# LaValle Ch 6: Combinatorial Motion Planning

3/4/18



a.k.a. "exact motion planning".

"combinatorial" comes from the need to check every possibility

"exact" captures need to use geometric models accurately

Perhap "exact combinatorial MP" would be a more fitting name.

These algorithms are ~~exac~~ complete. The narrow passage "problem" is not a problem for these algorithms!

General formulations of general M.P. problems are PSPACE-hard, so ~~are~~ best attacked with sample-based methods.

However, when the dimension of C-space is low and geometries are simple (e.g., convex polygons & quadrics) exact combinatorial MP algorithms can be far superior to sample-based methods, especially when clearances are tight (e.g., assembly problems).

In E.C.M.P. algorithms, geometric representation ②

becomes important and can't be hidden in a

Collision detection black box.

Some ECMP algs are impractical even if they are best.

e.g. Canny's Roadmap Method, 1986. Still not implemented. in its generality.

Nearly all ECMP algs construct some sort of roadmap, which has the following properties:

① Accessibility: from any $q \in C_{free}$, it is simple & efficient to compute a path $\tau : [0,1] \rightarrow C_{free}$ such that $\tau(0) \in S(G)$ and $\tau(1) = q$.

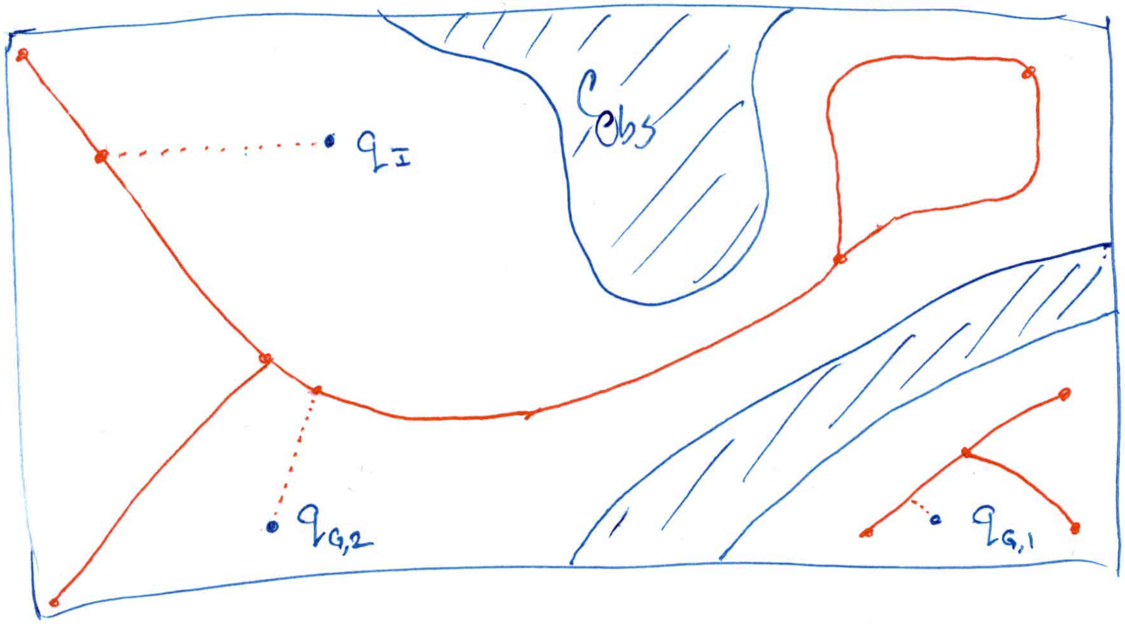
The swath of the Graph.

② Connectivity preserving: Using accessibility, it is alway possible to connect $q_I$ & $q_G$ to $G$ (at $s_1$ & $s_2$). If a path $\exists$ in $C_{free}$ connecting $q_I$ to $q_G$, then a path from $s_1$ to $s_2$ exists! That means $G$ captures the connectivity of $C_{free}$.

3/4/18

3

The two properties in pictures:




Easy to connect any $q_I, q_G$ pair to $\mathcal{G}$.

If path exists $((q_I, q_{G,2}))$, it is found by discrete graph search.

If path doesn't exist $((q_I, q_{G,1}))$, it is reported that no solution exists (by discrete graph search).

---

Virtually every roadmap is a 1D retraction of $C_{free}$.

Typically $C_{free}$ is $\left\{\begin{matrix}\text{partitioned}\\ \text{decomposed}\end{matrix}\right\}$ into cells, and then the cells are retracted to 1D arcs in $C_{free}$.

The best known ECMP alg for the "generalized (piano) movers'" problem is due to John Canny 1986 and is $O(2^n)$ where n is the dimension of C-space. Geometries of bodies & robot link must be semi-algebraic sets.

The first ECMP alg for robotics was on the "piano movers'" problem. Published by Schwartz and Sharir in the 1980's. Was based on Collins' Decomposition, which is $O(2^{2^n})$.

However the # of connected components of $C_{free}$ is $O(2^n)$. This gave Canny motivation to improve upon the results of Schwartz & Sharir.

# LaValle Ch6.2 Polygonal Obstacles in C-space

Applicable cases: both are in the plane:

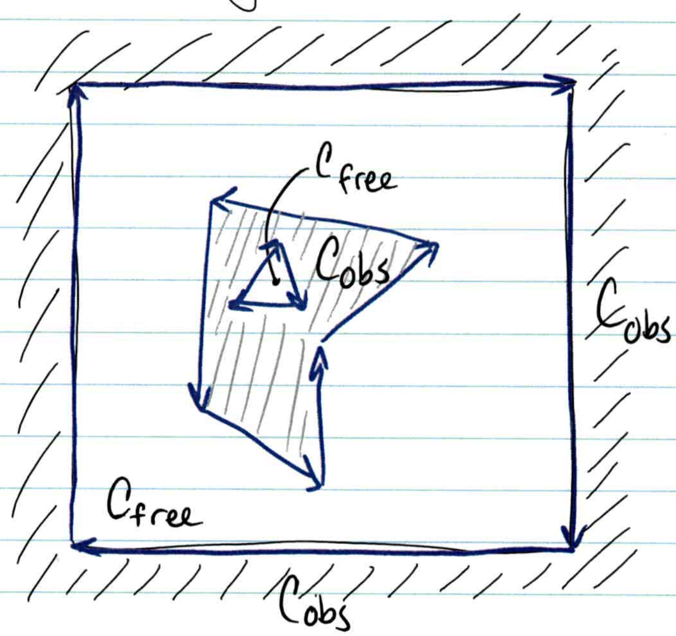
(1) point robot in polygonal world ($C_{obs}$ = O in World)

(2) polygonal robot translating in polygonal world. ($C_{obs}$ constructed by Minkowsky difference)

Motion planning reduces to planning the path of a point thru a polygonal obstacle field

We need a convenient data structure for representing the polygons and their connectivity.

Polygons may be nonconvex including possibility of having holes.




Vectors follow bndries such that $C_{obs}$ is always on the left.

We need to keep track of holes and allow possibility of decomposed polygons.

For efficient access to data needed for planning when

A vertex

$C_{obs}$ is a collection of polygons

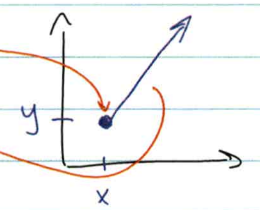
Lavalle recommends:

could have many edges emanating. Show just one here.

vertex:

vertex.location ← x,y coords

vertex.half_edge ← any half edge

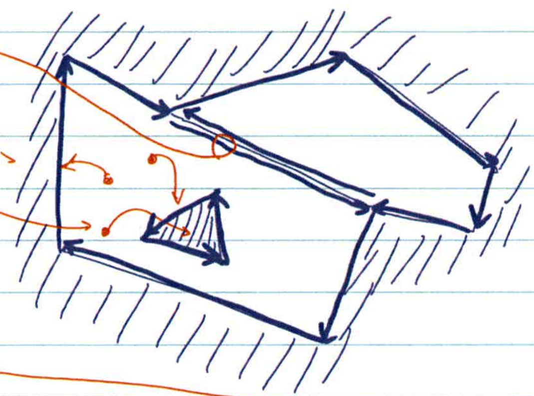



face:

face.half_edge_inner

face.half_edge_outer

face.hole_in_face



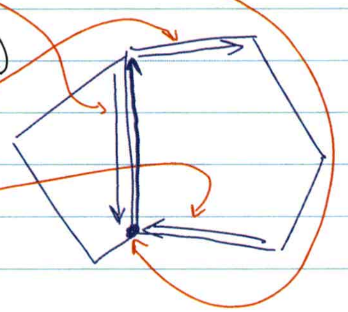
half_edge:

half_edge.origin_vertex

" .twin_edge

" .internal_face (NIL if obstacle)

makes list doubly-connected
- " .next_edge
- " .previous_edge



---

## 6.2.2 Vertical Cell Decomposition

Decompose $C_{free}$ into regions (cells) that are:

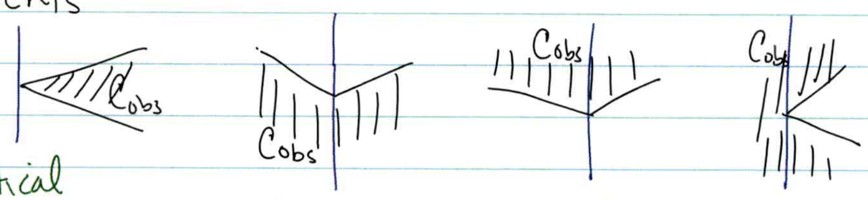
1. Connection of any 2 points in cell is "easy."
2. Cell adjacency info easily extracted.
3. "Easy" to tell cell membership of any given point.

If the cell decomp. satisfies the above 3 characteristics, then motion planning reduces to graph search.

Defining the decomposition for $C_{free} \subset \mathbb{R}^2$ and $C_{free}$ and $C_{obs}$ are polygonal.

Sweep a vertical line from left to right
Keep track of data at "critical" events

Critical events

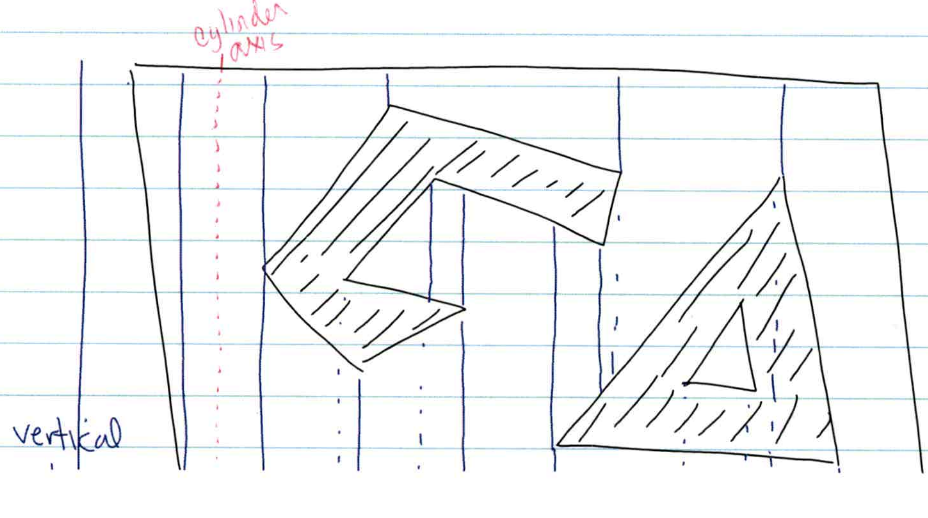


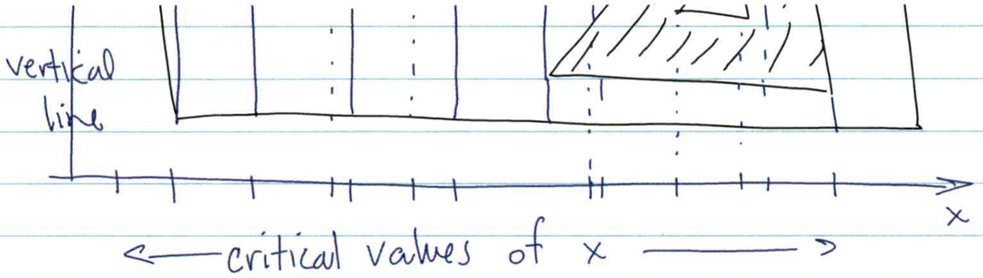

vertical line splits into two

lower or upper intersection changes edges

new piece of line emerges.

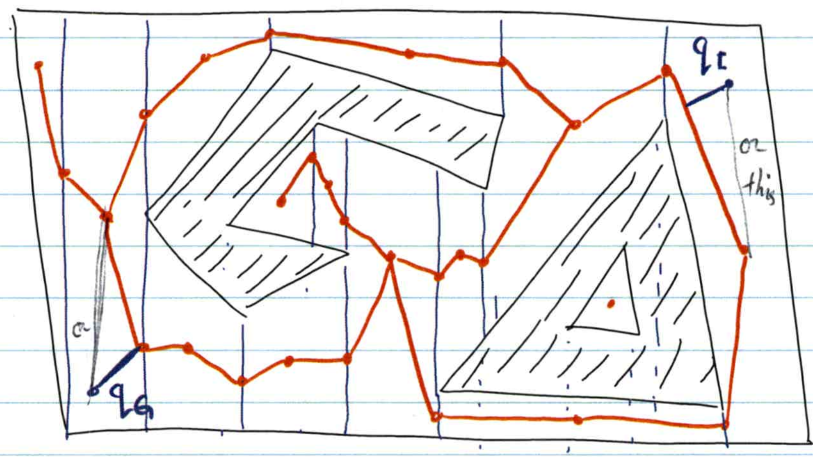
Example






Now add one vertex of $\mathcal{G}$ in each 1-cell & each 2-cell.

Connect vertices to those in adjacent cells.




Note: Figure 6.4 in LaValle has an extra edge and a missing edge. Can you find them?

Query phase:

Place $q_I$ & $q_G$.
connect to roadmap
search graph
extract path.

---

Complexity

Straightforward alg. (vertices not sorted on sweep dir)

Let $n$ be the # of vertices of $C_{obs}$.

Then there are $O(n)$ critical events

For each ~~critical~~ vertex event, intersect the vertical line w/ all $O(n)$ edges of $C_{obs}$.

$\Rightarrow O(n^2)$

Best known alg. is $\boxed{O(n \cancel{lg} \log n)}$

Sort line segments intelligently, ~~then go through once doing crit. events & intersections~~

Sort vertices according to their distance along the sweep direction. This operation is $O(n \cancel{lg} \log n)$. Process $O(n)$ critical points. This requires examinin two edges per critical point, so $O(n)$ work.

$O(n \log n) + O(n) = O(n \log n)$

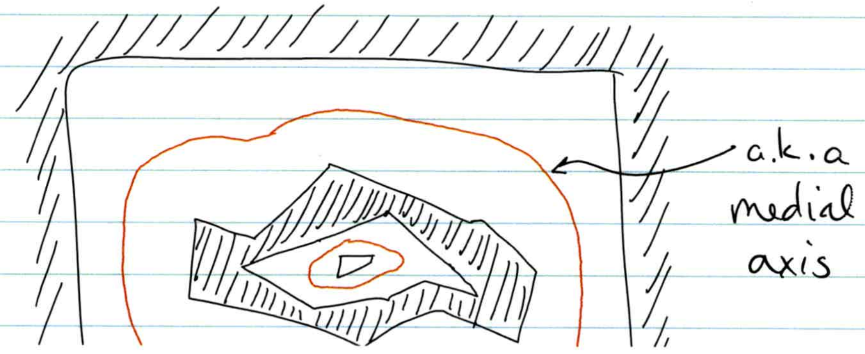
---

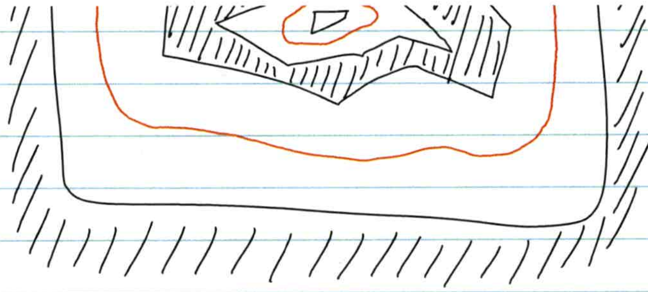
Maximum Clearance Roadmaps – Generalized Voronoi Diagrams

Used for inaccurate ~~mobile~~ not nec. mobile robots

Create roadmap that is maximally far from all obstacles.

As above, assume polygonal $C_{obs}$ & $C_{free}$

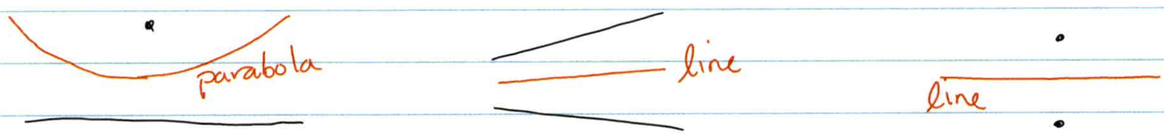




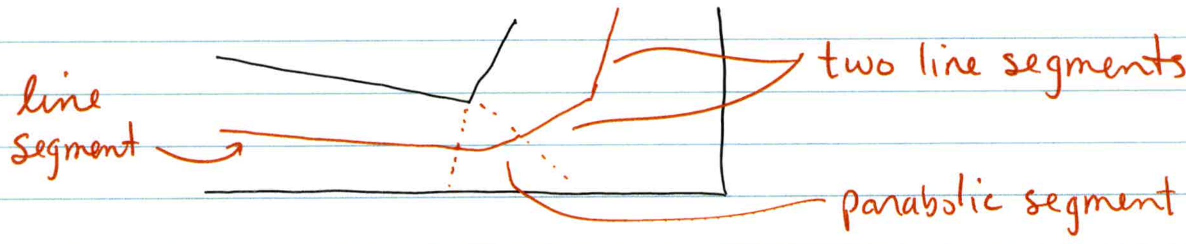
axis

Three cases:

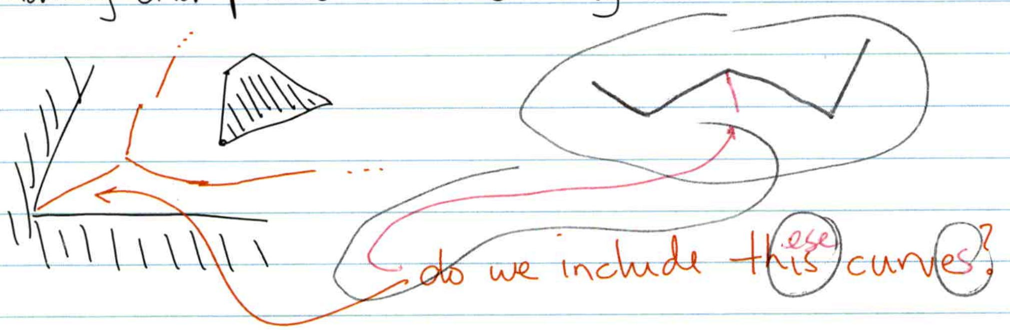
vertex-edge        edge-edge        vertex-vertex

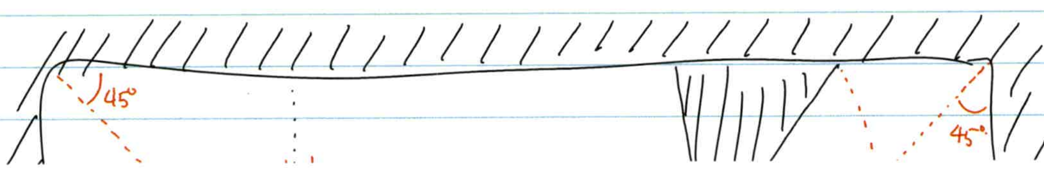


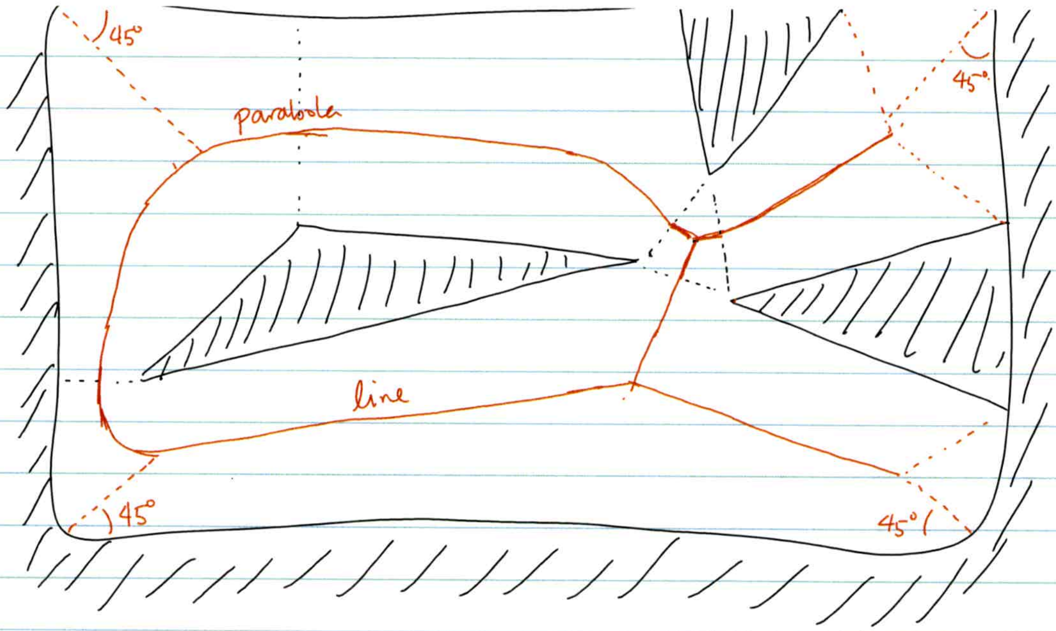

The roadmap is formed by connecting these three primitives. Consider a simple case:




Handling end-points is not clearly defined.






Complexity:

Let $n$ be the number of edges (and vertices) of $C_{obs}$.

$O(n^2)$ – For every feature pair, compute the line or parabola.

$O((n^2)^2)$ – Intersect every pair of line-line or line-parabola pairs to piece together the roadmap.

$\therefore$ Straight forward approach is $\boxed{O(n^4)}$ where $n$ is the number of geometric features.

Best known alg is $O(k \lg k)$, where $k$ is the # of curve segments in the roadmap. But $k = O(n^2)$, so best known alg is $\boxed{O(n^2 \lg(n^2))}$

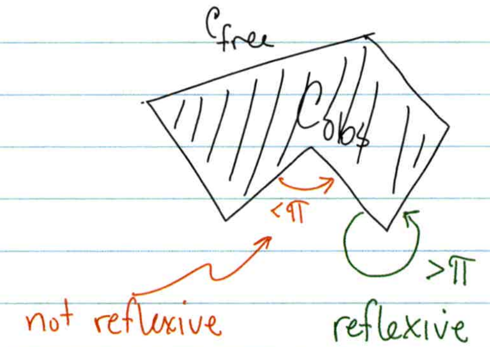
---

Shortest Path Roadmaps

Useful when mobile robot is accurately controlled and short paths are important.

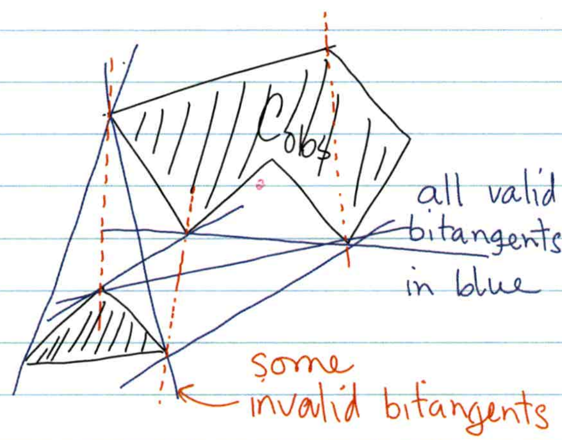
Definition: Reflexive vertex: vertex of $C_{obs}$ with angle in $C_{free}$ greater than $\pi$

locally convex vertex




Definition: Bitangent:

A bitangent edge connects two reflexive vertices that are mutually visible AND extending the bitangent edge infinitesimally beyond the vertices does not intersect $Int(C_{obs})$

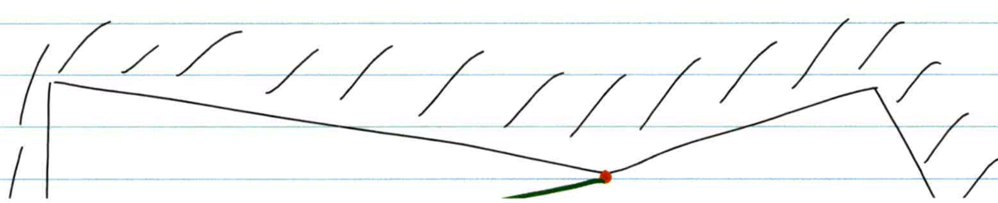


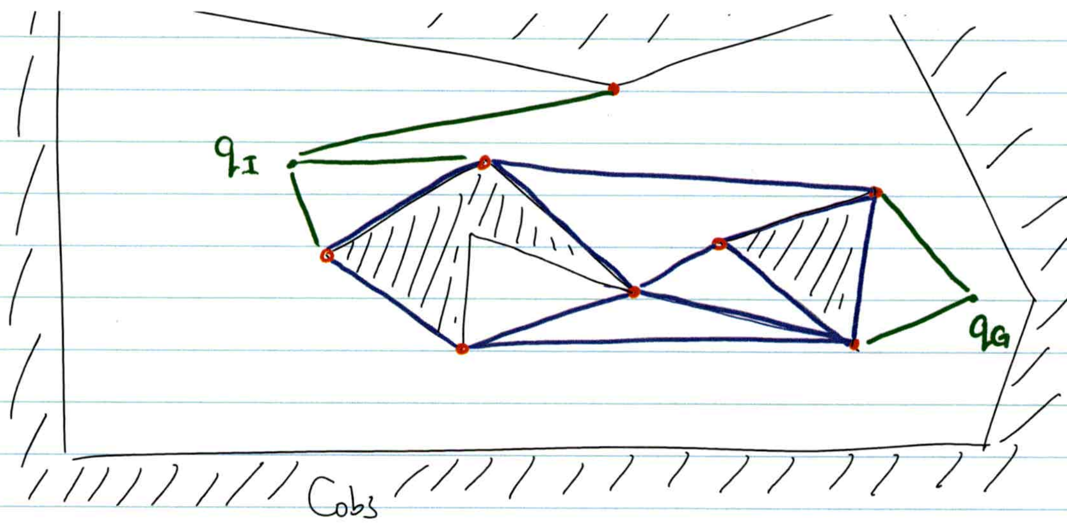

## Construct the roadmap

Initialize $G$ as the set of reflexive vertices

Create edges between all pairs of vertices in $G$ that define a bitangent edge.

Note: By definition the edge between consecutive reflexive vertices on one body is a valid bitangent edge.






To solve query, insert $q_I$, $q_G$, then create edges from $q_I$ & $q_G$ to every visible vertex in S (see above green links).

Finally search graph for a path.

---

Complexity:

Straight forward approach.

Let n be # of vertices of Cobs.

Find all reflexive vertices – $O(n)$
+
Find all valid bitangents – $O(n^2)$ $\Big\} \; O(n^2)$

Check all bitangents for intersections with Cobs edges. – $O(n)$

$\therefore$ overall we have $O(n^3)$

Better algs exist – $\boxed{O(n^2 \lg n)}$

Better algs exist – $\boxed{\begin{array}{l} O(n^2 \lg n) \\ \hline O(n \lg n + m) \end{array}}$

and

where $m$ is the # of roadmap edges.