Lavalle_Ch6_1-2

Wednesday, April 15, 2009
4:00 PM

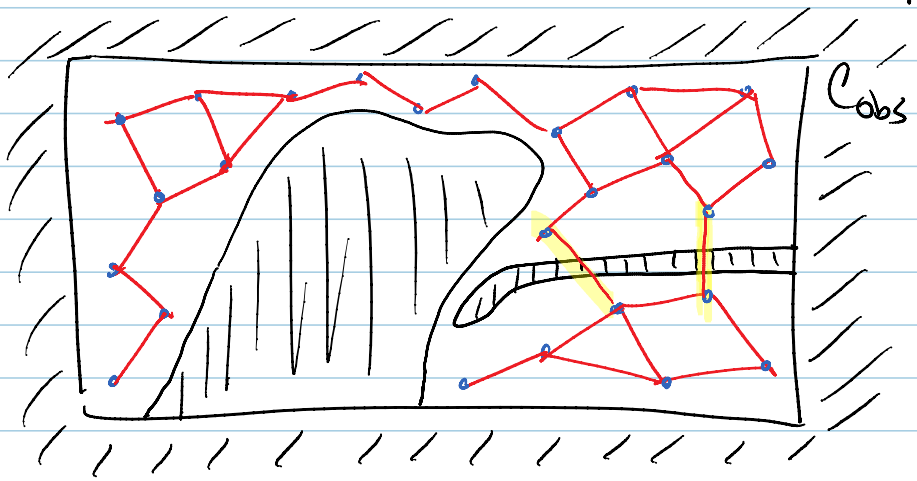
Combinatorial Motion Planning

a.k.a. Exact motion planning

"Combinatorial" comes from the need to check every possibility.

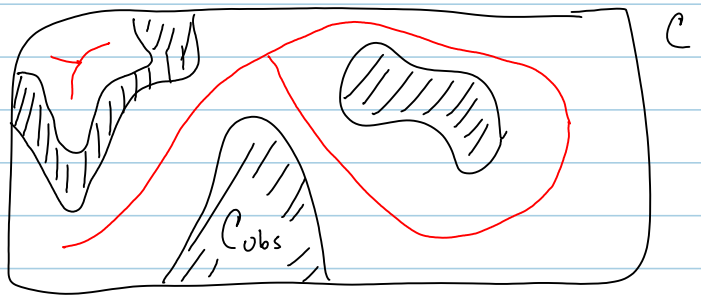
Combinatorial M.P. algs. are complete. i.e. in finite time the alg will report a solution or the fact that a solution does not exist.

The algs. in Ch 5 are not complete! Why? Because we only approximate C_{free} with a topological graph. One never knows if disconnect components of the graph are actually connected. If arcs are tested at discrete points, then arcs could be invalid, thus generating invalid plans.



The algs in Ch 6 are based on exact knowledge of C_{free}. The "narrow passage" problem is not a problem for exact methods!

Virtually every combinatorial method constructs a roadmap – a retraction of C_{free} to a 1-dimensional set.



First general algs. appeared in the early 1980's. Schwartz & Sharir proved that the Piano Movers' problem is intractible. Based on Collins' Decomposition of C_{free} into cells. But Collins' decomp is $O(2^{2^d})$ ie. doubly exponential!

(where n is the dimension of C space).

(for the generalized movers' problem)

The best possible algorithm for exact M.P. is $O(2^n)$. (Canny 1986 thesis). This complexity comes from the fact that in the worst case, the # of

connected components of C space is $O(2^n)$.

Implementation of Canny's alg is so difficult that it has never been done.

Despite high algorithmic complexity, there are still problems where combinatorial/exact method are tractible & the best choice.

e.g. Mobile robot that is small incomparison to geometric features of its environment.

Let G be a topological graph that maps into C_{free}.
Let S be the swath of G.

Definition (LaValle): G is a roadmap if:

- G is accessible from C_{free} $\Rightarrow$ trivially easy to connect any $q \in C_{free}$ to S
- G preserves connectivity of C_{free} $\Rightarrow$ there is a one-to-one and onto correspondence between the components of G and those of C_{free}

Question: Is the second bullet subsumed by the first?

Section 6.2 – C_{free} & C_{obs} are polygonal

Applicable cases: both are in the plane:

(1) point robot in polygonal world

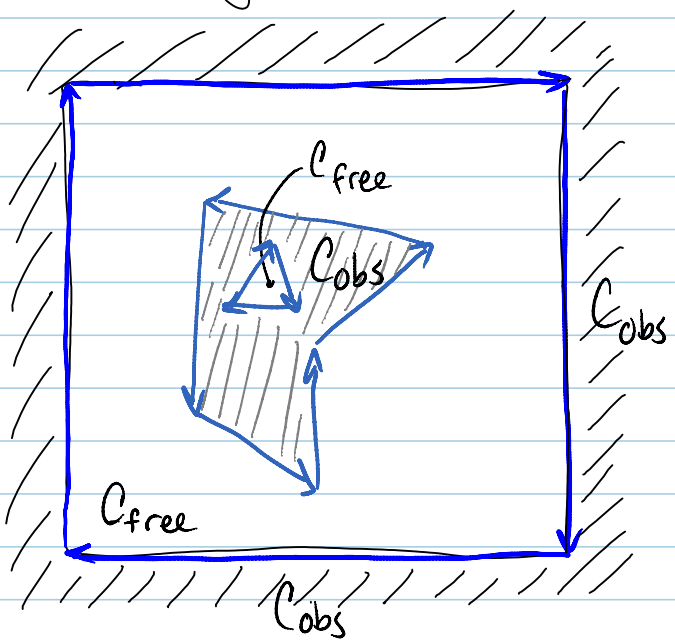
(2) polygonal robot translating in polygonal world.

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

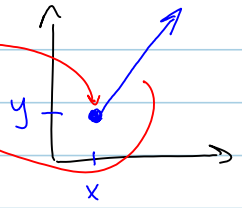
could have many edges emanating. Show just one here.

Lavalle recommends:

vertex:

vertex.location ← x,y coords

vertex.half_edge ← any half edge

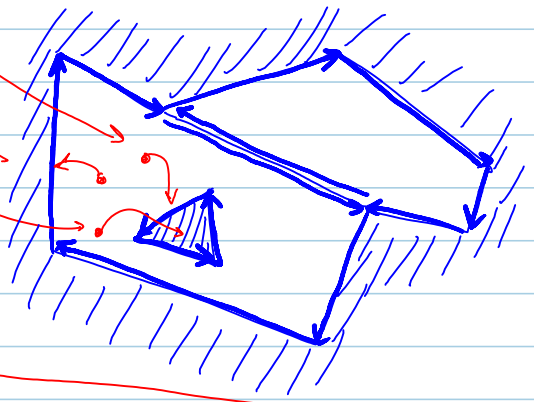


face:

face.half_edge_inner

face.half_edge_outer

face.hole_in_face



half_edge:

half_edge.origin_vertex

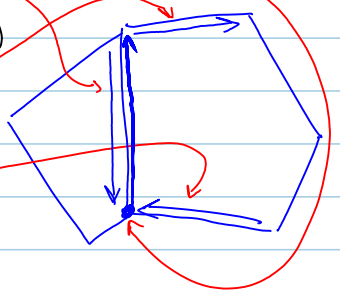
" .twin_edge

" .internal_face (NIL if obstacle)

" .next_edge

" .previous_edge

makes list doubly-connected



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

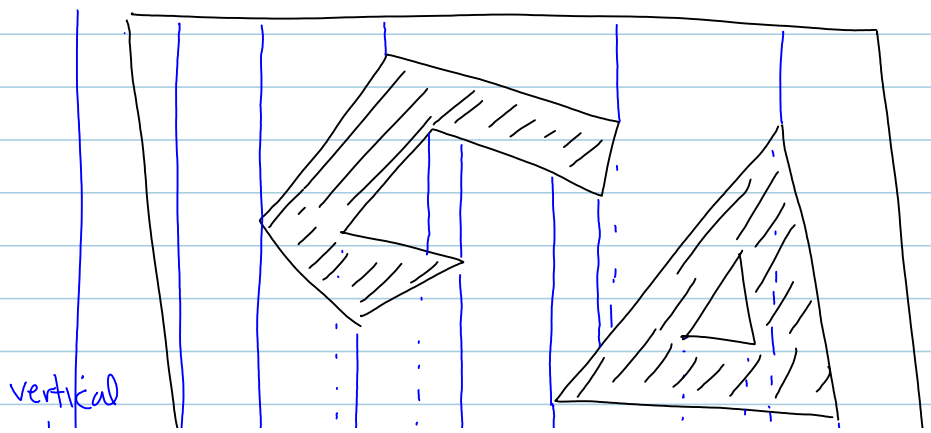
Critical events

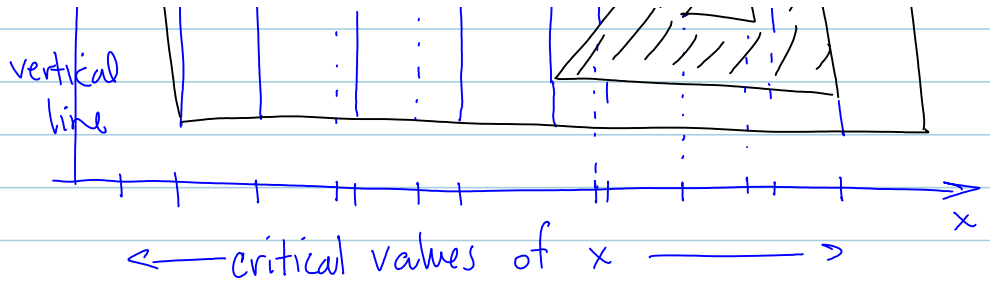
vertical line splits into two

lower or upper intersection changes edges

new piece of line emerges.

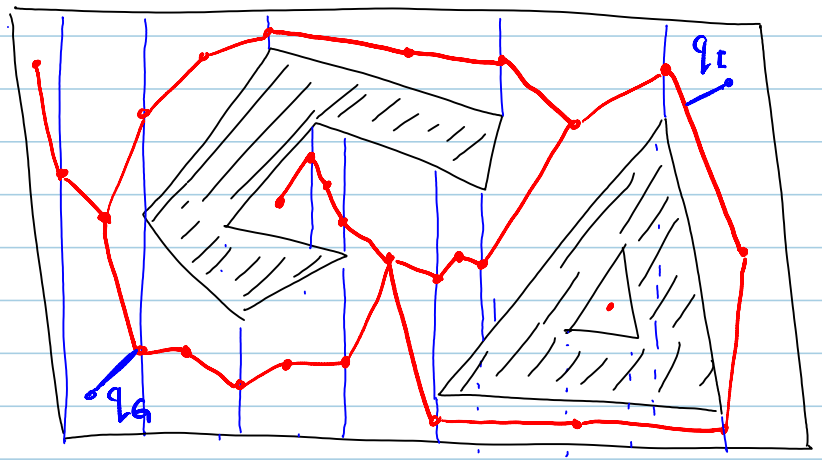
Example





Now add one vertex of 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

Straightforwad alg.

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

Then there are $O(n)$ critical events

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

$\Rightarrow O(n^2)$

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

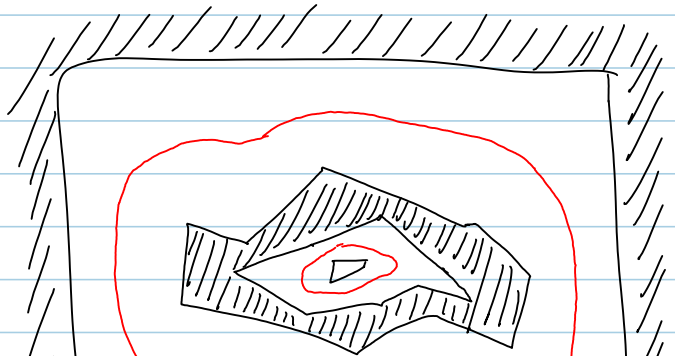
Sort vertices according to their distance along the sweep direction. This operation is $O(n \lg n)$. Process $O(n)$ critical points. This requires examining two edges per critical point, so $O(n)$ work.

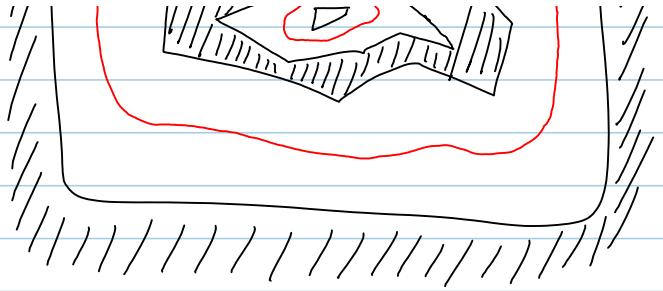
Maximum Clearance Roadmaps – Generalized Voronoi Diagrams

Used for inaccurate mobile robots

Create roadmap that is maximally far from all obstacles.

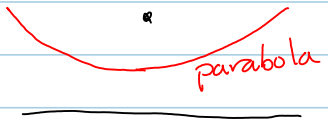
As above, assume polygonal C_{obs} & C_{free}



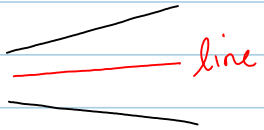


Three cases:

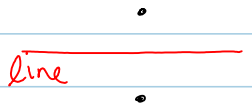
vertex – edge



edge – edge



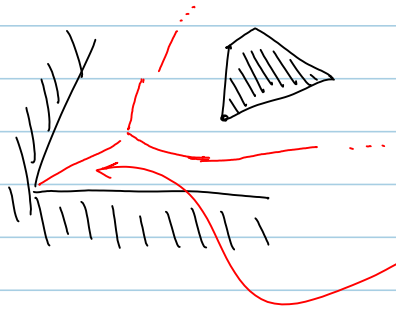
vertex – vertex

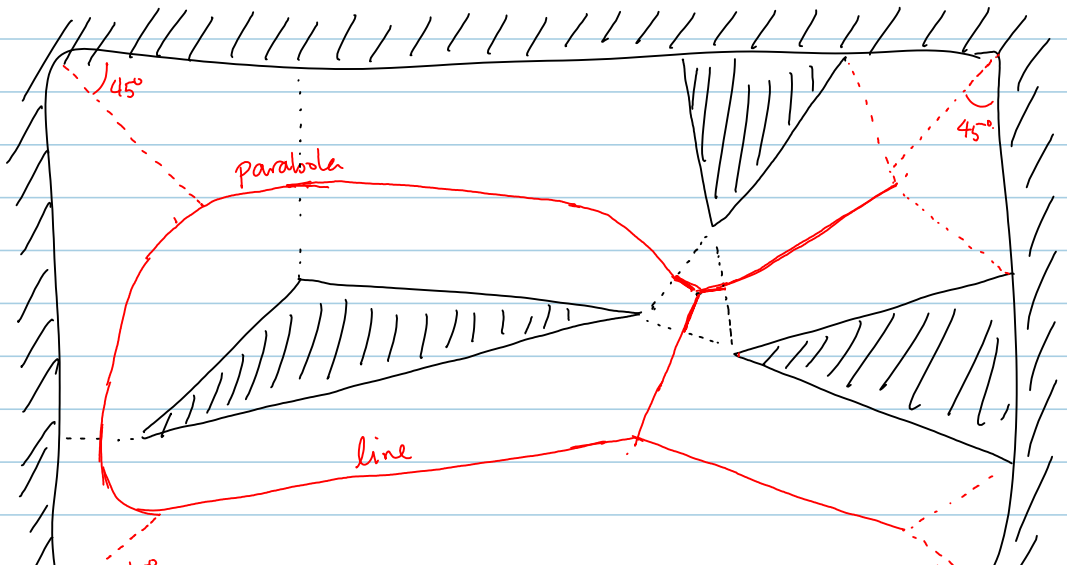


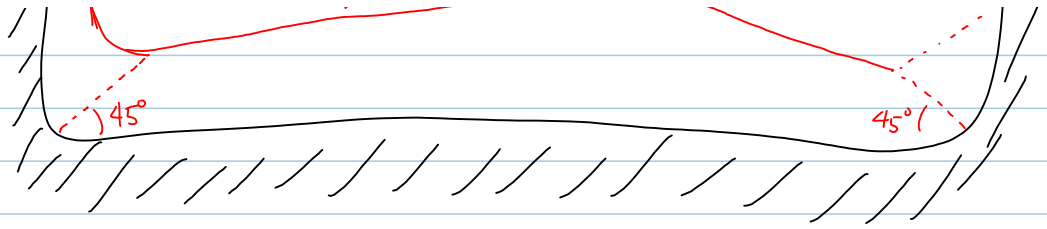
The roadmap is formed by connecting these three primitives. Consider a simple case:

∴

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.

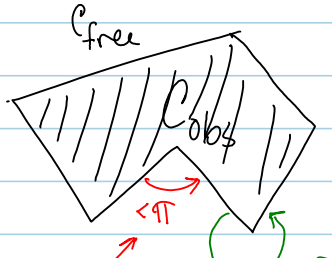
∴ Straight forward approach is $\boxed{O(n^4)}$ where n is the number of geometric features.

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 $\boxed{O(n^2 \lg(n^2))}$

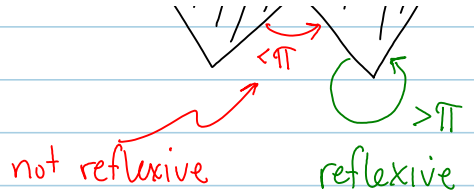
Shortest Path Roadmaps

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 π

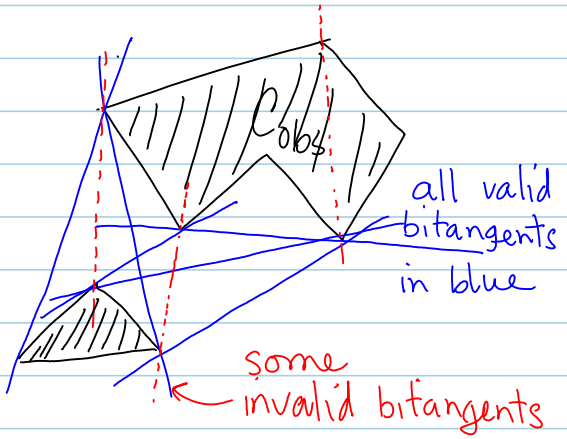


in C_{free} greater than π



Definition: Bitangent:

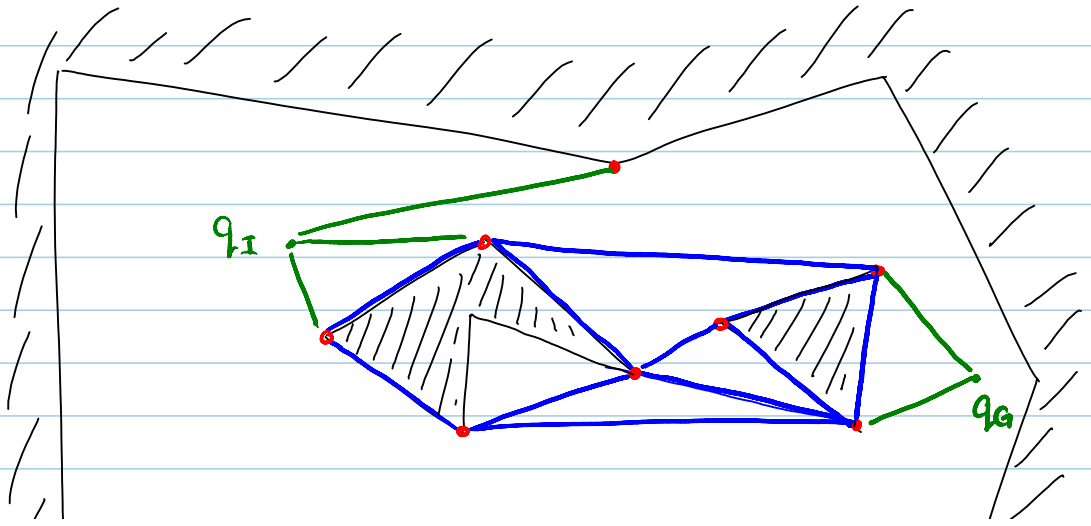
A bitangent edge connects two reflexive vertices that are mutually visible AND extending the bitangent edge beyond the vertices does not intersect $Int(C_{obs})$

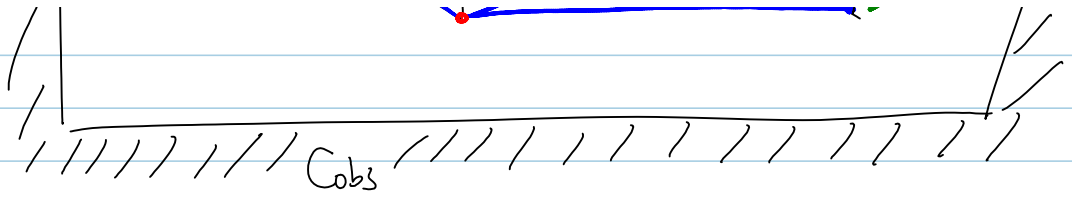


Construct the roadmap

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 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 (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 – $O(n^2)$
} $O(n^2)$

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

∴ overall we have $O(n^3)$

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

and $\boxed{O(n \lg n + m)}$ where m is the # of roadmap edges.