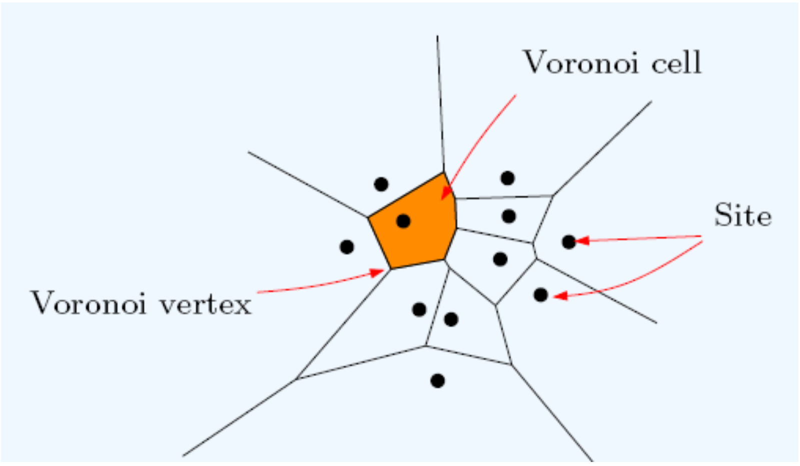
- A reduced visibility graph consists of
  - nodes that are convex
  - edges that are “tangent” (i.e. do not head into the object at either endpoint)



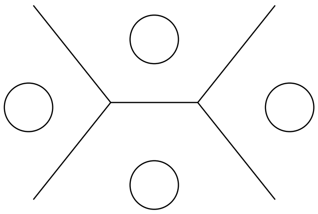
Can we extend beyond $\Re^2$

---

# Voronoi Diagrams



# Another Roadmap Idea: Voronoi Diagrams



Recall the brushfire algorithm produces a set of points that are equidistance from all obstacles

Almost by definition, this set of points (the Voronoi diagram) is a roadmap since each “watershed” can be connected to the backbone, then the backbone connects to all other “watersheds”

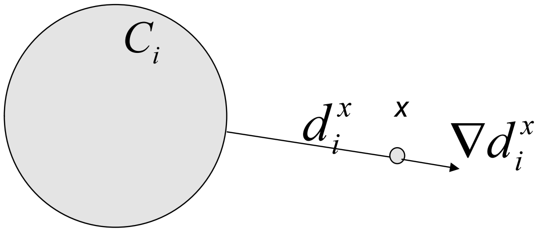
The books discusses deformation retracts which are a generalization of Voronoi diagrams to higher dimensions

---

# Beyond Points: Basic Definitions

*Single convex object distance function*

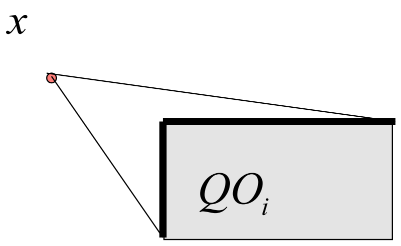
$$d_i^x(q) = \min_{c \in \mathcal{QO}_i} d(q, c)$$



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

# X for “X-ray”

Points within line of sight



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

---

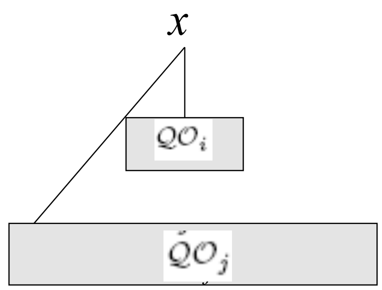
# Visible Distance Functions

- *Single-object*

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

- *Multi-object*

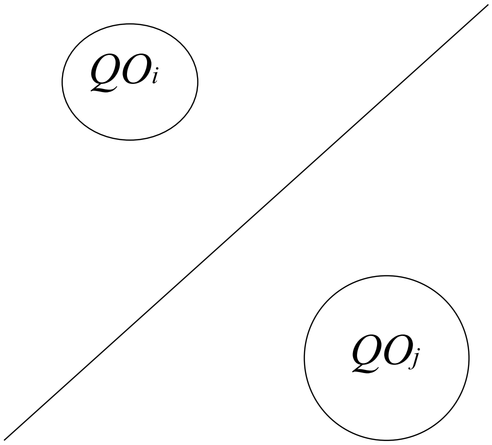
$$D(x) = \min_i d_i(x)$$



# Two-Equidistant

- *Two-equidistant surface*

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

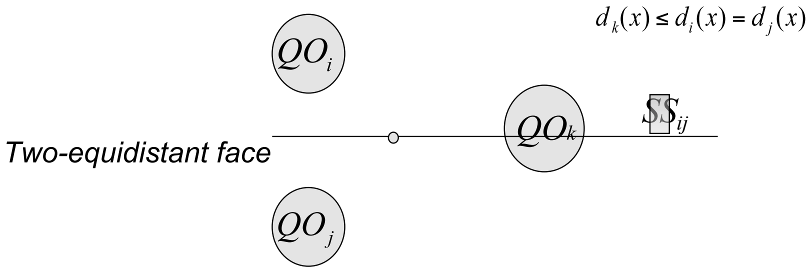


16-735, Howie Choset, with significant copying from G.D. Hager who
loosely based his notes on notes by Nancy Amato

---

# More Rigorous Definition

Going through obstacles

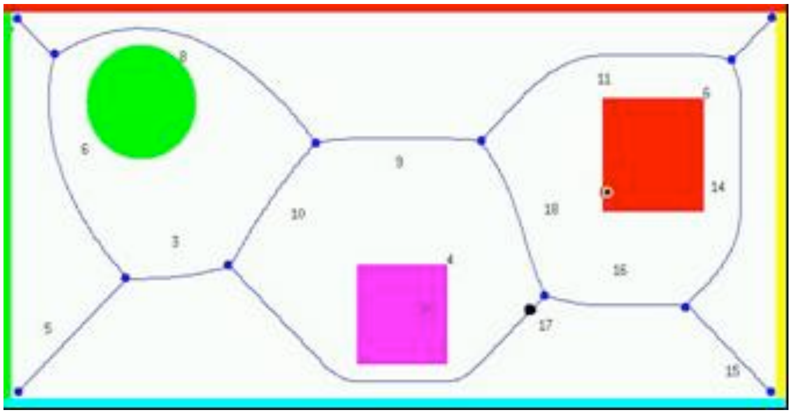


$$F_{ij} = \{x \in SS_{ij} : d_i(x) = d_j(x) \le d_h(x), \forall h \ne i, j\}$$

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

# General Voronoi Diagram

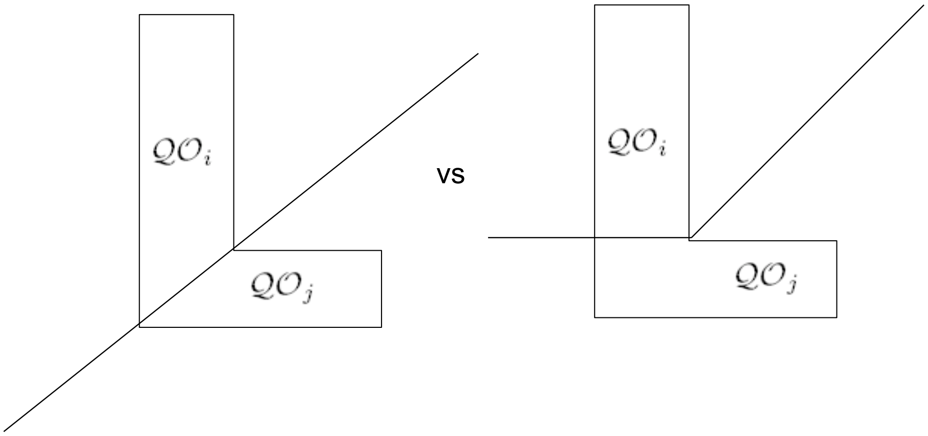
$$\mathrm{GVD} = \bigcup_{i=1}^{n-1} \bigcup_{j=i+1}^{n} F_{ij}$$



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

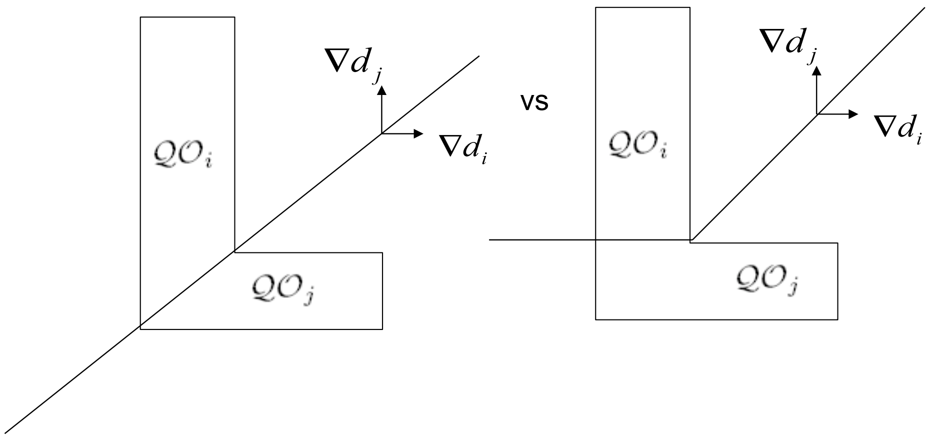
---

# 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

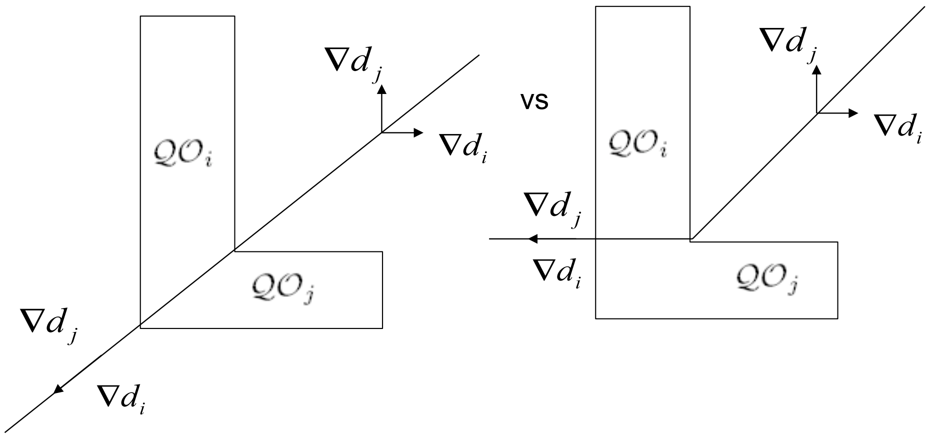
# What about concave obstacles?



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

---

# 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*

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

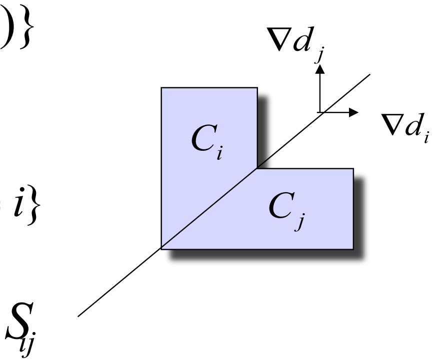
Two-equidistant surjective surface

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

Two-equidistant Face

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

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

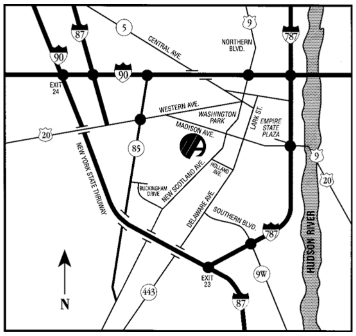


# Is This A Roadmap?

- How do we know the roadmap is connected?
  - The book talks about deformation retracts as a way of showing this; we will not cover this

## Silhouette Method

## Canny's Roadmap Algorithm
## The Opportunistic Path Planner



---

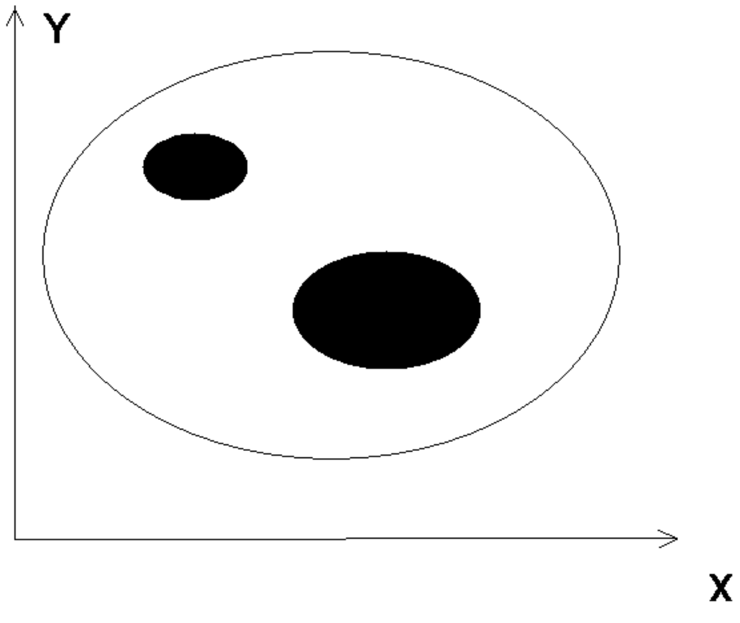
## The Basic Ideas

- Pick a sweeping surface (n-1 dimensional)
- As sweeping happens, detect extremal points (roadmap) and critical points (= places where connectivity changes)
  - Note extremal points are 1-dimensional curve
- For each slice where a critical point occurs, repeat this process recursively
  - This connects extremal points
- Use this as the roadmap
  - A variation: use this to define a more interesting roadmap

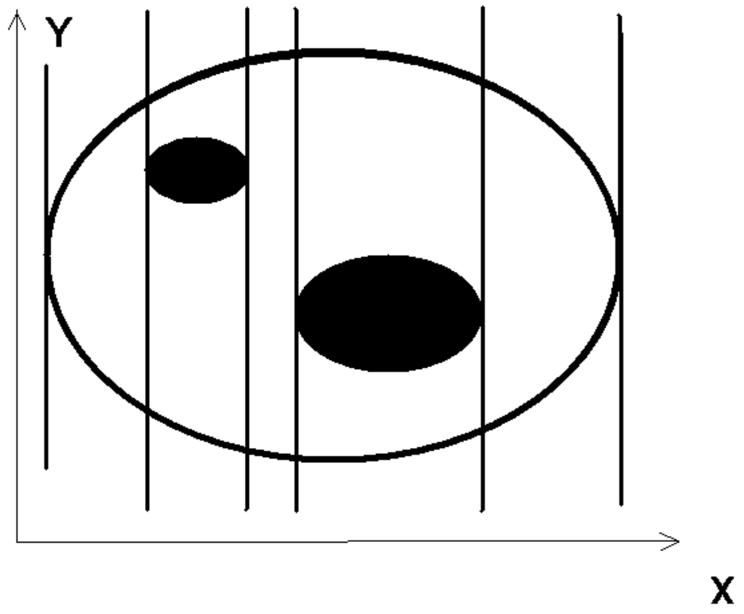
# Accessibility and Departability (1)

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.

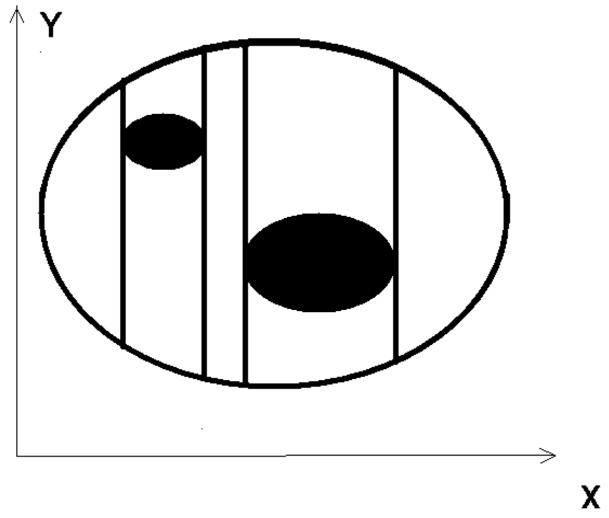
# Accessibility and Departability (2)



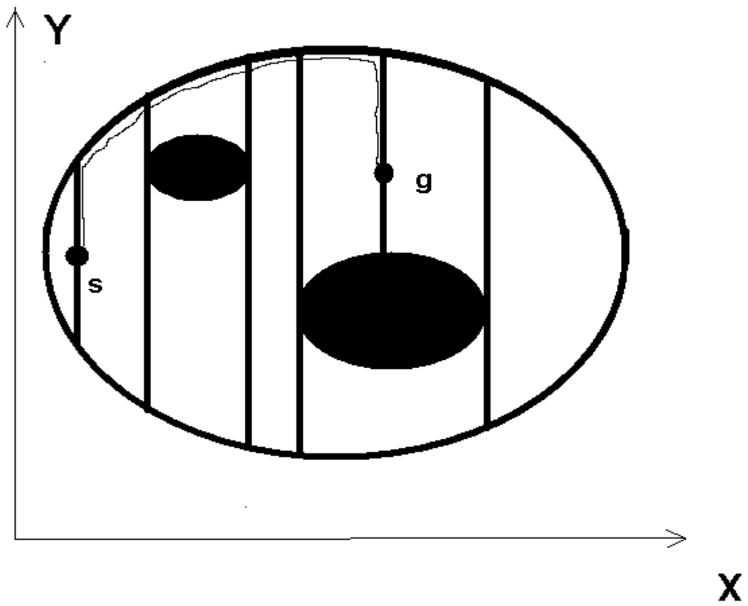
# Accessibility and Departability (3)



# Accessibility and Departability (4)



## Accessibility and Departability (5)



## 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 $\pi_1$ is the projection on to the first coordinate

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

# How To Find Extrema (1)

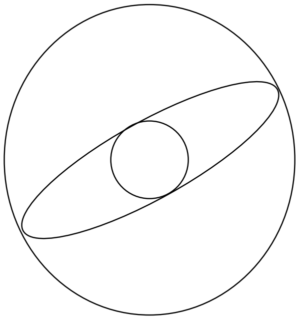
When constructing the silhouette curves, we look for extrema of $\pi_2|S_c$, the extrema of the projection of $S_c$ in a second direction.

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 $\pi_{12}|S$, $\sum(\pi_{12}|S)$,

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 $h=\pi_{12}(x,y,z)=(x,y)$.

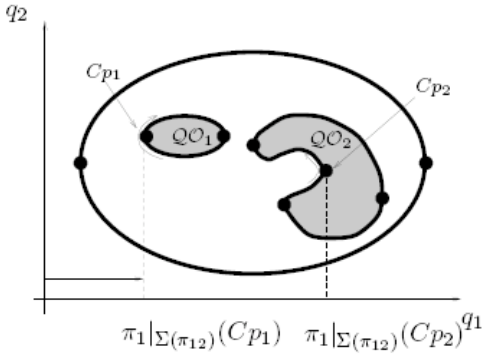
$$d(f,h) = \begin{bmatrix} 2x & 2y & 2z \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

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

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



$\pi_1(q)$

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.

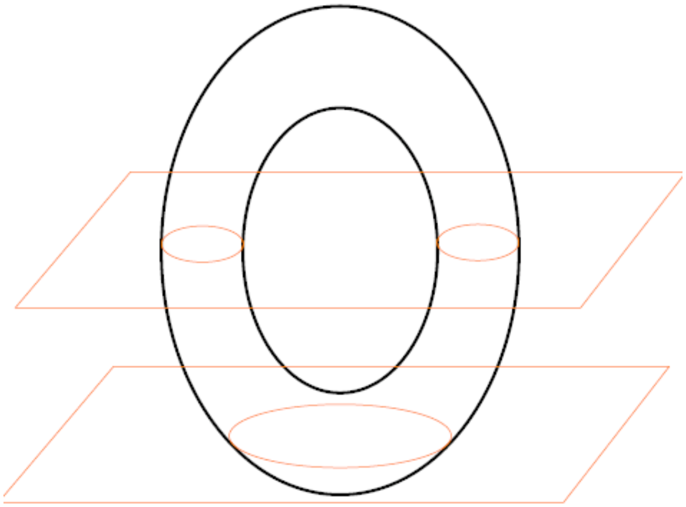
This matrix forms the top m-1 row $D(f,\pi)$

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

Slice function $\pi_1$ has $D(f,\pi)$t $[1, 0, \ldots, 0]^T$ which forms the bottom row of

When roadmap tangent lies in slice plane, this means $T^{\perp}$ t $\nabla\pi_1(q)$ lies in $T^{\perp}$ ne ar $D(f,\pi)$ gonal to each other looses rank

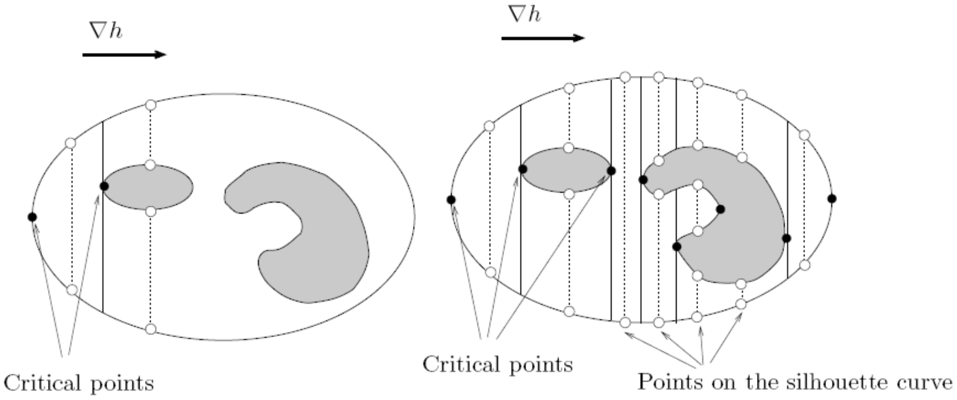
# Connectivity change at Critical Points



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

---

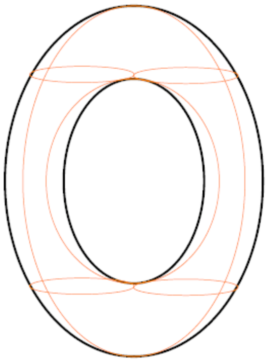
# Between Critical Points



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

# Building the Roadmap (Conclusion)

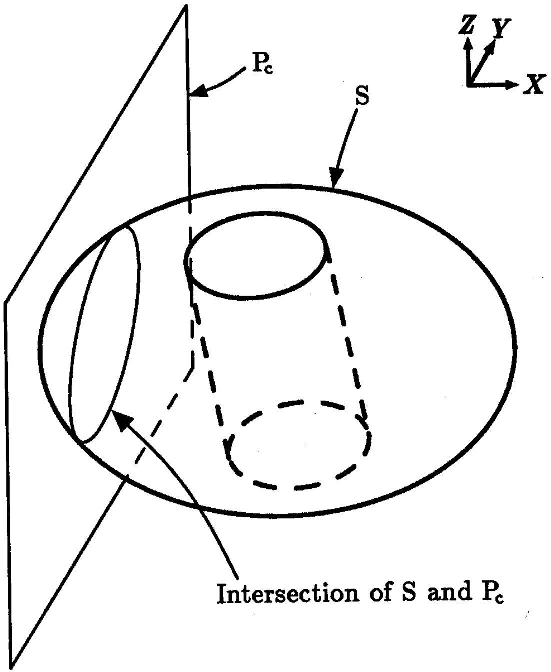
- We can now find the extrema necessary to build the silhouette curves
- We can find the critical points where linking is necessary
- We can run the algorithm recursively to construct the whole roadmap



---

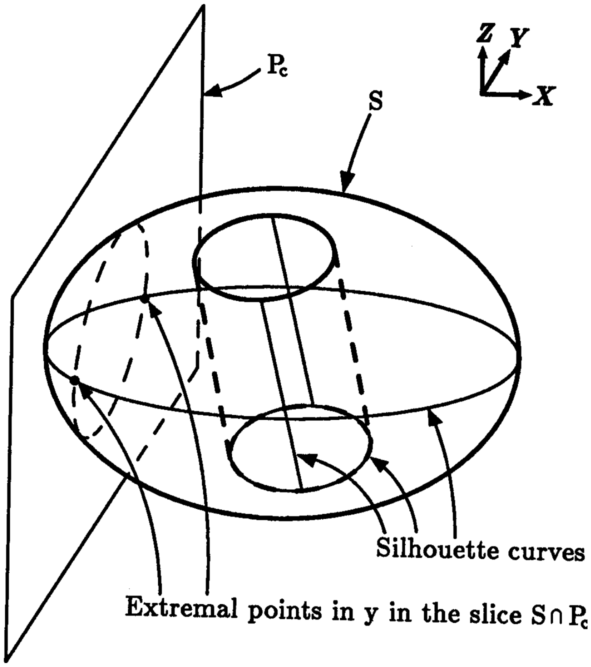
# Illustrative Example (1)

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**.



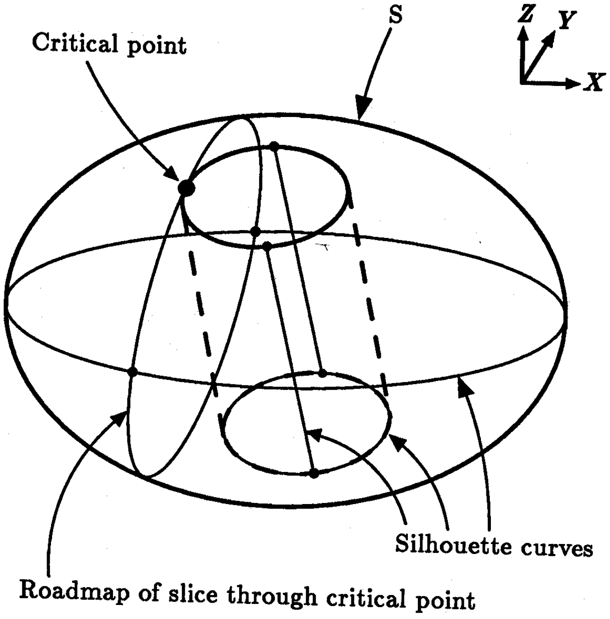
# Illustrative Example (3)

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